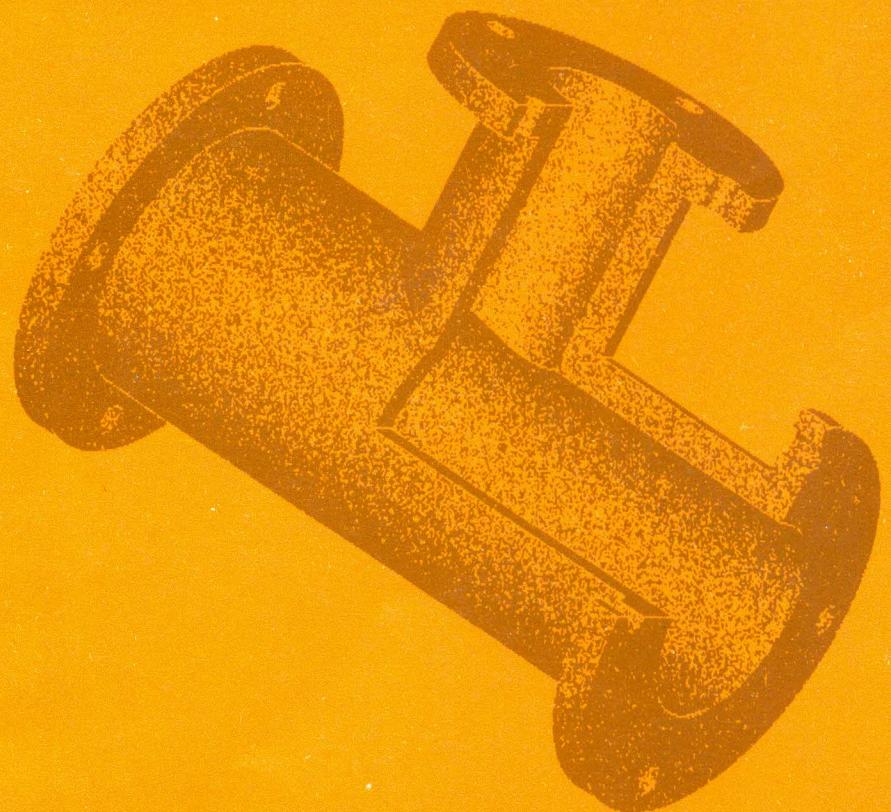


**машиная
графика
и геометрическое
моделирование**



Академия наук СССР Сибирское отделение
Вычислительный центр

МАШИННАЯ ГРАФИКА
И ГЕОМЕТРИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Сборник научных трудов

Под редакцией
А.М.Мацокина



Новосибирск 1990

Сборник посвящен различным аспектам применения средств машинной графики. Все работы сборника можно разбить на три группы: базовые средства машинной графики, машинная графика в САПР машиностроения, машинная графика в географических и картографических приложениях.

The collection of papers is dealt with different aspects of computer graphics application. All the papers may be divided into three groups: basic facilities of computer graphics, computer graphics in CAD of mechanical engineering and computer graphics in geographic and cartographics applications.

© ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР СО АН СССР

Предисловие

Предлагаемый вашему вниманию сборник продолжает серию публикаций о работах сотрудников Лаборатории машинной графики ВЦ СО АН СССР (г. Новосибирск), которой в прошедшем году исполнилось 25 лет. Эти годы были знаменательны созданием в лаборатории таких программных средств, как СМОГ (1972 г.), ГРАФИТ (1982 г.), СПЕЙС (1984 г.), ФОРТ (1987 г.), получивших впоследствии достаточно широкое признание среди специалистов и прикладников. Наряду с постоянным развитием ранее созданных средств все эти годы проводились и экспериментальные работы, непосредственные результаты которых в последующем оказали заметное влияние на системы, получившие коммерческий успех. К их числу можно отнести системы СПО ГД (1980 г.), ПУАССОН (1987 г.), GKS для дисплея Гамма 4.2 (1988 г.).

Статьи настоящего сборника также представляют как уже завершенные, так и находящиеся в развитии и только начатые проекты. Все статьи сборника можно разбить на три группы.

Статьи первой группы (В.А.Дебелова и Ю.А.Ткачева) отражают "классическое" для машинной графики направление в базовом программном обеспечении. В работе В.А.Дебелова описан оригинальный подход к построению графических меню для интерактивных программных комплексов, реализуемых на языках Фортран и Си. Автору удалось преодолеть противоречивость возникающих здесь требований необходимости: а) визуального контроля за процессом и результатами построения меню и б) обеспечения доступа к сгенерированным структурам данных из языков Фортран и Си. В статье Ю.А.Ткачева содержится подробное описание пакета "пассивной" машинной графики GRAPH'ER. Пакет, несмотря на свою функциональную полноту и возможность получать разнообразные графические эффекты, поразительно прост для освоения при работе на IBM PC любых типов. Думается, что обе описанные выше разработки можно смело отнести к числу систем международного уровня.

Вторая группа представленных в сборнике работ посвящена средствам машинной графики для САПР машиностроительного типа.

Многообещающий подход к построению систем обработки машиностроительных чертежей (более точно – систем геометрических построений) развивается в работе В.М.Голубева. В последние годы как за рубежом, так и у нас в стране в этой области наблюдается заметный всплеск сообщений о конкретных разработках. Тем важнее на этом фоне выглядят сформулированные В.М.Голубевым общие принципы и конкретные постановки задач, а также их практическая реализация. По-видимому, на данном этапе большое значение имеют работы, которые можно было бы отнести к области "экспериментального программирования", т.е. практического доказательства возможности и эффективности того или иного подхода к построению систем определенного назначения. Именно так можно оценивать работу Н.С.Шупты, доказавшего "теорему существования" систем моделирования твердых тел, созданную в рамках подхода, получившего у специалистов наименование "конструктивной геометрии".

Четыре работы сборника посвящены новому для лаборатории направлению, связанному с разработками географических (картографических) информационных систем и экспертных систем на их основе. Эти работы в совокупности отражают динамику про-движения в этом направлении. Если статья В.Г.Сиротина и Е.Е. Витяева посвящена описанию одного из проектов в данной области, а статья В.Г.Сиротина и А.В.Коновалова – промежуточным результатам ведущейся разработки географической информационной системы "Альба", то статьи В.И.Торшина и С.А.Уполь-никова посвящены описанию и рассмотрению уже полученных окончательных результатов. В работе В.И.Торшина обобщается опыт построения интерактивной системы, в которой прикладные данные носят преимущественно картографический характер. В статье С.А.Упольникова рассматривается проблема правдоподобного изображения рельефа местности, вводимые при этом предположения, упрощения, используемые алгоритмы и структуры данных.

Заканчивая представление работ сборника, остается сооб-щить, что большинство из них были доложены на проводившейся в конце октября – начале ноября 1989 г. в г. Новосибирске Пятой Всесоюзной конференции по машинной графики, а также по-жалеть об отсутствии полиграфических возможностей проиллюстрировать представленные работы цветными снимками с экранов графических дисплеев. 4

В.А. Дебелов

ОБ ОДНОМ ПОДХОДЕ К РАЗРАБОТКЕ
ИНТЕРАКТИВНЫХ ГРАФИЧЕСКИХ ПРОГРАММ

В связи с переходом на IBM PC перед нами встали две задачи технологического характера:

- традиционная в рамках ВЦ СО АН СССР - адаптация графического пакета СМОГ [1, 2] и создание вокруг него операционной среды для разработки графических программ, применявших по крайней мере вывод на дисплеи и ввод с клавиатуры и мыши, входящих в состав большинства ПЭВМ;
- автоматизация ряда рутинных работ, возникающих при создании интерактивных графических программ и связанных с формированием меню, планированием раскroя экрана, разработкой сценария диалога.

Все описываемое в данной работе программное обеспечение (обозначим его TS) реализовано в виде библиотек подпрограмм и комплекса интерактивных графических программ в рамках MS DOS с использованием MS компиляторов с языков Ассемблера, Фортран-77, Си.

I. Операционная среда

При создании операционной среды мы придерживались стратегии разумной достаточности, учитывая современные тенденции и стандарты с тем, чтобы при необходимости достаточно просто перенести систему TS, т.е. среду, и, следовательно, все по-

строенное в ней программное обеспечение. В систему входят функции вывода на экран графической информации, ввода управляющей информации от пользователя и функции вывода, ввода и редактирования простейших данных языка Фортран - строк, целых и вещественных чисел.

I.I. Модели экрана

Физически, или конструктивно, различаются адаптеры и мониторы дисплеев на IBM PC, которые позволяют получать цветное или монохроматическое изображение, изображения с различной детализацией (растром), разным числом виртуальных экранов.

Растровая модель (или растровый экран). Экран представляется прямоугольной матрицей $N_a \times M_a$ отображаемых пикселов. Для каждого адаптера a N_a и M_a свои, в общем случае пиксель неквадратный. Начало системы координат $(0, 0)$ - в левом верхнем углу экрана, ось X_r направлена слева направо, ось Y_r - сверху вниз, единицы измерения - физические дискреты растра.

Текстовая модель (текстовый экран). В предлагаемой системе возможно несколько растровых шрифтов, каждый из которых f характеризуется литературной площадкой - $L_{nf} \times L_{mf}$ пикселов. Текстовый экран представляется прямоугольной матрицей отображаемых литературных площадок - $N_{fa} \times M_{fa}$. В зависимости от адаптера и шрифта N_{fa} и M_{fa} свои. Адресация (na, nb) литературных площадок: na - номер строки сверху вниз, считая с нуля; nb - номер площадки в строке (столбец) слева направо, считая с нуля.

Непрерывная модель (непрерывный экран). В данной модели считается, что экран - это прямоугольная область размером $P_x \times P_y$ миллиметров. Начало системы координат $(0, 0, 0, 0)$ в левом нижнем углу экрана, ось X_e направлена слева направо, ось Y_e - снизу вверх, единица измерения по осям - миллиметр.

Реальные размеры отображаемой части экрана меняются от одного дисплея к другому, и, более того, используя регулировки, их можно изменять на одном и том же дисплее. Для адаптеров CGA, EGA, VGA мы остановились на значениях

$P_x = 240$ мм, $P_y = 180$ мм для видимого изображения. С одной стороны, это приблизительно соответствует истине, а с другой – соотношение 4 : 3 выбрано исходя из соотношений 640 : 480 для VGA или 800 : 600 для некоторых EGA, имеющих квадратный пиксель. С нашей точки зрения, более удобно планировать экран, если описывать изображения почти в реальных размерах, а не на подобласти квадрата $[0.0, 1.0] \times [0.0, 1.0]$ из R^2 , как в GKS.

Непрерывный и растровый экраны описывают одну и ту же прямоугольную область, совпадают. Для каждого шрифта существует привязка текстового экрана к растровому. Текстовый экран не обязательно покрывает весь растровый. Основой, на которой строятся все модели, является растровая. Непрерывная модель представляет удобство программирования, поскольку не связана с дискретностью конкретных адаптеров и обеспечивает независимость и переносимость программ для достаточно широкого класса применений.

Подобные модели используются и для устройства твердой копии. В основном они имеют внутрисистемное назначение, кроме непрерывной. Непрерывная модель для носителя изображения на устройстве твердой копии будет называться ЛИСТ (термин СМОГ).

В системе TS для построения изображений пользователь может одновременно применять текстовый экран, растровый экран и ЛИСТ, т.е. строить изображение для твердой копии, используя экран как видеоконтрольное устройство. Непрерывная модель экрана используется только для одной цели – для задания на экране поля вывода, соответствующего выбранному на ЛИСТЕ окну. Часто в качестве окна выбирается весь ЛИСТ целиком, например, когда твердая копия не нужна вовсе. Основное назначение окна – выбор для визуального контроля или построения в интерактивном режиме части изображения на ЛИСТЕ. Общий принцип использования различных моделей таков, что части изображения, описываемые при помощи различных моделей, просто добавляются на экране.

