



Академия наук СССР Сибирское отделение
В ы ч и с л и т е л ь н ы й ц е н т р

МАШИННАЯ ГРАФИКА И ЕЕ ПРИЛОЖЕНИЯ

(Сборник научных трудов)

Под редакцией
А.М. Мацокина



Новосибирск 1983

Содержание

Предисловие.....	4
Дебелов В.А., Мацокин А.М., Чубарев А.И. Подход к разработке распределенных систем для двухмашинных комплексов.....	5
Чубарев А.И. Средства программирования сателлита в рамках СПО ГД.....	23
Торшин В.И. Программное управление графическими устройствами АРМ.....	45
Вильданов Р.Н. Реализация программного обеспечения АРМ в СПО ГД.....	51
Дебелов В.А., Плеханов С.А. Вывод формульной информации на графопостроитель.....	57
Сиротин В.Г. Язык ГРАФИТ для описания, редактирования, хранения и визуализации моделей двумерных геометрических объектов и их чертежей.....	63
Шупта Н.С. Интерактивный интерпретатор входного языка пакета прикладных программ ГРАФИТ.....	92
Голубев В.М. Проставление размерной и технологической информации на машиностроительных чертежах....	101
Голубев В.М., Савров Ю.А., Сиротин В.Г., Токарев В.П., Чурюмов Б.С., Шупта Н.С. Некоторые практические аспекты применения средств машинной графики в системе автоматизации технологической подготовки производства.....	108
Упольников С.А. Алгоритмы конструирования моделей трехмерных объектов на ЭВМ.....	115
Потапов В.П., Вылегжанин В.Н., Витковский Э.И. Система машинной графики для автоматизированного проектирования элементов шахтных полей.....	136

Предисловие

Тематика большинства статей предлагаемого вниманию читателя сборника определяется проблемой построения распределенной системы машинной графики для САПР. В первой работе описывается один из возможных подходов к построению распределенных систем, взятый за основу при проектировании программного обеспечения графического диалога СПО ГД для двухмашинного комплекса. В следующих трех статьях нашли свое отражение некоторые проблемы, возникающие при разработке программных средств СПО ГД на сателлите.

Статья Дебелова В.А. и Плеханова С.А. была получена на графопостроителе с помощью программ, описываемых в этой работе, и является иллюстрацией автоматизации процесса подготовки документации.

Вторая группа представленных в сборнике работ связана с решением задачи построения моделей двумерных геометрических объектов и их редактирования в интерактивном режиме. В одной из статей анализируется опыт эксплуатации САПР режущего инструмента, использующего описание в работах сборника программные средства.

В статье Упольникова С.А. сделана попытка формализации представления и обработки трехмерных объектов, на основе которой предложены алгоритмы, реализующие теоретико-множественные операции над полиэдрами.

Большое методологическое значение имеет последняя работа сборника, в которой для создания САПР элементов шахтных полей применяются несколько систем машинной графики.

А.М. Мацокин

В.А.Дебелов, А.М.Мацокин, А.И.Чубарев

ПОДХОД К РАЗРАБОТКЕ РАСПРЕДЕЛЕННЫХ СИСТЕМ ДЛЯ
ДВУХМАШИННЫХ КОМПЛЕКСОВ

Эффективное применение автоматизированных рабочих мест (АРМ) для решения сложных практических задач во многих случаях возможно лишь с использованием высокопроизводительных ЭВМ. В связи с этим возникает задача создания распределенного программного обеспечения. Один из подходов к решению этой проблемы описывается в настоящей работе на примере системы программного обеспечения графического диалога СПО ГД [1,2]. Исходным положением к созданию СПО ГД было следующее. Имеются две ЭВМ, сопряженные аппаратно, в каждой из которых в составе штатного программного обеспечения содержится по библиотеке модулей, выполняющих конкретные алгоритмы обработки информации (см. рис. I).

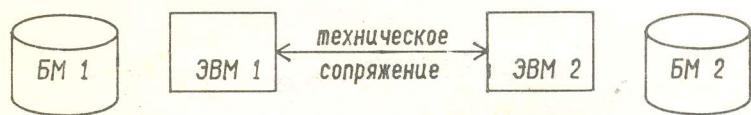


Рис.1. Исходное положение

Одна из ЭВМ – ЭВМ1 – обладает большой вычислительной мощностью и значительными ресурсами оперативной и вторичной памяти. Вторая ЭВМ имеет более скромные характеристики по скорости и памяти, но обладает богатым набором интерактивных

средств либо некоторых специализированных процессоров. В настоящее время такие двухмашинные конфигурации встречаются довольно часто в САПР и системах автоматизации научных исследований. Например, ЭВМ1 – это БЭСМ-6 или ЕС ЭВМ, а ЭВМ2 – М-400, СМ-3, СМ-4, входящие в состав автоматизированного рабочего места АРМ-Р или АРМ-М.

При разработке программного обеспечения (ПО) для двухмашинной конфигурации встали следующие первоочередные задачи, которые надлежало решать в комплексе:

1. Обеспечение канала связи между ЭВМ.
2. Обеспечение средств вызова, синхронизации работы модулей, расположенных на разных ЭВМ.
3. Разработка технологии программирования двухмашинных модулей.