Преобразования координат. В каждый момент времени точно известны адаптер, установленный шрифт и ЛИСТ. В системе TS,

как правило, не дублируются функциональные возможности для различных моделей, например, построение залитых полигональных областей может обеспечиваться только в растровой модели. В связи с этим, чтобы с одной стороны не ограничивать пользователя, а с другой - дать ему средства определять взаиморасположение частей изображения, построенных с применением различных моделей, предоставляются процедуры пересчета координат:

- пиксель (nx , ny) растрового экрана в позицию (na , nb) текстового. В качестве результата выдается литерная площадка, на которой расположен тестируемый пиксель. В качестве результата обратной процедуры выдается определенный пиксель литерной площадки;

- пиксель (nx , ny) в точку непрерывного экрана (xe , ye) - центр пикселя и обратно;

- пиксель (nx , ny) в точку ЛИСТА (xp , yp) - центр пикселя и обратно.

Следует помнить, что последовательное применение прямой и обратной процедур перевода, как правило, не дает исходные координаты.

I.2. Виртуальная клавиатура

Общий подход в ряде известных стандартов (GKS, CGI,...) заключается в попытке обобщить все пути ввода данных в программу при помощи сложных - с точки зрения логики их работы - виртуальных устройств ввода. В системе TS нет жесткой фиксации не только устройств, но даже и их классов. Выделившееся к настоящему времени в системе TS основное устройство ввода - виртуальная клавиатура (ВК) - обычно моделируется алфавитно-цифровой клавиатурой. ВК вырабатывает входной поток целых чисел, которые трактуются как коды нажимаемых клавиш. В составе процедур нижнего уровня показательны следующие (имена условные):

- GET1(C) - ввод очередного кода С нажатой клавиши. Все клавиши, в том числе и функциональные, имеют определенные коды за исключением управляющих типа Shift, Control.

- GET2(C, S, P, M) - аналогична предыдущей, позволяет квалифицированно распознать ситуацию на ВК при вводе: S - код сканирования клавиатуры, т.е. некоторый порядковый номер клавиши, в отличие от значения С не зависит от состояния управляющих клавиш; P - код состояния управляющих клавиш, таких, как Shift, Caps и т.п.;

- BACK - возврат во входной поток последнего введенного через СЕТ'ы кода. Операцию GET - BACK над кодом одной и той же клавиши можно делать многократно.

Мышь. Это устройство является еще одной из реализаций ВК, равноправное с клавиатурой. Поскольку мышь обычно используется для задания направления перемещения и в качестве функциональной клавиатуры, то, как ВК, она может выдавать во входной поток: коды движения по четырем (стрелки) или восьми (дополнительно "Home", "End", "PgUp", "PgDn") направлениям, код клавиши "Enter" и состояние кнопок в момент ввода. Только параметр M процедуры GET2 говорит о конкретном устройстве в составе ВК, вызвавшем ввод, и для мыши - состояние кнопок.

Ввод с ВК осуществляется без всякого эха. Считается, что это - задача процедур более высокого функционального уровня. Очевидно, что программы, построенные только на GET1, менее зависимы от конфигурации оборудования, чем построенные на GET2.

В заключение отметим, что ВК не является монопольным владельцем конкретных устройств, на которых она реализуется. Использование мыши в качестве ВК означает только, что программиста устраивает такой способ описания действий с мышью. В рамках системы TS допускается создание процедур, обеспечивающих самостоятельное использование мыши (наряду с ВК), например, процедуры ввода кривых или рукописных знаков.

2. Базовые компоненты программного обеспечения

Система TS на базовом уровне включает пакет вывода на растровый экран GRAPH'ER (подробно описан в [3]) и комплекс процедур обслуживания виртуальной клавиатуры (VK). Над

ними построены остальные уровни. В данном пункте описываются простейшие средства для организации интерактивной работы и средства вывода графической информации.

2.1. Вывод графической информации

Графическое изображение на экране может формироваться:

а) посредством обращений к процедурам пакета СМОГ - рисование на ЛИСТе;

б) посредством обращения к графическим функциям пакета GRAPH'ER - рисование на растровом экране. В обоих случаях соответствующая графическая информация будет добавлена к изображению на экране. В рамках пакета СМОГ предполагается, что установка необходимых режимов делается через GRAPH'ER:

- выбор требуемой дискретности экрана,
- программирование палитры,
- установка режима комбинирования вновь поступающих примитивов и содержимого экрана (OR, AND, XOR, ...),
- установка типа линий и шаблона заливки и т.д.

Другими словами, СМОГ можно рассматривать как надстройку над пакетом GRAPH'ER, обеспечивающую решение таких задач, как построение графиков, изолиний и т.п.

Все построения через процедуры пакета СМОГ ведутся на ЛИСТе. Специальные интерфейсные процедуры определяют, куда будет направляться вырабатываемая графическая информация. Возможны следующие варианты:

- только вывод в метафайл для дальнейшей отрисовки на устройствах получения твердых копий. Игнорируется текущее окно на ЛИСТе и вывод на экран отключен;
- отображение в поле вывода на экране (растровом экране) с выполнением всех необходимых координатных преобразований графической информации, попадающей в окно на ЛИСТе. Вывод в метафайл отключен;
- совмещенный, т.е. вывод в метафайл всей графической информации, формируемой на ЛИСТе, и отображение на экране в текущем поле вывода информации, попадающей в текущее окно на ЛИСТе.

В каждый момент времени может быть только одно преобразование "окно - поле вывода", но разрешается его изменять.

Отметим, что между обращениями к процедурам пакета СМОГ могут выполняться дополнительные построения на экране непосредственно через GRAPH'ER, но в метафайле они не генерируются. Таким образом, работа с экраном может рассматриваться в рамках пакета СМОГ как вспомогательная по отношению к формированию изображения на ЛИСТе - к формированию окончательного документа. С точки зрения пакета GRAPH'ER СМОГ можно рассматривать как генератор графических примитивов.

В рамках прикладной программы имеется возможность добавлять к содержимому изображения на ЛИСТе и/или экране как все изображение, записанное в метафайл, так и его часть. Последнее обеспечивается наличием в метафайле специальных примитивов разметки.

Вывод рисунков из метафайлов на графопостроители и принтеры осуществляется автономными программами системы ТС. Завершается реализация Системы Управления Графическим Выводом (СУГВ) [4], которая работает в интерактивном режиме и позволяет выполнить постобработку изображений, закодированных в метафайлах. Возможности СУГВ - от простого копирования рисунка на устройство до компоновки плаката, используя информацию нескольких метафайлов и аппарат аффинных преобразований, окон и экранов.

2.2. Вывод, ввод и редактирование данных

Ряд процедур служит для вывода на текстовый экран строк, целых и вещественных чисел. Например, оператор

```
CALL PROC1(KG, '\Bес = %LD ka\', NA, NB)
```

выведет в строку NA, начиная с позиции NB, при KG = 153, текст

Бес = 153 кг.

Шаблон вывода задается по правилам языка С. Для редактирования значения KG необходимо обратиться к другой процедуре

```
CALL PROC2 ('\Исправь значение веса = \', KG, INPUT, NA,
```

NB, L, NQ), после выполнения которой в KG будет исправленное значение. Во всех аналогичных процедурах переменная INPUT задает режим ввода или редактирования. Значение L указывает, что при выполнении операции под запись числа нельзя отводить более L литер, и используется для защиты других областей экрана. Параметр NQ – выходной и сообщает номер клавиши, завершившей ввод/редактирование текста или числа. В данном комплекте процедур имеется средство, позволяющее задать набор завершающих клавиш, например, все функциональные. Кроме того можно задать несколько диапазонов запрещенных кодов, которые будут игнорироваться процедурами при вводе. Например, все английские буквы, а для ввода восьмеричных чисел достаточно разрешить только символы 0 – 7. При вводе/редактировании чисел осуществляется синтаксический контроль, например, чтобы в числе не было двух знаков, точки у целого.

Аналогичные процедуры существуют для вывода, ввода и редактирования текстовых строк и вещественных чисел.

Второй вариант процедур – графический – позволяет позиционировать тексты с точностью до пикселя. Различия еще в том, что в алфавитном режиме изображаются целые литерные площадки (фон – своим цветом, символ – своим), а в графическом – только символы. Как показала практика, полезно иметь оба режима отображения, как с точки зрения их вида, так и для планирования раскroя экрана.

2.3. Управление маркером

В системе TS предлагается единственное средство, позволяющее построить эхо или обратную визуальную связь при вводе координатной информации, – это маркер. Маркер отображается в виде крестика на экране. Имеется возможность включать его отображение и выключать, задавать цвет и размеры. Система выполняет размещение маркера в требуемую позицию, при этом маркер из предыдущей позиции автоматически убирается, а изображение, бывшее под ним, восстанавливается. Поддерживается два маркера:

– на растровом экране; позиция задается в системе координат растра. Его проявление не зависит от действий СМОГа;

- на ЛИСТе; подчиняется всем правилам, применяемым к остальным графическим примитивам СМОГа, также клишируется по окну, но в метафайле не помещается.

На основе применения данного механизма и ВК в системе построены средства ввода позиций на растровом экране и ЛИСТе, "резиновые" нить, окружность, многоугольник и др.

2.4. Протоколы

Выше введен графический метафайл СМОГ - это описание моментального снимка, как в стандарте CGM. В GKS метафайл - скорее протокол работы с системой. В системе TS введено два протокола.

Выходной протокол. Создается на уровне обращений к функциям пакета GRAPH'ER. Возможные применения: а) для создания демонстрационных слайд-фильмов, б) в качестве исходной информации для СУГБ.

Входной протокол. Вся работа с ВК в прикладной программе, точнее, все принимаемые коды, протоколируются - формируют входной протокол. В дальнейшем весь этот протокол или его выбранные фрагменты можно подавать в ВК для отработки, другими словами - для подмены действий пользователя. Аналогичные средства применяются в ряде редакторов, например, для создания макроопераций. В нашем случае в качестве основного применения предполагается создание демонстрационных и обучающих копий диалоговых программ. Было бы очень полезно иметь такое же средство на уровне MS DOS или таких управляющих программ, как NORTON COMMANDER. Тогда можно было бы создавать "макросы", а не BAT-файлы на целую серию отдельных задач как системных, так и пользовательских.

3. Меню текстовых кнопок

Техника работы с текстовыми меню применялась при реализации системы СМОГ на мини-ЭВМ Labtam-3215 и перенесена на IBM PC с учетом опыта практической эксплуатации. Логическое меню в системе TS - это прямоугольная матрица $m_{i,j}$ (N, M) текстовых кнопок. Каждая кнопка занимает прямоугольный

бокс из $V_{i,j}$ строк по $U_{i,j}$ литерных площадок, т.е. кнопка – это многострочный текст. На текстовом экране выбирается некоторый прямоугольник (поле меню), на котором размещаются все кнопки. Бокс кнопки привязывается к полю меню. В меню не обязательно должны быть все $N * M$ кнопок. Пустые места матрицы \mathcal{M} задаются как $U_{i,j} = 0$. Отметим, что порядок размещения боксов на поле меню, а следовательно, на экране, не зависит от их взаимного расположения в матрице \mathcal{M} .