Данные задачи были решены при помощи создания логического протокола передачи управляющей информации и данных между ЭВМ и соответствующим программным обеспечением. ПО связи позволяет сформировать и передать данные в другую ЭВМ, опросить наличие данных, переданных с другой ЭВМ, встать на их ожидание, получить их (подробнее см. [3, 4]). Рассмотрим, какие структуры двухмашинных программ позволяет создавать разработанное базовое обеспечение.

На рис. 2 представлена структура двухмашинной программы, которую можно назвать "центральная обработка". Здесь показан режим работы, когда выполнение программы (сценарий работы) на двухмашинном комплексе определяется на ЭВМ1.

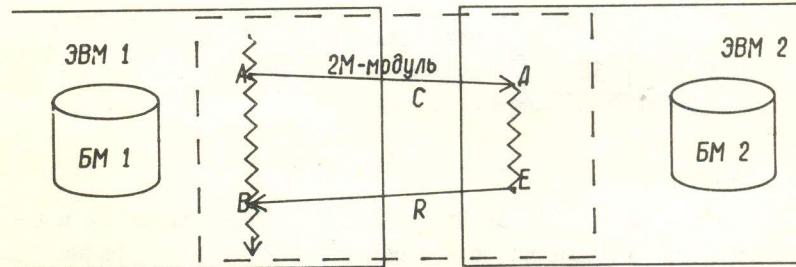


Рис.2. Схема центральной обработки

На рисунке под библиотекой одномашинных (1М) модулей понимается обычная одномашинная библиотека, построенная на основе штатных средств операционной системы (ОС), соответствующей ЭВМ. 1М-модуль получается трансляцией подпрограммы и может использовать все штатные средства ОС, в том числе и другие модули библиотеки. Двухмашинный модуль (2М-модуль) выделен пунктиром и распределен по обеим ЭВМ, он строится на основе средств ПО связи. Обе компоненты 2М-модуля обозначены пилообразной линией (ось времени направлена вниз). Выделены точки взаимодействия А, В, Д, Е. Это точки, в которых используется ПО связи. Дуга С обозначает запуск компоненты 2М-модуля на ЭВМ2, а дуга Р – сообщение в ЭВМ1 об окончании работы компоненты ДЕ. Все линейные участки 2М-модуля (... , А), (А, В), (В, ...), (Д, Е) строятся только с использованием тех функций, которые запрограммированы в виде модулей и находятся в библиотеках соответствующих ЭВМ. Поэтому, если существующего программного опыта на ЭВМ недостаточно для построения 2М-модуля, то библиотеки должны пополняться требуемыми модулями. Понятие 2М-модуля можно расширить, позволяя несколько раз обращаться из ЭВМ1 к разным (в общем случае) компонентам на ЭВМ2.

Как видно из рис. 3, в режиме центральной обработки 2М-модуль состоит из одной компоненты на ЭВМ1 и нескольких, в общем случае разных, компонент на ЭВМ2.

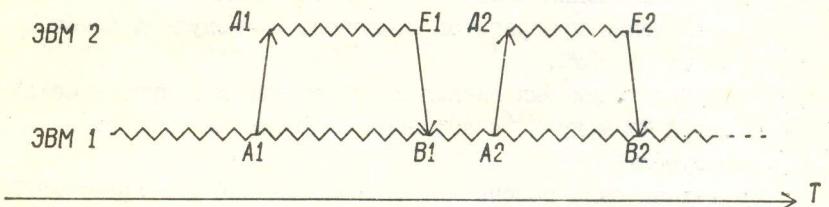


Рис.3. Взаимодействие компонент
2М-модуля во времени

Отметим, что приведенная структура 2M-модуля предполагает наличие параллельной обработки компонент по времени: $(A_1, B_1) \parallel (D_1, E_1)$, $(A_2 \cdot B_2) \parallel (D_2 \cdot E_2)$ на рис. 3. Средства ПО связи позволяют узнать о наличии информации, переданной с другой ЭВМ, поэтому точки B_1, B_2, \dots (рис. 3) при работе компоненты 2M-модуля на ЭВМ1 могут быть определены довольно точно. Это сокращает непроизводительные простой ЭВМ1 и/или позволяет выполнить как можно больше работ во время работы компоненты на ЭВМ2.

В постановке задачи было отмечено, что ЭВМ2 имеет интерактивные средства и, таким образом, приспособлена к диалоговой работе с человеком – будем в дальнейшем называть его абонентом. На схеме рис. 2 нигде не отмечено наличие абонента. Это действительно так: поскольку любой диалог с ЭВМ предполагает соответствующую программную поддержку, то в схеме рис. 2 диалог может вестись во время работы компоненты (Д, Е). Если рассмотреть временную диаграмму, представленную на рис. 3, то видно, что процессор ЭВМ2 будет определенное время простаивать. В случае диалогового характера работы будет простаивать и абонент. По этим соображениям было введено понятие автономного процесса на ЭВМ2, т.е. процесса, который выполняет 1M-модули (возможно диалоговые) из библиотеки 2 по заданиям абонента. Временная диаграмма загрузки ЭВМ в этом случае показана на рис. 4. Здесь, как и выше, пилообразной линией обозначены компоненты 2M-модуля. При реализации такого программного обеспечения необходимо решить задачи

- соотношения приоритетов компонент 2M-модуля и автономного процесса на ЭВМ2;
- синхронизации выполнения и разрешения конфликтов между компонентой 2M-модуля и автономным процессом по доступу к общим ресурсам.