Основные операции над меню:

- определить меню в системе. Задается полная информация о меню. Одновременно может быть определено несколько меню;
- вывести меню. Одновременно может быть выведено несколько меню;
- погасить меню;
- изменить место поля меню на экране;
- изменить параметры визуализации меню: цвета линий и текстов, шрифт, необходимость обвода кнопок и всего поля, видимость, когда неактивно. Последнее означает, что меню будет автоматически появляться на экране только на время ввода альтернативы по этому меню и исчезать сразу после завершения ввода. Работа с меню в таком режиме не изменяет изображение на экране. Можно потребовать, чтобы меню присутствовало постоянно;
- обновить изображение кнопки. Применяется при изменении текста кнопки при выведенном меню;
- ввести номер альтернативы при помощи меню. Это основная операция, ради которой и создаются меню. Процедура осуществляется навигацией по кнопкам меню согласно их расположению в матрице \mathcal{M} , а также естественной трактовке клавиш со стрелками и клавиш "Home" (в начало) и "End" (к последней). В отличие от остальных кнопок активная выделяется инверсным изображением. Нажатие "Enter" приводит к формированию результата работы этой операции в виде двузначного номера (i, j) активной кнопки. Если в процессе работы с ВК поступит код клавиши, не используемый для целей навигации по меню, то этот код пропускается в прикладную программу с соответствующим признаком;

- выдать номер активной кнопки. Это можно узнать даже для невысвеченного меню. При гашении и высвечивании активность кнопки к выбору не изменяется;

- установить определенную кнопку в активное состояние.

Такая операция часто применяется, когда задание одной и той же работы (альтернативы) можно выполнить несколькими путями - например, выбор через меню или функциональную клавиатуру. В последнем случае меню все равно должно правильно отражать сложившуюся ситуацию.

Отметим, что процедура навигации по меню и выбора альтернативы реагирует на коды клавиш со стрелками, поступающими с мыши, только в том случае, когда на мыши зажата левая кнопка. Иначе эти коды трактуются как неинтерпретируемые и передаются программе.

Специальный экранный редактор FM - MAKER, входящий в систему TS, позволяет в интерактивном режиме сконструировать изображение меню на экране и получить файл с текстом процедуры на языке С, обращение к которой в прикладной программе определяет меню в системе.

4. Графические панели

Вторая разновидность меню, применяемых в системах TS, - графические панели (GP). GP - это стационарная часть или начальное состояние изображения на экране при решении прикладной программы. Панель, в отличие от меню, создается только при помощи редактора FM - MAKER, позволяющего строить на экране картинки любой сложности из различных типов графических примитивов (см. ниже).

Работая с редактором, программист в диалоговом режиме создает на экране панель управления своей задачей. На этой панели размещаются: различные пиктограммы, которые должны служить как кнопки графического меню, для выбора альтернатив; области, в которых его программа должна строить изображения в процессе счета. Для привязки (или настройки) программы к конкретному GP введен специальный примитив (назовем его пункт), который представляет неотображаемую на

экране и ЛИСТе точку, координаты которой и два определяемых программистом во время работы с редактором целых числовых параметра могут быть получены в программе. Отметим, что использование GP предполагает определенную дисциплину работы, когда программа должна настраиваться на заранее определенную геометрию экрана. Числовые параметры пункта могут трактоваться по усмотрению программиста. Но следует учитывать, что один должен быть положителен и уникален в пределах одного GP, служит номером. Второму дано название типа пункта. Рекомендуется пунктам, имеющим одинаковое назначение, присваивать один тип.

При помощи пунктов можно определять места расположения пиктограмм, свободные области на панели, области, предназначенные для вывода результатов счета в графическом или числовом виде и т.д.

Операции над панелями:

- простое отображение GP как картинки на то место на экране, где его сформировал программист. При желании может быть заполнен программный массив информацией о пунктах: номер, тип, координаты px, py на растровом экране;

- отображение в указанном положении. При отображении GP сдвигается таким образом, чтобы единственный пункт с типом 0 разместился в точке с указанными координатами;

- динамическое отображение. Работа с GP напоминает описанную в п.2.3 для маркера. В очередной раз GP отображается в новой специфицированной позиции, а изображение экрана, которое он затер в предыдущем положении, восстанавливается.

Ряд программных средств в системе TS и поддерживающих их работу примитивов в GP введен специально для организации навигации по пиктограммам панели как по кнопкам меню. При создании, посредством редактора GP, предполагающих такое применение, требуется соблюдать определенную дисциплину при назначении номеров и типов пунктов. При определении в GP очередной пиктограммы как световой кнопки, программист задает, каким образом выделяется эта кнопка будучи активной, путем составления дочерней панели subGP и, если требуется, то в качестве разделов HINT (подсказка) и HELP (краткая или

развернутая помощь) еще ряд subGP, которые в процессе решения программы будут вызываться автоматически.

Таким образом, предлагается технология не программирования, а создания в диалоге при помощи редактора FM-MAKER панели управления задачей, определения в ней световых кнопок, конструирования для них текстовых и графических, возможно, очень объемных, подсказок.

Функциональные возможности системы TS по управлению графическими панелями все время усложняются. Синхронно с этим наращивается мощность редактора FM-MAKER, одна из основных задач которого - дать программисту за терминалом проиграть варианты сеанса, непосредственно наблюдая образ, каким он будет в той или иной ситуации, за исключением изображений, получаемых в процессе счета.

5. Редактор FM-MAKER

Графический экранный редактор FM-MAKER (быстрый составитель меню) сначала разрабатывался только для создания текстовых меню и панелей. Но поскольку оказалось, что с его помощью можно строить достаточно произвольные изображения, его стали применять и как редактор рисунков, например, для системы авторской подготовки публикаций [5].

Для построения изображений применяются только ВК (клавиатура и/или мышь) и визуальная обратная связь. Никакие числовые данные о геометрии не вводятся, поскольку строится визуальный интерфейс. Основные примитивы:

- ломаная;
- прямоугольник (*);
- прямоугольник со скругленными углами (*);
- окружность, дуга окружности (*);
- эллипс (*);
- многоугольник (*);
- текст;
- палитра и др.

Звездочкой помечены примитивы, которые могут быть представлены как только границей, так и с залитой некоторым цветом внутренностью.

Редактирование осуществляется в режимах "резинового" (по отношению к форме) или перемещаемого примитива. Какая-либо характерная точка примитива, например выбранная вершина многоугольника, связывается с маркером, который перемещается под управлением ВК. На экране все время отображается текущая информация, в том числе и координаты маркера. Ввод является разновидностью редактирования формы примитива. Специальные средства позволяют осуществлять операции перемещения, копирования и удаления групп примитивов или подкартинок.

Результатом работы редактора является специфический метафайл. В связи с тем, что каждая панель строится для конкретного адаптера дисплея с его ограничениями на растр, представление цветов, то введен ряд утилит, обеспечивающих автоматический пересчет сформированных GP для других адаптеров. Практика показала, что после пересчета GP обычно не требует редактирования. Исключением являются панели, содержащие круги и окружности, которые в зависимости от степени неквадратичности пикселя раstra могут переводиться в эллипсы.

Автор считает своим долгом выразить благодарность коллегам - сотрудникам Лаборатории машинной графики ВЦ СО АН СССР, тесный контакт с которыми во время разработки и отладки системы (на их программах) в значительной степени определил ее современный вид.

Л и т е р а т у р а

1. Математическое обеспечение граffопостроителей. I уровень.
СМОГ: Инструкция по программированию - Новосибирск: Изд. ВЦ СО АН СССР, 1976.
2. Математическое обеспечение граffопостроителей. 2 уровень.
СМОГ: Инструкция по программированию - Новосибирск: Изд. ВЦ СО АН СССР, 1976.
3. Ткачев Ю.А. Комплекс графических программ Graph'er, -
Наст. сб. - С. 20-59.

4. Дебелов В.А., Русков А.В. Технология графического вывода в крупном ВЦ или сети// Технология математического моделирования. - 1989. - С. 76-81.
5. Дебелов В.А., Плеханов С.А. ФОРТ - графическое программное средство оформления документов// Программные средства машинной графики. - Новосибирск, 1985. - С. 22-40.

Ю.А. Ткачев

КОМПЛЕКТ ГРАФИЧЕСКИХ ПРОГРАММ
GRAPH'ER

В статье описан комплект программ GRAPH'ER, предоставляющий базовые средства построения изображений на экране персональных компьютеров, совместимых с IBM PC и оборудованных графическим адаптером типа CGA, EGA или VGA. От имеющихся графических библиотек, например библиотеки GRAPH.LIB фирмы MICROSOFT [1], комплект GRAPH'ER отличает значительно более широкий набор графических операций и более высокая производительность.

I. Краткое описание графических адаптеров

Все графические адаптеры фирмы IBM совместимы "снизу вверх", т.е. адаптер EGA обеспечивает все режимы CGA, а VGA, в свою очередь, может работать как в режиме CGA так и EGA. В настоящее время базовой системой ввода/вывода поддерживаются следующие режимы:

- 320 x 200 элементов отображения, 4 цвета, 2 фиксированные палитры (CGA);
- 640 x 200 элементов отображения, 2 цвета, 2 фиксированные палитры (CGA);

- 320 x 200 элементов отображения, 16 цветов из 64 возможных (EGA);
- 640 x 200 элементов отображения, 16 цветов из 64 возможных (EGA);
- 640 x 350 элементов отображения, 16 цветов из 64 возможных (EGA);
- 640 x 480 элементов отображения, 16 цветов из 256K возможных (EGA);
- 320 x 200 элементов отображения, 256 цветов из 256K возможных (EGA).

Некоторые графические адаптеры, совместимые с адаптерами фирмы IBM, могут обеспечивать и другие режимы работы:

- 640 x 480 элементов отображения, 16 цветов из 64 возможных (EGA+);
- 800 x 600 элементов отображения, 16 цветов из 64 возможных (EGA+).

В графических режимах каждая точка экрана является непосредственно адресуемой, т.е. цвет каждого пикселя задается независимо от остальных путем изменения значения в соответствующей ячейке графической памяти. Если каждой точке экрана соответствует один бит, то одновременно могут отображаться только два цвета, двум битам соответствуют 4 цвета, четырем битам - 16 цветов и восьми битам - 256 одновременно отображаемых на экране цветов.

С точки зрения программиста, использующего графические возможности ПЭВМ, адаптеры CGA/EGA/VGA [2] представляют собой некоторое количество оперативной памяти, расположенной в адресном пространстве процессора, начиная с адреса A000: 0000 (для EGA/VGA) или B000 : 0000 (для CGA) и кончая адресом BFFF: FFFF. Таким образом, графическая память расположена вне тех 640 кбайт, которые отводятся операционной системе и задачам пользователя. Хотя, как нетрудно видеть, протяженность адресного пространства, отведенного под графическую память, составляет только 128 кбайт, общее количество памяти, установленной на плате адаптера EGA/VGA, может достигать 512 кбайт. Структура графической памяти этих адаптеров такова, что процессор, записывая некоторое значение в определенный байт, од-

новременно может изменить содержимое еще трех байтов. Правила, по которым происходит изменение значений, записанных в ячейках графической памяти (и тем самым изменение изображения на экране монитора), определяются содержимым доступных по записи регистров графического адаптера. В зависимости от режима работы на каждую точку экрана отводится от одного до восьми битов графической памяти. Соответствие пикселей экрана и ячеек графической памяти также определяется режимом работы адаптера и может существенно различаться для разных режимов.

Кроме графической памяти, на плате адаптера имеется несколько (16 для EGA и 256 для VGA) 6- или 18-разрядных регистров, которые устанавливают соответствие между значением, записанным в ячейке памяти, и цветом пикселя на экране. Как известно, цвет каждой точки экрана складывается из трех составляющих: красной, зеленой и синей. Относительная яркость каждой из них для i-го цвета записывается в i-м регистре. Поскольку в адаптере EGA эти регистры являются 6-разрядными (по два разряда на каждую составляющую), то всего возможны 64 различные комбинации основных цветов. Для VGA имеется 256K (2^{18}) различных комбинаций, так как на каждый основной цвет отводится уже по 6 разрядов. Изменяя значения в этих регистрах, можно менять цвета на экране, не меняя содержимого графической памяти. Кроме указанных, имеется еще несколько доступных по записи регистров, которые определяют такие параметры, как число точек в строке экрана, расстояние (в адресном пространстве) между началами соседних строк, число строк на экране, начало отображаемой области графической памяти. Последнее особенно важно. Если в адаптере установлено достаточное количество памяти, то возможна одновременная работа с несколькими графическими "страницами". Разумеется, на экране в каждый момент времени отображается только одна из них, но формировать изображение можно в каждой странице одновременно и независимо от остальных. Визуализация построенного изображения достигается занесением соответствующего значения в регистр адреса начала отображаемой памяти.