Эти задачи были решены при разработке СПО ГД, автономный процесс имеет более высокий приоритет (см. [4]). Описанная схема предполагает, что ЭВМ2 представлена в одноконсольном варианте (для одного абонента). К недостаткам реализации СПО ГД следует отнести отсутствие у абонента средств смены упомянутых приоритетов.

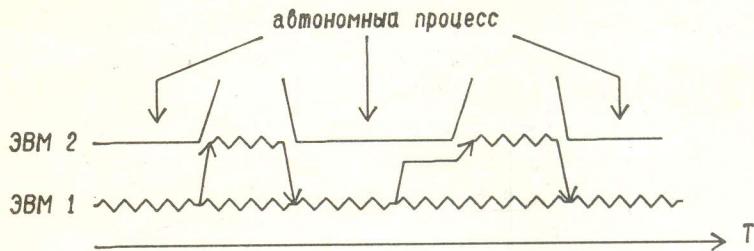


Рис.4. Взаимодействие при наличии автономного процесса

По изложенной выше схеме строятся многие сателлитные графические системы, например, такие, как в [5, 6, 7]. Отличия в их системной организации заключаются в том, насколько богатый набор функций предоставляется абоненту автономным процессом и насколько просто этот набор расширять. В системе СПО ГД набор функций на сателлите ЭВМ2 (АРМ-Р или АРМ-М) расширяется путем пополнения библиотеки ИМ-модулей.

В случае, когда ЭВМ2 является многоконсольной, диаграмму для ЭВМ2 на рис. 4 можно повторить для каждой из консолей. В этом случае компонента 2М-модуля на ЭВМ1 должна строиться как многотерминальная задача, а ПО связи должно обеспечивать инициацию компоненты на ЭВМ2 для нужной консоли. Программные средства поддержки такой мультипроцессорной среды на ЭВМ2 в рамках СПО ГД в настоящее время не реализованы.

Рассмотрим режим работы двухмашинной программы, который получил у нас название терминальной обработки (см. рис. 5).

Эта схема ничем не отличается от изложенной выше, за исключением номеров ЭВМ. Но, согласно постановке задачи, ЭВМ2 обладает интерактивными средствами, и считается, что она в основном работает в диалоге с абонентом. ЭВМ1 в этом случае дополняет вычислительную мощность сателлита и расширяет его возможности по ресурсам памяти (архивы, базы данных).

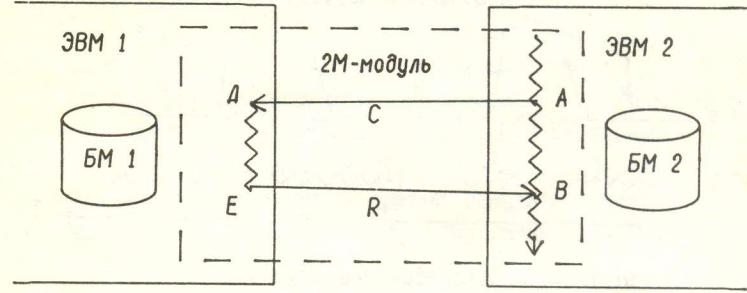


Рис.5. Схема терминальной обработки

Отметим, что в данном режиме отсутствует автономный процесс, 2M-модуль управляет как диалогом с абонентом, так и работой на ЭВМ.

Совмещенный режим – это такой режим, когда в каждый момент времени tandem ЭВМ работает либо в режиме центральной, либо терминальной обработки. Другими словами, эти два режима могут чередоваться последовательно во времени (см. рис. 6).

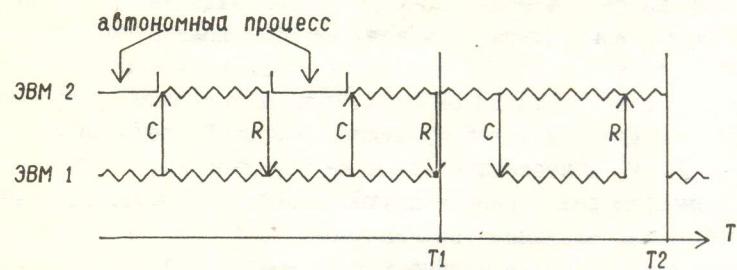


Рис.6. Совмещенный режим

Отметим, что в совмещенном режиме необходимо решить задачу обеспечения перехода из одного режима в другой – моменты времени T_1 и T_2 на рис. 6. Для решения этой задачи на уровне автономного процесса было разрешено осуществлять запуск (переход в) режима центральной обработки (см. рис. 7). При этом

запрещается делать такой запуск, если система уже находится в этом режиме. Выход из него автоматически возвращает систему в автономный процесс. Неверная диаграмма приводится на рис.8.