2. Общее описание комплекта

Комплект GRAPH'ER представляет собой набор подпрограмм, написанных на языке ассемблера. Передача параметров соответствует соглашениям языка FORTRAN фирмы Microsoft. Все параметры подпрограмм (если явно не оговорено противное) являются целыми двухбайтовыми переменными, длина массивов не должна превышать 64 кбайт. Начало экранной системы координат находится в левом верхнем углу экрана, ось X направлена вправо, ось Y - вниз.

Все множество подпрограмм комплекта можно разбить на несколько групп. В первую включаются программы "физической" установки режима работы адаптера, т.е. программы задания разрешения экрана по осям X и Y, количества одновременно отображаемых цветов, выбора или программирования палитры, задания начального адреса отображаемой области памяти и т.д.

Вторую, самую большую, группу образуют подпрограммы установки "логических" режимов работы и изменения раstra в соответствии с установленными режимами. Растром будем называть прямоугольный массив пикселей, каждый из которых имеет некоторый цвет из диапазона [0-L]. Изменение раstra заключается в изменении цвета составляющих его пикселей. Размеры раstra и количество допустимых цветов определяются режимом работы адаптера. Одновременно может быть доступно несколько растров, один из которых является видимым. Если R_i - растр, то переход к новому состоянию раstra R_i задается следующей функцией:

$$R_i = g(R_i, P, F, M, C_1, C_0, CR, SR_1, SR_2, \dots, SR_k),$$

где P - графический примитив; F - функция преобразования цвета пикселя; C_1 и C_0 - основной и дополнительный цвета; M - маска (линейная или площадная); CR - клиппирующий прямоугольник; SR_1, SR_2, \dots, SR_k - активные экранирующие прямоугольники.

Графический примитив вместе с клиппирующим и активными экранирующими прямоугольниками определяет те пиксели раstra, которые, возможно, поменяют цвет при выполнении преобразова-

ния. А именно, изменяют свое состояние те пиксели раstra, которые, во-первых, принадлежат примитиву Р, во-вторых, лежат внутри клипирующего прямоугольника CR, в-третьих, лежат вне всех активных экранирующих прямоугольников SR_1, SR_2, \dots, SR_k (каждому экранирующему прямоугольнику приписан некоторый приоритет; активными считаются только те, которые имеют приоритет, больший заданного числа).

В комплекте CGRAPH'ER реализованы следующие графические примитивы:

- отрезок;
- окружность;
- прямоугольник;
- треугольник;
- замкнутая ломаная (многоугольник);
- круг;
- залитый прямоугольник;
- залитый треугольник;
- залитый многоугольник;
- массив бинарных матриц.

Последний примитив требует пояснений. Массив бинарных матриц есть упорядоченный набор (не более 256 штук) задаваемых пользователем матриц, элементы которых могут принимать значения 0 или 1. Размер матриц (число строк и столбцов) произведен, но постоянен для всего массива. Одновременно может быть задано до десяти различных массивов бинарных матриц. Кроме того, имеется четыре предопределенных массива, размеры и содержимое которых не может быть изменено. Чтобы вывести матрицу на экран, необходимо указать номер массива, порядковый номер матрицы в этом массиве и координаты точки на экране, которая соответствует левому нижнему углу матрицы. При отработке этого примитива изменяют цвет пиксели раstra, соответствующие тем элементам матрицы, которые равны 1.

С использованием аппарата массивов бинарных матриц в комплекте CGRAPH'ER реализованы процедуры вывода текста. Упомянутые выше четыре предопределенные массива содержат кодировку изображения символов алфавита различных размеров - 8 x 8, 8 x 14, 8 x 19 и 10 x 24 пикселей. Учитывая то, что матрицы могут быть произвольного размера, можно не только создавать

различные шрифты, но и кодировать с помощью матриц достаточно сложные изображения типа пиктограмм и т.п.

Остальные аргументы в приведенной выше формуле — F , C_1 , C_0 и M — задают правило преобразования цвета изменяющихся пикселей.

F определяет функцию преобразования цвета пикселя $\tilde{C} = F(C, C_{01})$, где C — это цвет пикселя в растре R_i , а C_{01} — один из двух заданных цветов — C_0 или C_1 .

В комплекте GRAPH'ER реализованы следующие функции преобразования цвета:

- 'NULL', $F(C, C_{01}) = 0$ для любых C, C_{01} ;
- 'OVERLAY', $F(C, C_{01}) = C_{01}$ для любого C ;
- 'XOR', $F(C, C_{01}) = C(xor)C_{01}$ — побитовая операция 'ИСКЛЮЧАЮЩЕЕ ИЛИ';
- 'AND', $F(C, C_{01}) = C(and)C_{01}$ — побитовая операция 'И';
- 'OR', $F(C, C_{01}) = C(or)C_{01}$ — побитовая операция 'ИЛИ';
- 'TBL', $F(C, C_{01}) = a_{C,C_{01}}$, где A — заданная пользователем матрица размером $k \times k$ (k — число цветов в установленном режиме работы адаптера), явным образом определяющая все возможные комбинации двух цветов — элемент a_{ij} равен цвету, который получается при сложении цвета i с цветом j ;
- 'FUNCTION' — в этом режиме вместо изменения цвета пикселя происходит обращение к процедуре пользователя, которой в качестве параметров передаются координаты пикселя, его цвет в растре R_i и цвет C_{01} .

Какой именно цвет, C_0 или C_1 , будет участвовать в преобразовании пикселя, определяется координатами этого пикселя и установленной маской M . Маски бывают двух типов — линейные и площадные. Линейная маска представляется целой двухбайтовой переменной и может влиять на цвет пикселей при построении линейных графических примитивов: отрезков, окружностей, ломаных. Если j -й бит двоичного представления маски равен 1, то j -й по модулю 16 от начала примитива пиксель в растре R_i будет иметь цвет $\tilde{C} = F(C, C_1)$. В противном случае (т.е. если j -й бит маски равен 0) пиксель получает цвет $F(C, C_0)$.

Площадная маска представляется матрицей размером 8 на 8 бит. Влияние площадной маски на цвет пикселей аналогично вли-

янию линейной, только оно может проявляться при выводе как линейных, так и площадных примитивов (залитых прямоугольников, треугольников и т.д.).

Третью группу подпрограмм комплекта составляют подпрограммы работы с массивами пикселей. Они позволяют:

- запомнить состояние всего раstra или его части в предоставленном пользователем массиве или файле;

- восстановить все запомненное изображение или его часть в том же самом или любом другом месте раstra, возможно, с клиппированием и экранированием относительно установленного клипирующего и активных экранирующих прямоугольников.

Четвертую группу образуют несколько информационных подпрограмм. По запросу пользователя они сообщают такие сведения, как, например, разрешение экрана и число возможных цветов; номер установленного шрифта и его размеры; число доступных графических страниц; количество памяти, необходимой для сохранения изображения указанных размеров и т.д. Кроме того, имеется подпрограмма, с помощью которой можно проконтролировать корректность работы подпрограмм комплекта. При нормальном завершении работы каждая подпрограмма устанавливает в нуль внутреннюю переменную - индикатор ошибки. Ненулевое значение этой переменной свидетельствует о некорректности работы последней вызванной подпрограммы комплекта.

Наконец, в пятую группу входят подпрограммы, которые можно условно назвать "сервисными". Действия, выполняемые ими, могут быть проделаны с помощью нескольких обращений к подпрограммам более низкого уровня (например, из второй группы), но с меньшей эффективностью. Сюда относятся подпрограммы:

- вывода текста в различных режимах;

- запоминания протокола работы комплекта, т.е. последовательности вызова графических подпрограмм с возможностью последующей автоматической отработки всего протокола или некоторой его части; одновременно может вестись несколько протоколов;

- работы с "маркерами"; маркером можно объявить произвольный набор графических примитивов и затем перемещать его по экрану, не затрагивая остального изображения; можно также сде-

лать маркер мерцающим, и тогда он будет то появляться, то исчезать в указанном месте, опять-таки не изменяя изображения на экране.

3. Описание подпрограмм комплекта

В этом пункте приводится полное описание всех подпрограмм комплекта с указанием устанавливаемых ими кодов ошибок. Все подпрограммы устанавливают код ошибки 99, если к ним обратились до перехода в графический режим, т.е. до вызова подпрограммы GrStReg или GrFtReg. Кроме того, если при формировании протокола во внутренней области памяти произошло переполнение, то вызвавшая его подпрограмма устанавливает код ошибки 95.

3.1. Подпрограммы физических установок

3.1.1. GrSmReg - задание режима работы адаптера
Обращение: call GrSmReg(nr)

nr - номер режима работы адаптера:

I - текстовый режим 80 x 25;

2 - 320 x 200 элементов отображения, 4 цвета, 2 фиксированные палитры (CGA);

4 - 640 x 200 элементов отображения, 2 цвета, 2 фиксированные палитры (CGA);

5 - 320 x 200 элементов отображения, 16 цветов из 64 возможных (EGA);

6 - 640 x 200 элементов отображения, 16 цветов из 64 возможных (EGA);

7 - 640 x 350 элементов отображения, 16 цветов из 64 возможных (EGA);

8 - 640 x 480 элементов отображения, 16 цветов из 256K возможных (VGA);

9 - 320 x 200 элементов отображения, 256 цветов из 256K возможных (VGA);

10 - 640 x 480 элементов отображения, 16 цветов из 64 возможных (EGA+);

II - 800 x 600 элементов отображения, 16 цветов из 64 возможных (EGA+);

Коды ошибок:

I - недопустимый номер режима.

Замечания:

I. Обращение к процедуре вызывает очистку экрана и установку стандартной палитры.

2. По умолчанию устанавливаются следующие логические режимы:

основной цвет графики - белый;

дополнительный цвет графики - черный;

маска - отсутствует;

клиппирование - по границам экрана;

экранирование - отсутствует;

функция преобразования цвета пикселей - "OVERLAY";

графическая позиция - левый верхний угол экрана;

цвет текста - белые буквы на черном фоне;

рабочий шрифт - 25 строк по 80 символов;

текстовая позиция - левый верхний угол экрана.

3. Не для каждого адаптера имеется возможность определить, является ли тот или иной режим для него допустимым, поэтому даже нулевой код ошибки не гарантирует последующей нормальной работы комплекта.

3.1.2. GrFtReg - фиктивное задание режима работы адаптера
Обращение: call GrFtReg(nr)

параметр совпадает с параметром процедуры GrStReg.

Коды ошибок:

2 - несоответствие физического и логического режимов работы адаптера.

Замечания:

I. Эта подпрограмма отличается от GrStReg тем, что не меняет физического режима работы графического адаптера (тем самым не очищает экран и не изменяет установленной палитры), а производит только инициализацию внутренних переменных в соответствии с номером устанавливаемого режима.

2. Обращение к GrFtReg целесообразно в двух случаях - при совместной работе нескольких графических пакетов и при использовании порожденных процессов: физическая установка режима производится средствами другого графического пакета или в одном из подпроцессов;

3. Несоответствие физического и логического режимов работы может привести к непредсказуемым последствиям.

3.I.3. GrStPal - выбор или программирование палитры

Обращение: call GrStPal (nc, r, g, b)

для режима 2: nc - номер палитры;

r - фоновый цвет;

для режима 4: r - основной цвет (CGA) или фоновый цвет (EGA/VGA);

для остальных графических режимов:

nc - номер программируемого регистра палитры;

r - относительное количество красной составляющей;

g - относительное количество зеленой составляющей;

b - относительное количество синей составляющей.

Коды ошибок:

3 - недопустимый номер палитры или программируемого регистра;

4 - недопустимое количество какой-либо составляющей.

Замечания:

1. Допустимые (для установленного режима работы адаптера) значения параметров можно узнать, обратившись к процедуре GrGtFPar.

2. По умолчанию установлено следующее соответствие между номером цвета и его представлением на экране:

0 - черный;

1 - синий;

2 - зеленый;

3 - голубой;

4 - красный;

5 - фиолетовый;

6 - коричневый;

7 - светло-серый;

8 - темно-серый;

9 - ярко-синий;

10 - ярко-зеленый;

I1 - ярко-голубой;
I2 - ярко-красный;
I3 - ярко-фиолетовый;
I4 - желтый;
I5 - белый.

3. Обращение к процедуре GrStReg с любым параметром влечет установку стандартной палитры.

3.1.4. GrVisPg - выбор видимой на экране графической страницы

Обращение: call GrVisPg (np)

np - номер страницы.

Коды ошибок:

5 - недопустимый номер страницы.

Замечание:

Число доступных (для установленного режима работы адаптера) графических страниц можно узнать, обратившись к процедуре GrGtFPar.

3.1.5. GrStExm - разрешение использования расширенной памяти

Обращение: call GrStExm (itp)

itp - тип расширенной памяти:

1 - extended;

2 - expanded.

Коды ошибок:

6 - не удалось обнаружить дополнительную (extended) память;

7 - драйвер EMS не установлен (expanded).

Замечания:

1. Дополнительная память может использоваться комплектом GRAPH'ER для запоминания массивов пикселей (см. подпрограммы GrCtRI, GrCtRIC) и хранения протокола работы комплекта GrStExm.

2. Допускается одновременное использование как extended, так и expanded памяти - для этого необходимо дважды обратиться к GrStExm.

3.2. Подпрограммы логических установок и работы с растром

3.2.1. GrStPg - установка рабочей графической страницы (растра)

Обращение: call GrStPg (np)
np - номер страницы.

Коды ошибок:

5 - недопустимый номер страницы.

Замечания:

1. Все последующие обращения к подпрограммам работы с растром будут изменять состояние только рабочей графической страницы, никак не затрагивая остальных.

2. Число доступных (для установленного режима работы адаптера) графических страниц можно узнать, обратившись к процедуре GrGtFPar.

3.2.2. Установка основного и дополнительного цветов

Обращение: call GrStCol (nc)
call GrStBCol (nc)

nc - номер основного или дополнительного цвета.

Коды ошибок:

10 - недопустимый номер цвета.

Замечания:

1. Установленные основной и дополнительный цвета действуют для всех графических страниц.

2. Влияние основного и дополнительного цветов на результирующий цвет пикселей раstra описано в п. 2.

3. Число доступных цветов (для установленного режима работы адаптера) можно узнать, обратившись к процедуре GrGtFPar.

3.2.3. GrStMsk - установка одной из ранее заданных масок

Обращение: call GrStMsk (nm)

nm - номер рабочей маски; если nm < 0, то рабочей становится площадная маска с номером abs(nm), в противном случае устанавливается линейная маска; если nm = 0, то все последующие графические примитивы будут выводиться на экран без маски.

Коды ошибок:

II - недопустимый номер маски.

Замечания:

I. Установленная маска действует для всех графических страниц.

2. Влияние площадной и линейной масок на результирующий цвет пикселей раstra описано в п. 2.

3. Число доступных масок можно узнать, обратившись к процедуре **GrGtLPar**.

4. Задание линейных масок осуществляется подпрограммой **GrLiMsk**, а площадных - **GrBoMsk**.

3.2.4. **GrLiMsk** - задание линейной маски.

Обращение: call **GrLiMsk** (**nm**, **msk**)

nm - номер задаваемой линейной маски;

msk - маска.

Коды ошибок:

II - недопустимый номер маски.

Замечания:

I. Влияние линейной маски на результирующий цвет пикселей раstra описано в п. 2.

2. Число доступных масок можно узнать, обратившись к процедуре **GrGtLPar**;

3. Заданная маска автоматически становится рабочей.

Пример:

если **msk** = 61680, то маска имеет вид: IIII0000III0000.

3.2.5. **GrBoMsk** - задание площадной маски

Обращение: call **GrBoMsk** (**nm**, **msk**)

nm - номер задаваемой площадной маски;

msk - массив длиной 8 байт, в котором содержится описание задаваемой маски.

Коды ошибок:

II - недопустимый номер маски.

Замечания:

I. Влияние площадной маски на результирующий цвет пикселей раstra описано в п. 2.

2. Число доступных масок можно узнать, обратившись к процедуре **GrGtLPar**;

3. Первый байт соответствует первой строке маски, второй байт - второй строке и т.д.;

4. Заданная маска автоматически становится рабочей.

Пример:

если элементы массива имеют следующие значения: 24, 60, I26, 255, 255, I26, 60, 24 то будет задана маска:

```
0 0 0 1 1 0 0 0  
0 0 1 1 1 1 0 0  
0 1 1 1 1 1 1 0  
1 1 1 1 1 1 1 1  
1 1 1 1 1 1 1 1  
0 1 1 1 1 1 1 0  
0 0 1 1 1 1 0 0  
0 0 0 1 1 0 0 0
```

3.2.6. GrStFun - установка функции преобразования цвета пикселя

Обращение: call GrStFun (nf)

nf - номер устанавливаемой функции преобразования:

0 - 'XOR';

1 - 'NULL';

2 - 'OVERLAY';

3 - 'OR';

4 - 'AND';

5 - 'TBL';

6 - 'FUNCTION'.

Коды ошибок:

I2 - недопустимый номер функции.

Замечания:

1. Влияние функции преобразования на результирующий цвет пикселей растра описано в п. 2.

2. Число доступных функций можно узнать, обратившись к процедуре GrGtLPar.

3. Функции 'TBL' и 'FUNCTION' становятся доступными только после обращения к подпрограммам GrTBL и GrFUN соответственно.

3.2.7. GrTBL - задание матрицы преобразования цвета пикселя

Обращение: call GrTBL (arr)

arr - байтовый массив, задающий преобразование.

Замечания:

1. Длина массива должна составлять $k \times k$ байт, k - число доступных цветов.

2. Так как содержимое массива не копируется, а запоминается только его адрес, не рекомендуется использовать занимаемую массивом arr память для других нужд.

3. Заданный массив преобразования оказывает влияние на цвет пикселей только в том случае, если установлена функция преобразования 'TBL' (см. подпрограмму GrStFun).

3.2.8. GrFUN - задание подпрограммы преобразования цвета пикселей

Обращение: call GrFUN (ns, subr)

ns - номер формата подпрограммы пользователя;

subr - имя подпрограммы пользователя, обращение к которой будет происходить всякий раз при изменении состояния растра.

Замечания:

1. В настоящей версии пакета реализован только один формат пользовательской подпрограммы ($ns = 1$), а именно subr (nx, ny, c, c01), где на входе (nx, ny) - координаты изменяемого пикселя, c - его текущий цвет, а c01 - основной или дополнительный цвет; на выходе из подпрограммы c должно равняться новому значению цвета или -1, если состояние пикселя менять не надо.

2. В своей подпрограмме пользователь волен совершать любые действия, в том числе обращаться ко всем подпрограммам пакета GRAPH'ER .

3. Обращение к заданной подпрограмме преобразования происходит только в том случае, если установлена функция преобразования 'FUNCTION' (см. подпрограмму GrStFun).

3.2.9. GrClRect - задание клиппирующего прямоугольника

Обращение: call GrClRect (nx, ny, mx, my)

(nx, ny) - координаты левого верхнего угла клиппирующего прямоугольника;

(mx, my) - координаты правого нижнего угла клиппирующего прямоугольника.

Замечания:

1. По умолчанию установлено клиппирование по границам растра.

2. Клиппирование производят только те подпрограммы комплекта, в названии которых присутствуют буквы CS, например, GrPixCS, GrPlgnCS и т.п.

3.2.10. GrScRect - задание экранирующего прямоугольника

Обращение: call GrScRect (nx, ny, mx, my, np)

(nx, ny) - координаты левого верхнего угла экранирующего прямоугольника;

(mx, my) - координаты правого нижнего угла экранирующего прямоугольника;

np - приоритет.

Коды ошибок:

I3 - задано слишком много экранирующих прямоугольников.

Замечания:

1. Если не задано ни одного прямоугольника, экранирование не производится.

2. Экранирование производят только те подпрограммы комплекта, в названии которых присутствуют буквы CS, например, GrPixCS, GrPlgnCS и т.п., и относительно только тех экранирующих прямоугольников, которые в данный момент активны, т.е. имеют достаточно высокий приоритет (см. подпрограмму GrActSc).

3. Приоритеты у разных прямоугольников могут совпадать.

3.2.11. GrDelSc - удаление экранирующих прямоугольников

Обращение: call GrDelSc (np)

np - приоритет удаляемых экранирующих прямоугольников.

Коды ошибок:

I4 - нет ни одного прямоугольника с указанным приоритетом.

Замечания:

I. Удаляются все экранирующие прямоугольники, имеющие указанный приоритет.

2. Эта процедура действительно удаляет экранирующие прямоугольники, поэтому для активации и дезактивации экранов лучше пользоваться подпрограммой GrActSc.

3.2.12. GrActSc - активизация экранирующих прямоугольников
Обращение: call GrActSc (np)

np - минимальный приоритет активизируемых экранирующих прямоугольников.

Замечание:

Активизируются (т.е. начинают влиять на вывод графических примитивов) все экранирующие прямоугольники, имеющие приоритет, больший или равный np.

3.2.13. GrPrBin - задание массива бинарных матриц

Обращение:

narr - номер задаваемого массива;

dx - размер матриц по оси X;

xs, ys - сдвиг матриц при выводе относительно начала координат;

yb - расстояние от верха матрицы до ее базовой линии;

nx - число байт, приходящихся на кодировку одной строки одной матрицы;

ny - число строк в матрице;

nb - число байт в массиве arr, приходящихся на кодировку изображения одной матрицы;

arr - байтовый массив, в котором содержится кодировка всех 256 бинарных матриц.

Коды ошибок:

I5 - недопустимый номер задаваемого массива;

I6 - слишком много байт, приходящихся на кодировку одной матрицы.

Замечания:

I. Количество одновременно заданных массивов и максимальное число байт в кодировке одной матрицы можно узнать, обратившись к процедуре GrGtIPar.

2. Для параметров подпрограммы должны выполняться следующие соотношения: $dx \leq nx*8$, $nb = nx*ny$.

3. Параметры x_s и y_s имеют смысл только при использовании подпрограмм вывода текста по литературным площадкам экрана.

4. Так как содержимое массива не копируется, а запоминается только его адрес, не рекомендуется использовать занимаемую массивом arr память для других нужд.

5. Изменение содержимого и особенно размеров матриц первых четырех массивов (т.е. имеющих номера с 0 по 3) может привести к непредсказуемым последствиям.

3.2.14. GrStBin - установка рабочего массива бинарных матриц

Обращение: call GrStBin (nm)

nm - номер рабочего массива бинарных матриц.

Коды ошибок:

I5 - недопустимый номер массива;

I7 - массив не был инициализирован.

Замечания:

1. Устанавливаемый массив должен иметь номер меньше четырех или должен быть ранее инициализирован подпрограммой GrPrBin.

2. Все подпрограммы комплекта, работающие с массивами бинарных матриц, в том числе подпрограммы вывода текста, выводят на экран матрицы именно рабочего массива.

3. По умолчанию установлен один из предопределенных массивов, а именно тот, при котором на экране при выводе текста получается в точности 25 строк по 80 символов в каждой.

4. Все параметры рабочего массива можно узнать, обратившись к подпрограмме GrGtBPar.