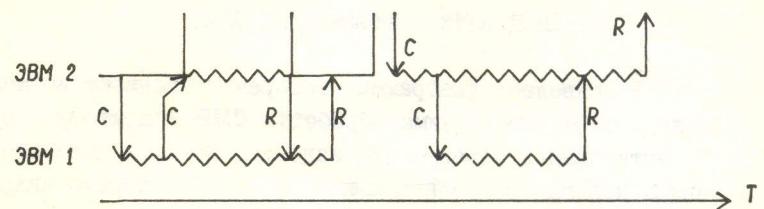


Рис.7. Запуск автономным процессом

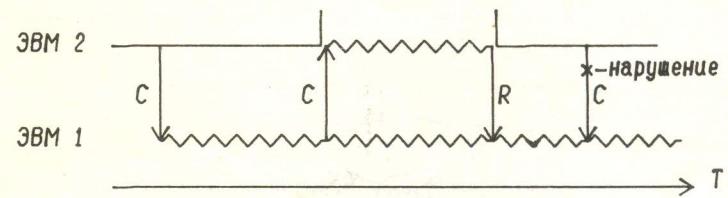


Рис.8. Неверная диаграмма

Почему мы различаем оба режима – центральной и терминалльной обработки? Поясним на примере работы с некоторой базой данных (БД) на ЭВМ1. В случае центральной обработки необходимо только один раз в начале работы 2М-модуля осуществить настройку на БД, а в дальнейшем выполнять необходимые работы без дополнительных указаний. В режиме терминалльной обработки компоненты 2М-модуля на ЭВМ1 должны независимо выполнять все операции, связанные с входом-выходом в/из БД. Другими словами, в этом режиме на ЭВМ1 должны выполняться только полностью замкнутые по отношению к внешней среде компоненты.

Во время работы 2М-модулей и автономного процесса обеспечивается возможность параллельной работы во времени ЭВМ двух-

машинной конфигурации. Кроме введенных режимов, мы рассматриваем еще режим пакетной обработки, когда пользователь на обеих ЭВМ может решать независимо программы, опирающиеся только на возможности библиотек ИМ-модулей.

Полный цикл деятельности САПР

На рис. 9 приведена диаграмма, которая показывает возможные переходы от одного режима обработки САПР к другому в процессе проектирования объекта или изделия. По сути дела, здесь представлен полный граф с петлями, вершинами которого являются режимы обработки:

- ЦО - центральная,
- АО - автономная,
- ТО - терминальная,
- ПО - пакетная,
- СО - совмещенная.

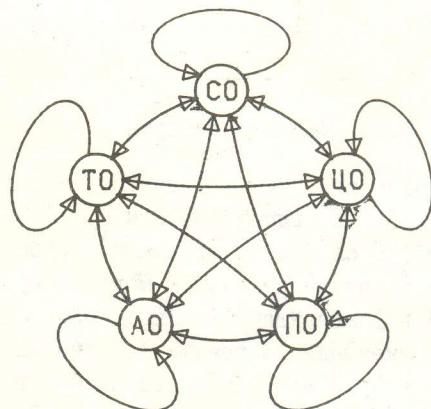


Рис. 9. Диаграмма возможных переходов от одного режима обработки в другой в процессе проектирования

Под пакетной обработкой понимается решение программы на одной ЭВМ без связи с другой и с абонентом. Переход из одного режима в другой не подразумевает какого-либо непрерывного процесса с использованием ЭВМ. На самом деле, сеансы обработки могут отстоять друг от друга сколь угодно далеко по времени. Это связано с тем, что данная схема игнорирует наличие такой работы (или режима работы), как работа человека без ЭВМ. Выделение СО как отдельного режима указывает на возможность программируемого перехода ТО - ЦО и обратно, даже без участия человека.

Предполагается, что человек при решении поставленной перед ним задачи для уменьшения полного времени ее решения использует только выделенные выше режимы обработки.

Библиотеки компонент 2M-модулей

Вернемся к рассмотрению режимов центральной и терминальной обработки. Здесь перед ПО запуска стоит задача вызова в решение некоторой поименованной компоненты, идентификация которой получена от ПО связи. Поскольку таких компонент на каждой ЭВМ может быть несколько, появляется понятие библиотек компонент 2M-модулей на каждой ЭВМ. На самом деле, может быть одна библиотека, которая содержит и IM-модули, и компоненты 2M-модулей, но функционально мы будем их различать. Компонента 2M-модуля на каждой ЭВМ должна строиться на основе использования только ПО связи и IM-модулей. Если IM-модуль имеет аналог в системе программирования, например, подпрограмму в языке Фортран, то компонента 2M-модуля может быть комплексом подпрограмм. От этого комплекса подпрограмм требуется, чтобы во временной развертке его работы соблюдалась правильная диаграмма, соответствующая выбранному режиму работы. Для вызова компонент в решение применяется средство типа LOADGO для мониторной системы "Дубна" на ЭВМ БЭСМ-6 [8], обеспечивающее динамическую подкачку модулей из библиотек.

На рис. 2 - 6 разными буквами помечены дуги, обеспечивающие связь между ЭВМ:

С (от CALL) - означает запуск поименованной компоненты 2M-модуля на другой ЭВМ;

Р (от RETURN) - означает сообщение в другую ЭВМ об окончании работы компоненты 2M-модуля.

В системе СПО ГД совместно с ПО связи реализовано ПО запуска компонент 2M-модулей, которое и реализует этот простой логический протокол. Передача данных между ЭВМ осуществляется в виде параметров указанных двух инструкций этого протокола. Использование только средств ПО связи дает возможность организации двухмашинных программ согласно диаграмме, приведенной на рис. 10. Здесь стрелками показаны только направления передачи данных. Вопросы определения момента и направления передачи ложатся полностью на алгоритм 2M-программы.

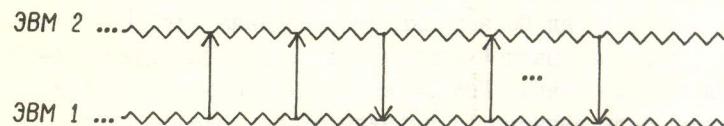


Рис.10.

Как уже отмечалось, ПО связи дает средства формирования и передачи сформированных данных на другую ЭВМ двухмашинной конфигурации, опроса наличия данных, переданных с другой ЭВМ, ожидание их передачи, прием.

При создании ПО связи необходимо учитывать операционную среду, в которой это ПО должно работать. На уровне ПО связи мы приняли модель канала, в которой ПО обеспечивает передачу по одному примитивному элементу данных: целое, вещественное число, текст, - и выделили операции начала и конца передачи (либо примитивы начала и конца передаваемых данных). Другими словами, на уровне входного языка ПО (модели канала) мы абстрагировались от технических деталей, таких, как квантование сообщения на порции, передача квантов, поддержка правильности

передачи по каналу. Эти вопросы решались и подробно изложены в [4]. Переходим к описанию возможностей ПО.

ФОРМИРОВАНИЕ и ПЕРЕДАЧА. Эти средства объединены на уровне входного языка. Данные различных типов в различных типах ЭВМ обычно представляются по-разному. Кроме того, на уровне систем программирования добавляются дополнительные типы данных, например, двойная точность на БЭСМ-6. Таким образом, ПО должно производить преобразование одного представления данных в другое.

Начало формирования и передачи данных осуществляется по оператору PUTP, по которому канал приводится в готовность, например, захватывается. Затем в канал передается через соответствующие входы ПО в нужной последовательности нужное количество данных нужных типов:

PUTI(I) – передает одно целое;

PUTB(B) – передает один байт без перекодировки;

PUTR(R) – передает одно вещественное значение;

PUTT(T,K) – передает текст из K байтов, возможно с перекодировкой, в представление символов, принятые в канале;

PUTD(D) – передается число с двойной точностью;

PUTQ – задается признак конца передаваемого сообщения.

ПРИЕМ ДАННЫХ. Прием информации из канала осуществляется также по одному данному нужного типа. При этом передающая и принимающая стороны должны быть согласованы:

GETP – инициализирует прием данных из канала в программу; в случае, если в канале еще нет этих данных, ПО становится на их ожидание автоматически; далее идет считывание;

GETI(I) – считывание одного целого;

GETB(B) – считывание одного байта;

GETR(R) – считывание одного вещественного;

GETT(T,K) – считывание текста из K байтов с перекодированием символов во внутренний код принимающей ЭВМ;

GETD(D) – считывание числа с двойной точностью;

GETQ – конец приема; одновременно означает отказ от информации, которая имеется в сообщении, но еще не считана.

Контроль типов передаваемых данных ПО связи не обеспечивается. Операторы PUT и GET – базовые, на их основе можно создавать подсистемы передачи данных с осуществлением контроля ти-

пов данных. Одна из таких подсистем описана в [2].

ОПРОС И СИНХРОНИЗАЦИЯ. При программировании компонент 2М-модулей, которые могут выполняться параллельно на двух ЭВМ (особенно для определения моментов связи между ними) необходимо иметь средства опроса канала и ожидания приема:

ANSW(K) – значение параметра K на выходе позволяет заключить, есть ли вообще связь с другой ЭВМ, существует ли сообщение от другой ЭВМ, занят ли канал передачей предыдущего сообщения в другую ЭВМ;

GETP – если сообщения еще нет, то программа переводится в состояние ожидания до его прихода;

PUTP – если канал еще занят передачей предыдущего сообщения, то программа переводится в состояние ожидания до освобождения канала.

Выше мы ввели понятие "канал занят передачей сообщения", такая ситуация возможна, поскольку в общем случае ПО связи может быть реализовано как параллельный процесс, что в данной работе не рассматривается (см. [4]). Отметим, что каких-либо специальных средств синхронизации работы 2М-модулей в рамках ПО связи не предусматривается. В целом описанные средства ПО позволяют организовать двухмашинную работу и подобны описанным в [9] для программирования диалоговой работы с терминалом в ОС ДИСПАК (экстракод 71).

За пределами нашего рассмотрения осталась задача первоначальной связи между двумя ЭВМ. Эта задача в каждом конкретном случае должна решаться особо с учетом возможностей операционных систем по обслуживанию технического сопряжения ЭВМ и решается в процессе реализации ПО связи.

Реализация СПО ГД

Выше были описаны возможности ядра СПО ГД и предложены режимы его использования. В ядро СПО ГД входят ПО связи, обеспечивающее межмашинный обмен информацией, и ПО запуска компонент 2М-модулей.