3.2.15. GrPix, GrPixCS - изменение состояния одного пикселя

Обращение: call GrPix (nx, ny)

call GrPixCS (nx, ny)

nx, ny - координаты изменяемого пикселя.

Замечания:

1. Принципы, в соответствии с которыми пикслю будет присвоен цвет, описаны в п. 2.

2. Подпрограмма GrPix отличается от CrPixCS тем, что изме-

няет состояние указанного пикселя вне зависимости от установленного клиппирующего и активных экранирующих прямоугольников.

3.2.16. GrSegm, GrSegmCS - отрезок прямой

Обращение: call GrSegm (nx, ny, ind)

call GrSegmCS (nx, ny, ind)

nx, ny - координаты конца отрезка; ind - если ind = 0, то отрезок не рисуется, но текущая позиция перемещается в точку с координатами (nx, ny); в противном случае рисуется отрезок из текущей позиции до точки (nx, ny).

Замечания:

1. При отсутствии маски или при установленной площадной маске отрезок рисуется всегда сверху вниз.

2. При обращении к GrSegm и GrSegmCS с ind = 0 установленная линейная маска сбрасывается на начало.

3. Подпрограмма GrSegm отличается от GrSegmCS тем, что рисует отрезок вне зависимости от установленного клиппирующего и активных экранирующих прямоугольников.

3.2.17. GrCirc, GrCircSC, GrFCirc, GrFCircCS - окружность или круг

Обращение: call GrCirc (nx, ny, r)

call GrCircCS (nx, ny, r)

call GrFCirc (nx, ny, r)

call GrFCircCS (nx, ny, r)

nx, ny - координаты центра;

r - радиус.

Замечания:

1. Для большинства режимов работы адаптера нарисованный круг на экране выглядит как эллипс.

2. GrFCirc и GrFCircCS рисуют заливый круг.

3.2.18. GrRect, GrRectCS, GrFRect, GrFRectCS - прямоугольник

Обращение: call GrRect (nx, ny, mx, my)

call GrRectCS (nx, ny, mx, my)

call GrFRect (nx, ny, mx, my)

call GrFRectCS (nx, ny, mx, my)

nx, ny - координаты левого верхнего угла прямоугольника;
mx, my - координаты правого нижнего угла прямоугольника.

Замечание:

GrFRect и GrFRectCS рисуют заливой прямоугольник.

3.2.19. GrTria, GrTriaCS, GrFTria, GrFTriaCS - треугольник

Обращение: call GrTria (x1, y1, x2, y2, x3, y3)

call GrTriaCS (x1, y1, x2, y2, x3, y3)

call GrFTria (x1, y1, x2, y2, x3, y3)

call GrFTriaCS (x1, y1, x2, y2, x3, y3)

x1, y1, x2, y2, x3, y3 - координаты вершин треугольника.

Замечание:

GrFTria и GrFTriaCS - рисуют заливой треугольник.

3.2.20. GrPlgn, GrPlgnCS, GrFPPlgn, GrFPPlgnCS - многоугольник

Обращение: call GrPlgn (n, arr)

call GrPlgnCS (n, arr)

call GrFPPlgn (n, arr)

call GrFPPlgnCS (n, arr)

n - число вершин;

arr(2, n) - массив координат вершин.

Замечание:

GrFPPlgn и GrFPPlgnCS рисуют заливой многоугольник.

3.2.21. GrBin, GrBinCS, GrFBin, GrFBinCS - бинарная матрица

Обращение: call GrBin (nx, ny, nmat)

call GrBinCS (nx, ny, nmat)

call GrFBin (nx, ny, nmat)

call GrFBinCS (nx, ny, nmat)

nx, ny - координаты точки привязки;

nmat - номер матрицы в рабочем массиве бинарных матриц

Замечание:

GrFBin и GrFBinCS рисуют заливую матрицу - те пиксели, которые соответствуют нулевым элементам, получают цвет в зависимости от установленного дополнительного цвета.

3.3. Подпрограммы работы с прямоугольными массивами пикселей

3.3.1. GrGtRP, GrGtRU, GrGtRI, GrGtRF - запоминание массива пикселей

Обращение: call GrGtRP (nx, ny, mx, my, arr)

call GrGtRU (nx, ny, mx, my, arr)

call GrGtRI (nx, ny, mx, my, narr)

call GrGtRF (nx, ny, mx, my, filename)

nx, ny, mx, my - координаты левого верхнего и правого нижнего углов запоминаемого изображения;

arr - имя байтового массива, в котором запоминается изображение;

narr - номер внутреннего массива, в котором запомнено изображение (выходной параметр);

filename - символьная переменная или константа, задающая имя файла, в котором запоминается изображение.

Коды ошибок:

20 - подпрограмма GrGtRI не смогла найти достаточно места в памяти для запоминания изображения;

21 - подпрограмма GrGtRF не смогла открыть файл с указанным именем.

Замечания:

1. Подпрограмма GrGtRP запоминает изображение в массиве пользователя в упакованном виде; формат хранения изображения и потребная длина массива существенно зависят от установленного режима работы адаптера; количество байт, необходимое для запоминания изображения указанных размеров, можно узнать, обратившись к подпрограмме GrRstrSiz.

2. Подпрограмма GrGtRU запоминает изображение в массиве пользователя таким образом, что на каждый пиксель отводится в точности один байт, в котором записывается цвет этого пикселя; первые восемь байт массива содержат координаты левого верхнего и правого нижнего углов запомненного в массиве изображения.

3. В настоящей реализации комплекта GRAPH*ER для подпрограмм GrGtRP и GrGtRU имеется ограничение - нельзя запомнить изображение, занимающее больше 64 кбайт памяти.

4. Подпрограмма GrGtRI сначала запрашивает у операционной системы необходимое для запоминания изображения количество памяти и затем сохраняет изображение в выделенном месте в некотором формате; в программу пользователя передается порядковый номер (идентификатор) области памяти, в которой хранится изображение; этот идентификатор может затем использоваться, например, при выводе изображения на экран или при освобождении занимаемой им памяти (см. подпрограмму GrRstrFree).

3.3.2. GrGtRPC, GrGtRUC, GrGtRIC, GrGtRFC - запоминание массива пикселей с клиппированием

Параметры и назначение этих подпрограмм полностью совпадают с параметрами и назначением подпрограмм предыдущего пункта. Отличие заключается в том, что запоминается только та часть изображения, которая попадает внутрь установленного клиппирующего прямоугольника.

3.3.3. GrStRP, GrSTRU, GrStRI, GrStRF - восстановление всего ранее запомненного изображения в том же месте растра

Обращение: call GrStRP (arr)
call GrSTRU (arr)
call GrStRI (narr)
call GrStRF (filename)

arr - имя массива, содержащего изображение;
narr - идентификатор области памяти, содержащей изображение;

filename - имя файла, содержащего изображение.

Коды ошибок:

22 - неверный идентификатор области памяти;

21 - подпрограмма GrStRF не смогла открыть файл с указанным именем.

Замечание:

Подпрограмма GrSTRU предназначена, в основном, для вывода на экран подготовленных самим пользователем изображений, все остальные подпрограммы этой группы корректно работают только с теми изображениями, которые были ранее запомнены подпрограммами GrGtRP, GrGtRI, GrGtRF и GrGtRPC, GrGtRIC, GrGtRFC.

3.3.4. GrStrPCS, GrStrUCS, GrStrICS, GrStrECS – восстановление всего ранее запомненного изображения в том же месте раstra с клиппированием и экранированием

Параметры и назначение этих подпрограмм полностью совпадают с параметрами и назначением подпрограмм предыдущего пункта. Отличие заключается в том, что восстановление изображения осуществляется с учетом клиппирующего и активных экранирующих прямоугольников.

3.3.5. GrStrP1, GrStrU1, GrStrI1, GrStrF1 – восстановление всего ранее запомненного изображения в новом месте раstra

Обращение: call GrStrP1 (nx, ny, arr)

call GrStrU1 (nx, ny, arr)

call GrStrI1 (nx, ny, narr)

call GrStrF1 (nx, ny, filename)

nx, ny – новые координаты левого верхнего угла восстанавливаемого изображения;

arr – имя массива, содержащего изображение;

narr – идентификатор области памяти, содержащей изображение;

filename – имя файла, содержащего изображение.

Коды ошибок:

22 – неверный идентификатор области памяти;

21 – подпрограмма GrGtrF1 не смогла открыть файл с указанным именем.

3.3.6. GrStrPCS1, GrStrUCS1, GrStrICS1, GrStrECS1 – восстановление всего ранее запомненного изображения в новом месте раstra с клиппированием и экранированием

Параметры и назначение этих подпрограмм полностью совпадают с параметрами и назначением подпрограмм предыдущего пункта. Отличие заключается в том, что восстановление изображения осуществляется с учетом клиппирующего и активных экранирующих прямоугольников.

3.3.7. GrSTRP2, GrSTRU2, GrSTRI2, GrSTRF2 – восстановление части ранее запомненного изображения в том же месте раstra

Обращение: call GrSTRP2 (nx, ny, mx, my, arr)

call GrSTRU2 (nx, ny, mx, my, arr)

call GrSTRI2 (nx, ny, mx, my, narr)

call GrSTRF2 (nx, ny, mx, my, filename)

nx, ny, mx, my – координаты левого верхнего и правого нижнего углов восстанавливаемой части изображения;

arr – имя массива, содержащего изображение;

narr – идентификатор области памяти, содержащей изображение;

filename – имя файла, содержащего изображение.

Коды ошибок:

22 – неверный идентификатор области памяти;

21 – подпрограмма GrGTRF2 не смогла открыть файл с указанным именем.

3.3.8. GrSTRPCS2, GrSTRUCS2, GrSTRICCS2, GrSTRFCS2 – восстановление части ранее запомненного изображения в том же месте раstra с клиппированием и экранированием

Параметры и назначение этих подпрограмм полностью совпадают с параметрами и назначением подпрограмм предыдущего пункта. Отличие заключается в том, что восстановление изображения осуществляется с учетом клиппирующего и активных экранирующих прямоугольников.

3.3.9. GrSTRP3, GrSTRU3, GrSTRI3, GrSTRF3 – восстановление части ранее запомненного изображения в новом месте раstra

Обращение: call GrSTRP3 (nx, ny, mx, my, kx, ky, arr)

call GrSTRU3 (nx, ny, mx, my, kx, ky, arr)

call GrSTRI3 (nx, ny, mx, my, kx, ky, narr)

call GrSTRF3 (nx, ny, mx, my, kx, ky, filename)

nx, ny, mx, my – координаты левого верхнего и правого нижнего углов восстанавливаемой части изображения;

кx, ку - координаты точки, в которую будет помещен левый верхний угол восстанавливаемой части изображения;
arr - имя массива, содержащего изображение;
narr - идентификатор области памяти, содержащей изображение;
filename - имя файла, содержащего изображение.

Коды ошибок:

22 - неверный идентификатор области памяти;
21 - подпрограмма GrGtRF3 не смогла открыть файл с указанным именем.

3.3.I0. GrStRPCS3, GrSTRUCS3, GrSTRICS3, GrStrFCS3 - восстановление части ранее запомненного изображения в новом месте раstra с клиппированием и экранированием

Параметры и назначение этих подпрограмм полностью совпадают с параметрами и назначением подпрограмм предыдущего пункта. Отличие заключается в том, что восстановление изображения осуществляется с учетом клиппирующего и активных экранирующих прямоугольников.

3.3.II. GrMvRstr - пересылка всего изображения из одной графической страницы в другую

Обращение: call GrMvRstr (pg1, pg2)

pg1 - номер графической страницы, из которой пересылается изображение;
pg2 - номер графической страницы, в которую пересылается изображение.

Коды ошибок:

5 - недопустимый номер страницы.

3.3.I2. GrRstrFree - освобождение занятой изображением области памяти

Обращение: call GrRstrFree (narr)

идентификатор освобождаемой области памяти.