В системе СПО ГД за основной язык программирования 1М-модулей и компонент 2М-модулей выбран язык Фортран. ПО связи ре-

ализовано в виде пакета подпрограмм с процедурным входным языком, доступным из Фортрана. ПО запуска используется в задачах неявно и программируется через ПО связи посредством формирования инструкций логического протокола [3]. ПО связи может быть использовано автономно, в нем моделируется виртуальный канал со своим представлением данных. Поскольку одной из ЭВМ двухмашинной конфигурации СПО ГД является СМ-3, то в качестве внутреннего представления данных в канале выбрано представление, принятое в трансляторе с языка Фортран на этой мини-ЭВМ. Это позволило при передаче данных всю перекодировку производить только на большой ЭВМ (БЭСМ-6 или ЕС). Но ПО позволяет связывать и любые другие ЭВМ (например, БЭСМ-6 с ЕС), поскольку данные между ЭВМ находятся в унифицированном формате, а ПО связи на разных ЭВМ функционально идентично.

Функциональные возможности СПО ГД как графической диалоговой системы описаны [2]. При ее построении использовался совмещенный режим обработки (см. рис. 7). Основную трудность вызывает установление первоначальной связи, поскольку операционные системы БЭСМ-6 и ЕС не имеют простых штатных средств дистанционного запуска задач в решение в случае использования нестандартных каналов связи. В данном отношении оказалось удобно сопряжение БЭСМ-6 и СМ-3 через телеграфный канал. Наиболее удачным (из реализованных) решением в настоящее время является комбинированный канал, когда БЭСМ-6 и СМ-3 сопряжены как по телеграфному каналу БЭСМ-6, так и по 7-му направлению (см. [4]).

Отметим, что СПО ГД будет работать на любом двухмашинном комплексе в случае реализации на ЭВМ входящих в него ПО связи и ПО запуска.

Отметим следующие прагматические характеристики СПО ГД:

- построена на штатных средствах ОС, входящих в конфигурацию ЭВМ;
- основные обрабатывающие алгоритмы, запрограммированные в виде ИМ-модулей, обладают свойством переносимости с одной ЭВМ на другую за счет использования языка Фортран и единой системы управления динамическими, в том числе и внешними, массивами (СУП). Об этой компоненте ядра системы СПО ГД речь пойдет ниже.

При разработке сколько-нибудь серьезной САПР или системы машинной графики так или иначе приходится программировать работу с внешней памятью. В рамках ядра СЛО ГД была разработана система СУП и реализована в виде пакета прикладных программ для Фортрана с процедурным входным языком. Средства, предоставляемые системой СУП, используются для программирования динамической работы с внешней памятью при создании 1М-модулей и компонент 2М-модулей. Система построена на основе штатного программного обеспечения и может использоваться автономно и в составе СЛО ГД. Разрабатывалась она, в первую очередь, с учетом стоящих перед нами задач из области машинной графики и вычислительной геометрии. По своей сути это не СУБД, а, скорее, аналог файловой системы. Дело в том, что в настоящий момент нет общепризнанной СУБД, более или менее удовлетворяющей различным требованиям разработчиков программных средств машинной графики. Это видно и на примере проектов международных графических стандартов – систем CORE [I] и GKS [II], в которых имеются только понятия графического сегмента и метафайла, т.е. линейные структуры. В САПР, конечно, важен более высокий уровень – это пакеты вычислительной геометрии, которые должны применяться в системах моделирования. Различные представления геометрических объектов (многогранники, параметрическое задание, поверхности Кунса и т.д.) требуют, в первую очередь, разработки структур данных, которые зависят от представления геометрии в ЭВМ и которые, в свою очередь, определяются характером работы с объектами (описание или ввод, объединение, пересечение, разность, отображение с уборкой невидимых линий и т.п.). Область вычислительной геометрии в настоящее время бурно развивается, и нельзя с уверенностью сказать, какие структуры данных окажутся наиболее удачными. Поэтому уровень структур данных в СУП не рассматривается, а является прерогативой пакета прикладных программ. В то же время нас не удовлетворяет штатное обеспечение. Рассматривая среду, в которой решаются Фортран-программы, видим что

– средства работы с файлами очень бедны (БЭСМ-6);

- нет явных средств динамически образовывать и уничтожать файлы (ЕС ЭВМ);
- существуют различия в системах управления файлами (БЭСМ-6, ЕС, СМ ЭВМ). Различия оказывают влияние на организацию работы с внешней памятью и, таким образом, сказываются на самих алгоритмах.

Основная задача при создании СУП - обеспечение Фортран-программ средствами динамического образования внешних динамических массивов с прямым доступом к их элементам. Решение этой задачи повлекло за собой разработку всего комплекса вопросов, связанных с хранением информации во внешней памяти и доступом к ней.

Основным понятием СУП является виртуальная внешняя память (ВП) - это набор данных прямого доступа на ЕС или часть магнитного диска на БЭСМ-6. Для ее организации и работы с ней используются стандартные средства операционных систем.

ВП строится как коллективная, что важно при создании архивов в рамках интегрированных САПР. Единицей информации выбран байт. На уровне ВП выделено понятие области - динамический байтовый массив.

Идентификация областей производится по имени и шифру пользователя. С каждой областью ассоциированы следующие основные атрибуты: пароли доступа, даты последних записи и чтения, количество обращений, длина в байтах.

Система СУП предоставляет следующие средства при работе с ВП и областями:

- организацию ВП;
- образование и уничтожение области в ВП;
- работу с несколькими ВП в одной программе;
- работу с несколькими областями в одной ВП;
- работу с одной и той же ВП одновременно в различных программах;
- обмен информацией между программными переменными и областями. Обменная порция может содержать любое число байтов. При записи за пределы области она (область) автоматически расширяется;
- сервисные функции, такие, как смена паролей, значений атрибутов и т.д.