Коды ошибок:

22 - неверный идентификатор области памяти.

Замечание:

Освободить можно только ту область памяти, которая ранее была занята подпрограммами GrGtRI или GrGtRIC.

3.4. Информационные подпрограммы

3.4.1. GrGtFPar - значения физических параметров

Обращение: call GrGtFPar (nreg, mpar)

nreg - номер интересующего режима работы адаптера или -1;

mpar - массив возвращаемых параметров:

mpar(1) - номер режима работы адаптера;

mpar(2) - разрешение экрана по оси X;

mpar(3) - разрешение экрана по оси Y;

mpar(4) - число одновременно отображаемых на экране цветов;

mpar(5) - допустимое количество основного цвета при программировании палитры;

mpar(6) - число доступных графических страниц;

mpar(7) - номер видимой на экране графической страницы;

mpar(8) - mpar(10)-резерв.

Замечание:

Если nreg = -1, то массив mpar заполняется значениями, соответствующими установленному режиму работы адаптера.

3.4.2. GrGtLPar - значения логических параметров

Обращение: call GrGtLPar (mpar)

mpar - массив возвращаемых параметров:

mpar(1) - число доступных линейных масок;

mpar(2) - число доступных площадных масок;

mpar(3) - число доступных функций преобразования цвета пикселя;

mpar(4) - максимальное число одновременно заданных массивов бинарных матриц;

mpar(5) - максимальное число байт в кодировке одной бинарной матрицы;

mpar(6)-mpar(10) - резерв.

3.4.3. GrGtSPar - значения установленных логических параметров

Обращение: call GrGtSPar(mpar)

mpar - массив возвращаемых параметров:

mpar(1) - номер режима работы адаптера;

mpar(2) - основной цвет;

mpar(3) - дополнительный цвет;

`mpar(4)` - номер функции преобразования цвета пикселя;
`mpar(5)` - номер маски;
`mpar(6)` - номер рабочего массива бинарных матриц;
`mpar(7)` - текущая X-координата;
`mpar(8)` - текущая Y-координата;
`mpar(9)` - цвет символов;
`mpar(10)` - цвет символьной площадки;
`mpar(11)` - номер текущей строки;
`mpar(12)` - номер текущего столбца;
`mpar(13)-mpar(20)` - резерв.

3.4.4. `GrGtBPar` - значения параметров массива бинарных матриц

Обращение: `call GrGtBPar (narr, mpar)`

`narr` - номер интересующего массива или -1;

`mpar` - массив возвращаемых параметров:

`mpar(1)` - номер массива;
`mpar(2)` - ширина бинарных матриц массива;
`mpar(3)` - высота бинарных матриц массива;
`mpar(4)` - сдвиг матрицы по оси X относительно начала координат при использовании подпрограмм вывода текста;
`mpar(5)` - сдвиг матрицы по оси Y относительно начала координат при использовании подпрограмм вывода текста;
`mpar(6)` - расстояние от верха до базовой линии матрицы;
`mpar(7)` - число строк на экране;
`mpar(8)` - число символов в строке;
`mpar(9)-mpar(10)` - резерв.

Замечание:

Число строк на экране и число символов в строке зависят как от размеров матриц, так и от установленного режима работы адаптера.

3.4.5. `GrRsrSiz` - число байт, необходимых для запоминания изображения

Обращение: `call GrRstSiz (nx, ny, mx, my, nb)`

`nx, ny, mx, my` - координаты левого верхнего и правого нижнего угла запоминаемого изображения;

nb - число байт, необходимых для запоминания изображения.

Замечание:

GrRstrSiz сообщает длину массива, необходимого для запоминания изображения подпрограммами GrGtRI или GrGtRIC, зависящую от режима работы адаптера, размеров и расположения на экране запоминаемого изображения.

3.4.6. GrGtErr - код ошибки

Обращение: call GrGtErr (*nerr*)

nerr - код ошибки, установленный последней подпрограммой комплекта GRAPH'ER.

3.5. Сервисные подпрограммы

3.5.1. Подпрограммы вывода текста

3.5.1.1. GrStTCol - установка цвета символа и символьной площадки

Обращение: call GrStTCol (*ncs*, *ncf*)

ncs - цвет символа;

ncf - цвет символьной площадки.

Коды ошибок:

10 - недопустимый цвет.

Замечание:

Установленные цвета действуют только при выводе текста подпрограммами GrTxt, GrATxt, GrTxtO, GrTxtO.

3.5.1.2. GrStTRos - установка текстовой позиции

Обращение: call GrStTPos (*nlin*, *nrow*)

nlin - номер строки;

nrow - номер столбца;

Коды ошибок:

30 - недопустимая текстовая позиция.

Замечание:

Число строк и столбцов для установленного режима работы адаптера и рабочего массива бинарных матриц можно узнать, обратившись к подпрограмме GrGtBPar.

3.5.I.3. GrGtTPos - возврат текстовой позиции

Обращение: call GrGtTPos (nlin, nrow)

nlin - номер строки;

nrow - номер столбца.

Замечание:

Эта подпрограмма сообщает установленную текстовую позицию.

3.5.I.4. GrTxt, GrTxtO - вывод текстовой строки

Обращение: call GrTxt (str)

call GrTxtO (str)

str - выводимая строка.

Замечания:

1. Подпрограммы GrTxt и GrTxtO выводят строки, начиная с установленной текстовой позиции.

2. По выходе из подпрограмм текстовая позиция перемещается на конец выведенной строки.

3. Подпрограмма GrTxt выводит строку, ограниченную произвольными одинаковыми символами, например, '/ЭТО СТРОКА/' или '?ЭТО НЕ ТА СТРОКА?'.

4. Стока, выводимая подпрограммой GrTxtO, должна заканчиваться нулевым символом.

3.5.I.5. GrTxt, GrATxtO - вывод текстовой строки

Обращение: call GrTxt (nlin, nrow, str)

call GrATxtO (nlin, nrow, str)

nlin - номер строки;

nrow - номер столбца;

str - выводимая строка.

Замечания:

1. Подпрограммы GrATxt и GrATxtO выводят строку, начиная с текстовой позиции (nlin, nrow).

2. По выходе из подпрограмм текстовая позиция перемещается на конец выведенной строки.

3. Подпрограмма GrATxt выводит строку, ограниченную произвольными одинаковыми символами, например, '/ЭТО СТРОКА/' или '?ЭТО НЕ ТА СТРОКА?'.

4. Стока, выводимая подпрограммой GrATxtO, должна заканчиваться нулевым символом.

3.5.2. Подпрограммы работы с графическим протоколом

Под графическим протоколом понимается последовательность вызовов подпрограмм комплекта GRAPH'ER. Протокол может сохраняться в трех видах: в массиве пользователя, во внутренней области памяти и в файле. Весь запомненный протокол или некоторая его часть могут быть автоматически отработаны, т.е. повторены вызовы всех подпрограмм с теми же значениями параметров, с какими подпрограммы вызывались во время формирования протокола. Одновременно и независимо друг от друга могут вестись несколько протоколов. Кроме запоминания последовательности вызовов графических подпрограмм, пользователь может записать в протокол так называемый протокольный маркер – поименованный набор целых чисел произвольной длины. Имеются подпрограммы отработки протокола, которые, встретив протокольный маркер, передают в программу пользователя его имя, длину и значения, интерпретация которых возлагается на самого пользователя.

Никак не отражаются в графическом протоколе вызовы следующих подпрограмм комплекта: GrStReg, GrFtReg, GrTBL, GrFUN, GrClRect, GrScRect, GrDelSc, GrActSc, GrPrBin, GrGtRP, GrGtRU, GrStRP, GrStRU, GrStRP1, GrStRU1, GrStRP2, GrStRU2, GrStRP3, GrStRU3, GrGtFFPar, GrGtLPar, GrGtSPar, GrGtBPar, GrRstrSiz, GrGtErr.

При отработке протокола или его части вывод графических примитивов производится относительно установленного в этот момент клиппирующего и активных экранирующих прямоугольников.

3.5.2.1. GrPrtBU, GrPrtBI, GrPrBF – начать новый графический протокол

Обращение: call GrPrtBU (arr)

call GrPrtBI (narr)

call GrPrtBF (filename, filedesc)

arr – имя массива пользователя, в котором будет запоминаться формируемый протокол;

narr – идентификатор внутренней области памяти, в которой будет запоминаться формируемый протокол (выходной параметр);

`filename` - имя файла, в котором будет запоминаться формируемый протокол;

`filedesc` - дескриптор файла протокола (выходной параметр).
Коды ошибок:

31 - слишком много одновременно открытых протоколов;

32 - подпрограмма `GrPrtBI` не смогла найти достаточное количество памяти;

33 - подпрограмма `GrPrtBF` не смогла открыть файл с указанным именем.

Замечания:

1. Пользователь должен сам заботиться о том, чтобы не произошло выхода за границы массива `arr`; длину запомненного в массиве протокола можно узнать, обратившись к подпрограмме `GrPrtSiz`.

2. Подпрограмма `GrPrtBI` запрашивает у операционной системы необходимое количество памяти и потом сама следит за наличием свободного места; в программу пользователя передается идентификатор этой области памяти (целое двухбайтовое число), которое затем может использоваться, например, при отработке протокола или при освобождении занимаемой протоколом области памяти (см. подпрограмму `GrPrtFree`).

3. Если `GrPrtBF` смогла открыть указанный файл протокола, то она передает в программу пользователя дескриптор файла (целое двухбайтовое число); все последующие операции с этим протоколом будут производиться с помощью именно дескриптора файла, а не имени.

4. С момента начала протокола и до тех пор, пока он не будет закрыт или приостановлен, в нем будут фиксироваться обращения ко всем подпрограммам комплекта, кроме перечисленных выше.

3.5.2.2. `GrPrtEU`, `GrPrtEI`, `GrPrtEF` - закончить формирование графического протокола

Обращение: `call GrPrtEU (arr)`

`call GrPrtEI (narr)`

`call GrPrtEF (filedesc)`

`arr` - имя массива пользователя с закрываемым протоколом;

narr - идентификатор внутренней области памяти, в которой
содержится закрываемый протокол;
filedesc - дескриптор файла протокола.

Коды ошибок:

- 34 - массив arr не содержит графического протокола;
- 35 - неверный идентификатор внутренней области памяти;
- 36 - неверный дескриптор файла протокола.

Замечание:

Закрытый протокол нельзя продолжить.

3.5.2.3. GrPrtSU, GrPrtSI, GrPrtSF - приостановить формирование протокола

Обращение: call GrPrtSU (arr)

call GrPrtSI (narr)

call GrPrtSF (filedesc)

arr - имя массива пользователя с приостанавливаемым протоколом;

narr - идентификатор внутренней области памяти, в которой
содержится приостанавливаемый протокол;

filedesc - дескриптор файла приостанавливаемого протокола.

Коды ошибок:

- 34 - массив arr не содержит графического протокола;
- 35 - неверный идентификатор внутренней области памяти;
- 36 - неверный дескриптор файла протокола.

Замечания:

1. После приостановки протокола прекращается запись в него
вызываемых процедур комплекта и протокольных маркеров.

2. Формирование приостановленного протокола можно возобновить,
обратившись к одной из подпрограмм GrPrtCU, GrPrtCI,
GrPrtCF.

3.5.2.4. GrPrtCU, GrPrtCI, GrPrtCF - продолжить формирование протокола

Обращение: call GrPrtCU (arr)

call GrPrtCI (narr)

call GrPrtCF (filedesc)

arr - имя массива пользователя с приостановленным протоколом;

narr - идентификатор внутренней области памяти, в которой
содержится приостановленный протокол;

filedesc - дескриптор файла приостановленного протокола.

Коды ошибок:

34 - массив *arr* не содержит графического протокола;

35 - неверный идентификатор внутренней области памяти;

36 - неверный дескриптор файла протокола.