При работе с ВП в системе СУП предполагается следующая идеология. Группа пользователей, решая совместную задачу или просто желающая завести общий архив, организует ВП. Один из пользователей становится ее администратором. Ему система предоставляет более широкие полномочия, такие, как наложение ареста на область, на пользователя, получение общей информации о ВП и т.д. Отметим, что СУП не накладывает каких-либо правил на работу администратора. Он сам определяет, что должен делать и какие дополнительные функции программировать. Система только дает ему полный доступ к любым областям ВП и их атрибутам.

В режиме коллективного использования ВП только одна из задач, решаяющихся одновременно, может захватить область на запись, другие же в это время могут работать только в режиме чтения информации из захваченной области.

В системе СУП отсутствует понятие сборки мусора, каталоги ВП организованы таким образом, что после уничтожения области занимавшейся ею пространство сразу же может быть использовано. При разработке СУП была усовершенствована структура каталогов, которая использовалась для графического архива системы СМОГ [12].

В системе СУП введено понятие ВПОЗУ – виртуальная внешняя память, которая моделируется на оперативной памяти. Использование ВПОЗУ вместо ВП удобно в том случае, когда общий объем динамически возникающих массивов достаточно мал, меньше чем объем доступной оперативной памяти. СУП при работе с ВПОЗУ обеспечивает все средства, которые предоставляются программам с ВП. Операции СУП над областями используют не полную идентификацию области, а только ее системный ключ, который ей присваивается во время выполнения операции открытия или образования.

В наличии единой системы управления динамически создаваемых и уничтожаемых динамических внешних массивов мы усматриваем основное различие от многих подобных систем. Внешние понимаются как внешние по отношению к обрабатывающему модулю, хотя на самом деле могут размещаться в ОЗУ. В ряде существующих систем решаются подобные задачи. В системе программирования Barrouz-6700 имеются средства динамической работы с фай-

лами внешней памяти на языке управления заданиями Work Flow Language [13, 14]. Во многих пакетах прикладных программ так или иначе решается вопрос о динамическом управлении массивами в оперативной памяти, например, в пакете ДИГРАФ.Н[15] или в системе MODULEF [16]. В нашем случае мы обеспечили единую систему управления динамическими массивами как во внешней памяти, так и в ОЗУ ЭВМ. Отметим достоинства реализованной системы СУП:

- один язык управления динамическими массивами для разных ЭВМ (ЕС, БЭСМ-6). Это качество следует рассматривать как основу переносимости ПО с одной ЭВМ на другую без перепрограммирования;

- один язык управления динамическими массивами во внешней памяти и в ОЗУ ЭВМ. Это удобно, так как модули обработки потоковой информации программируются без учета реального расположения массивов, настраиваются перед работой извне, поэтому при малых объемах исходных данных (часто в тестовых отладочных задачах) обеспечивается высокая скорость выполнения операций без использования внешней памяти.

В разработке программного обеспечения СУП принимал активное участие С.А.Упольников, которому авторы выражают свою глубокую признательность.

Л и т е р а т у р а

1. Дебелов В.А., Мацокин А.М. Структура программного обеспечения графических дисплеев. - Автометрия, 1978, №5, с.85-86.
2. Дебелов В.А., Мацокин А.М., Чубарев А.И. СПО ГД - система программного обеспечения графического диалога. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 64-71.
3. Дебелов В.А., Торшин В.И., Чубарев А.И. Использование АРМ-Р в комплексе с БЭСМ-6. - В кн.: Интерактивные системы: Тез. докл. и сообщ. III школы-семинара, Боржоми, 15-21 марта 1981. Тбилиси, 1981, кн. 2, с. 206-207.
4. Чубарев А.И. Средства программирования сателлита в рамках СПО ГД. - Настоящий сборник, с. 23-44.

5. Гинзбург А.И., Родионов Ю.И. Программное обеспечение графической системы "Дельта" - ЭВМ "Минск-32". - Автометрия, 1974, № 4, с. 73-83.
6. Бобков В.А., Голенков Е.А., Перчук В.Л. Графический пакет для ЭВМ БЭСМ-6 и мини-ЭВМ "Электроника-100И" с графическим дисплеем УГД-43. - Автометрия, 1978, № 5, с. 89-91.
7. Карлов А.А., Кирилов А.С., Смолякова Т.Ф. Программные средства для организации диалога на языке высокого класса: Препринт № II-12160. - М., 1979. - 14 с. - В надзаг.: ОИЯИ.
8. Мазный Г.Л. Программирование на БЭСМ-6 в системе "Дубна". - М.: Наука, 1978.
9. Корякин В.К. и др. Работа пользователей в режиме непосредственного доступа в ОС ДИСПАК (руководство по работе с отладчиком). - Информатор ИПМ АН СССР, 1976, вып. I2, 2.
- I0. Status report of the Graphic Standards planning Committee. - Comput. Graph., 1979, v.13, no.3.
- II. Graphical Kernel System(GKS). Functional description. Version: 6.6. - Draft International Standard ISO:ISO TC97/SC5/WG2. 1981 - May 25.
- I2. Дебелов В.А. Буферная часть СМОГ. - В сб.: Машинная графика и ее применение. Новосибирск, 1974, с. 19-37.
- I3. I/O SUBSYSTEM. В 7000/B6000 Ser.: Ref. manual. - Detroit: Burroughs Corp., 1977.
- I4. WORK FLOW LANGUAGE. В 7000/B 6000 Ser.: Ref. manual. - Detroit: Burroughs corp., 1977.
- I5. Бучнев А.А., Кудряков В.Ф. Управление данными в машинной графике и системах автоматизации проектирования. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 19-22.
- I6. Hecht F. French contribution to Interlib. - INDIA, June 1982. - Pt.2. Dynamic allocation of arrays, p. 35-48.