3.5.2.5. *GrPrtMrkU*, *GrPrtMrkI*, *GrPrtMrkF* - записать про-
токольный маркер в указанный протокол

Обращение: call *GrPrtMrkU* (*arr*, *mrk*, *ne*, *mrkarr*)

call *GrPrtMrkI* (*narr*, *mrk*, *ne*, *mrkarr*)

call *GrPrtMrkF* (*filedesc*, *mrk*, *ne*, *mrkarr*)

arr - имя массива пользователя с протоколом;

narr - идентификатор внутренней области памяти, в которой
содержится протокол;

filedesc - дескриптор файла протокола;

mrk - идентификатор протокольного маркера - целое двух-
байтовое число;

ne - количество элементов маркера;

mrkarr(ne) - двухбайтовый целый массив, содержащий элемен-
ты маркера.

Коды ошибок:

34 - массив *arr* не содержит графического протокола;

35 - неверный идентификатор внутренней области памяти;

36 - неверный дескриптор файла протокола;

37 - указанный протокол закрыт или приостановлен.

3.5.2.6. *GrPrtMrk* - записать протокольный маркер во все
открытые протоколы

Обращение: call *GrPrtMrk* (*mrk*, *ne*, *mrkarr*)

mrk - идентификатор протокольного маркера - целое двух-
байтовое число;

ne - количество элементов маркера;

mrkarr(ne) - двухбайтовый целый массив, содержащий элемен-
ты маркера.

3.5.2.7. GrPrtOAU, GrPrtOAI, GrPrtOAF - отработать весь протокол, не обращая внимания на встретившиеся протокольные маркеры

Обращение: call GrPrtOAU (arr)

call GrPrtOAI (narr)

call GrPrtOAF (filedesc)

arr - имя массива пользователя с отрабатываемым протоколом;

narr - идентификатор внутренней области памяти, в которой содержится отрабатываемый протокол;

filedesc - дескриптор файла отрабатываемого протокола.

Коды ошибок:

34 - массив arr не содержит графического протокола;

35 - неверный идентификатор внутренней области памяти;

36 - неверный дескриптор файла протокола.

Замечания:

1. Обращение к подпрограммам GrPrtOAI и GrPrtOAF будет запомнено во всех открытых на этот момент протоколах, в том числе в отрабатываемом.

2. Прежде чем отрабатывать закрытый протокол из файла, необходимо этот файл открыть, обратившись к подпрограмме GrPrtFOpn.

3.5.2.8. GrPrtOMU, GrPrtOMI, GrPrtOMF - отработать протокол с текущей позиции до следующего протокольного маркера

Обращение: call GrPrtOMU (arr, mrk, ne, mrkarr)

call GrPrtOMI (narr, mrk, ne, mrkarr)

call GrPrtOMF (filedesc, mrk, ne, mrkarr)

arr - имя массива пользователя с отрабатываемым протоколом;

narr - идентификатор внутренней области памяти, в которой содержится отрабатываемый протокол;

filedesc - дескриптор файла отрабатываемого протокола;

mrk - идентификатор встретившегося протокольного маркера - целое двухбайтовое число, передаваемое в программу пользователя;

ne - количество элементов встретившегося маркера;

mrkarr(ne) - двухбайтовый целый массив, в который будут записаны элементы встретившегося маркера.

Коды ошибок:

- 34 - массив arr не содержит графического протокола;
- 35 - неверный идентификатор внутренней области памяти;
- 36 - неверный дескриптор файла протокола.

Замечания:

1. Если при отработке дошли до конца протокола, то ne устанавливается равным -1, значения mrk и mrkarr при этом не определены.

2. Прежде чем отрабатывать часть закрытого протокола из файла, необходимо этот файл открыть, обратившись к подпрограмме GrPrtFOpn.

3. По выходе из подпрограмм текущая позиция сдвигается на конец встретившегося протокольного маркера.

4. Установка позиции может быть осуществлена обращением к подпрограммам следующего пункта.

3.5.2.9. GrPrtPosU, GrPrtPosI, GrPrtPosF - установка позиции или сброс на начало протокола

Обращение: call GrPrtPosU (arr, nmrk, mrk, ne, mrkarr)

call GrPrtPosI (narr, nmrk, mrk, ne, mrkarr)

call GrPrtPosF (filedesc, nmrk, mrk, ne, mrkarr)

arr - имя массива пользователя с отрабатываемым протоколом;

narr - идентификатор внутренней области памяти, в которой содержится отрабатываемый протокол;

filedesc - дескриптор файла отрабатываемого протокола;

nmrk - число маркеров, которые требуется пропустить без отработки протокола, или -1; mrk - идентификатор встретившегося nmrk-го протокольного маркера - целое двухбайтовое число, передаваемое в программу пользователя;

ne - количество элементов встретившегося маркера;

mrkarr(ne) - двухбайтовый целый массив, в который будут записаны элементы встретившегося маркера.

Коды ошибок:

- 34 - массив arr не содержит графического протокола;

- 35 - неверный идентификатор внутренней области памяти;

- 36 - неверный дескриптор файла протокола.

Замечания:

1. Если `nmrk` = -1, то происходит сброс на начало протокола.
2. Если при отработке дошли до конца протокола, то не устанавливается равным -1, значения `mrk` и `mrkarr` при этом не определены.
3. Прежде чем установить позицию в закрытом протоколе из файла, необходимо этот файл открыть, обратившись к подпрограмме `GrPrtFOpn`.
4. По выходе из подпрограмм текущая позиция сдвигается на конец встретившегося `nmrk`-го протокольного маркера.
5. Эти подпрограммы не производят отработку протокола, т.е. не выводят на экран встретившиеся графические примитивы, не изменяют режимов и т.д.

3.5.2.I0. `GrPrtFOpn` – открытие сформированного ранее файла графического протокола

Обращение: `call GrPrtFOpn (filename, filedesc)`

`filename` – имя файла, в котором содержится графический протокол;

`filedesc` – дескриптор файла протокола (выходной параметр).

Коды ошибок:

31 – слишком много одновременно открытых протоколов;

33 – нет файла с указанным именем.

3.5.2.II. `GrPrtSiz` – длина сформированного в массиве протокола

Обращение: `call GrPrtSiz (arr, len)`

`arr` – имя массива пользователя, в котором записан протокол;

`len` – количество двухбайтовых слов, занятых под протокол (выходной параметр).

Коды ошибок:

34 – массив `arr` не содержит графического протокола.

3.5.2.I2. `GrPrtFree` – освобождение занимаемой протоколом внутренней области памяти

Обращение: `call GrPrtFree(narr)`

`narr` - идентификатор освобождаемой внутренней области памяти.

Коды ошибок:

35 - неверный идентификатор внутренней области памяти.

Замечание:

Освободить можно только ту область памяти, которая ранее была занята под хранение графического протокола.

3.5.3. Подпрограммы работы с графическими маркерами

Под графическим маркером понимается произвольная последовательность обращений к подпрограммам комплекта GRAPH'ER, не-посредственно влияющих на состояние растра, т.е. подпрограмм установки логических режимов (основного и дополнительного цветов, функции преобразования цвета пикселей, маски) и вывода примитивов. Кроме того, каждый графический маркер обладает именем, приоритетом и координатами точки привязки. Пользователь может, меняя координаты точки привязки, перемещать маркер по экрану, не затрагивая сформированного изображения.

Графические примитивы, составляющие маркер, клипируются относительно того клипирующего прямоугольника, который был установлен во время формирования маркера и экранируются теми экранирующими прямоугольниками, которые имеют приоритет выше, чем приоритет маркера, независимо от того, являются эти экраны активными или нет.

3.5.3.1. GrMrkBeg - начать формирование графического маркера

Обращение: `call GrMrkBeg (mrkname, mrkpr, mrkx, mrky)`

`mrkname` - идентификатор маркера - целая двухбайтовая переменная (выходной параметр);

`mrkpr` - приоритет формируемого маркера;

`mrkx, mrky` - координаты точки привязки.

Коды ошибок:

40 - слишком много одновременно заданных маркеров.

Замечания:

I. Сообщаемый подпрограммой идентификатор маркера будет использоваться при всех действиях с этим маркером - перемещении, удалении, задании режима мерцания и т.д.

2. `mrkx`, `mrky` задают координаты точки привязки на момент формирования маркера – все последующие перемещения маркера будут заключаться в изменении координат составляющих маркер примитивов на вектор $[(\text{newx}-\text{mrkx}), (\text{newy}-\text{mrky})]$, где $(\text{newx}, \text{newy})$ – новые координаты точки привязки.

3. Сформированный маркер появляется на экране только после обращения к подпрограмме `GrMrkVis`.

3.5.3.2. `GrMrkEnd` – закончить формирование маркера
Обращение: call `GrMrkEnd (mrkname)`

`mrkname` – идентификатор графического маркера.

Коды ошибок:

41 – неверный идентификатор маркера;
42 – маркер уже сформирован.

Замечание:

Перемещение, удаление, визуализация, задание мерцания и т.д. допустимы только для сформированного маркера.

3.5.3.3. `GrMrkDel` – удаление графического маркера
Обращение:

`mrkname` – идентификатор удаляемого маркера.

Коды ошибок:

41 – неверный идентификатор маркера;
43 – маркер еще не сформирован.

Замечание:

Подпрограмма не только стирает маркер с экрана, но и освобождает занимаемую им память, так что дальнейшая работа с этим маркером становится невозможной.

3.5.3.4. `GrMrkVis` – визуализация графического маркера
Обращение: call `GrMrkVis (mrkname)`

`mrkname` – идентификатор маркера.

Коды ошибок:

41 – неверный идентификатор маркера;
43 – маркер еще не сформирован;
44 – подпрограмма не смогла найти достаточного количества памяти для сохранения изображения под маркером.

Замечание:

При возникновении ошибки 44 можно попытаться освободить память, занимаемую ненужными массивами пикселей (`grRstrFree`) или графическими протоколами (`GrPrtFree`).

3.5.3.5. GrMrkHid - стирание графического маркера

Обращение: call `GrMrkHid (mrkname)`

`mrkname` - идентификатор маркера.

Коды ошибок:

41 - неверный идентификатор маркера;

43 - маркер еще не сформирован.

Замечания:

1. Маркер просто стирается с экрана, но не удаляется.

2. Стереть с экрана можно как обычный маркер, так и мерцающий.

3. Рекомендуется стирать все маркеры при массовом выводе примитивов на экран, например, при отработке графического протокола - это значительно ускорит работу подпрограмм.

3.5.3.6. GrMrkGli - задание мерцания маркера

Обращение: call `GrMrkGli (mrkname, it1, it2)`

`mrkname` - идентификатор маркера;

`it1` - время, в течение которого маркер видим на экране;

`it2` - время, в течение которого маркер невидим на экране.

Коды ошибок:

41 - неверный идентификатор маркера;

43 - маркер еще не сформирован.

Замечания:

1. Времена `it1, it2` задаются в 1/18 с.

2. Для мерцающего маркера допустимо использовать подпрограммы `GrMrkVis` и `GrMrkHid`.

3. Отмена мерцания маркера достигается заданием `in2=0`.

3.5.3.7. GrMrkMov - сдвиг маркера

Обращение: call `GrMrkMov (mrkname, newx, newy)`

`mrkname` - идентификатор маркера;

`newx, newy` - новые координаты точки привязки маркера.

Коды ошибок:

4I - неверный идентификатор маркера;

43 - маркер еще не сформирован.

Замечание:

Сдвигать можно как видимый, так и невидимый в данный момент маркер.

Л и т е р а т у р а

1. Microsoft C RUN-Time Library Reference: Document N 410840017 - 500 - R04 - 0887. Pt N 048-014-104/Microsoft Corporation; IBM; UNIX. - S.1. - 1984-1987. - 102 p. (App. P.643-687).
2. IBM Enhanced Graphics Adapter: Personal Computer / IBM. - 1984. - 169 p.