А.И.Чубарев

СРЕДСТВА ПРОГРАММИРОВАНИЯ САТЕЛЛИТА В РАМКАХ СПО ГД

Введение

Настоящая статья посвящена описанию ПО АРМ – консольной части системы программного обеспечения графического диалога СПО ГД [1, 2, 3]. СПО ГД ориентирована на машинно-приборную конфигурацию, показанную на рис. I.

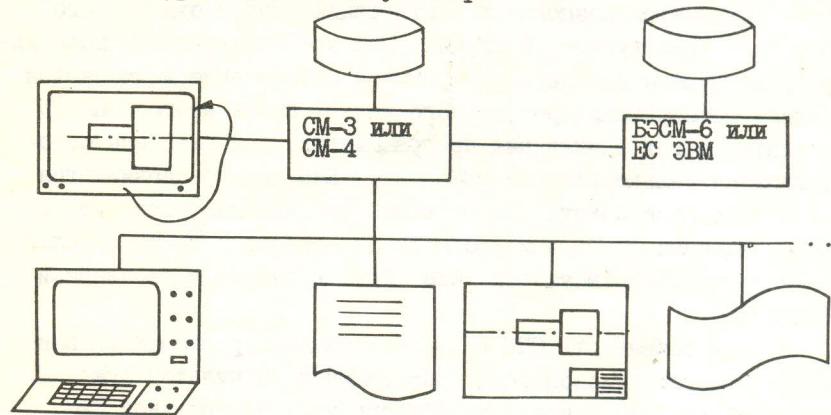


Рис. I

ПО АРМ реализована в настоящее время для АРМ-Р или АРМ-М, работающих под управлением ДОС М400. Поскольку СПО ГД относится к программируемым сателлитным системам, большое внимание при ее создании уделялось вопросам предоставления средств удобного программирования распределенного ПО и возможности эффективного переноса отдельных программных компонент с одной ЭВМ комплекса на другую.

В отличие от одноязыковых программируемых сателлитных систем [4], СПО ГД относится к многоязыковым, т.е. таким системам, у которых главная ЭВМ и машина-сателлит имеют различные системы программирования (хотя язык программирования один: Фортран). Такая система несколько сложнее в изучении: от разработчика ПО требуется знание двух ОС и протокола общения между двумя ЭВМ, но она имеет два существенных преимущества:

1. Обеспечивается более качественное программирование для каждой ЭВМ (в случае одноязыковых систем программирование консоли часто ограничивается).

2. Появляется возможность интеграции вновь создаваемого ПО с уже существующим (в случае АРМа это немаловажно, поскольку существующее для него программное обеспечение исчисляется многими мегабайтами программного кода). Кроме этого, СПО ГД заботится о тех прикладных программистах, чьи интересы ограничиваются только главной ЭВМ [2]. Для них обеспечиваются соответствующие пакеты подпрограмм, эмулирующие сателлит с фиксированными функциями. Возможности работы с таким сателлитом напоминают возможности GKS [5] в работе с дисплейной станцией.

Другая особенность СПО ГД состоит в двухуровневой организации диалога: пользователь, находящийся за пультом АРМа и взаимодействующий с прикладной программой центральной ЭВМ, может на фоне ее работы запросить выполнение автономной функции сателлита. Такие функции поддерживают пользователя при подготовке вводимых данных, их редактировании и простейших видах обработки. Прикладной программист, желающий пополнить такие функции, получает такие же возможности программирования, как и в случае создания распределенного ПО: основной

язык программирования (Фортран) плюс специальные пакеты подпрограмм по управлению возможностями СПО ГД. Общие принципы построения СПО ГД как двухмашинной сателлитной системы и возможности построения распределенного программного обеспечения в рамках этой системы изложены в [3].

В настоящей работе обсуждается архитектура ПО АРМ, назначение ее отдельных компонент и некоторые операционные возможности системы.

2. Архитектура ПО АРМ

Архитектуру ПО АРМ удобно представить в виде следующей, показанной в таблице, иерархии уровней:

Основной уровень	Подуровни (компоненты)
Базовый	ДОС М400 Исполняющая система Фортрана
Ядро ПО АРМ	Синхронизация Управление памятью (СУП)* Статистика* Загрузчик перекрытий Управление связью с большой ЭВМ Управление вводом-выводом Управление графическими устройствами Оболочка*
Прикладной	Библиотека прикладных программ (ПП) поддержки центральной ЭВМ Библиотека ПП поддержки автономной работы абонента

Примечание: Звездочкой помечены подуровни ядра, которые для ПО АРМ в рамках ДОС М400 не реализованы.

Кратко остановимся на основных уровнях ПО АРМ.

Базовый уровень включает компоненты штатного программного обеспечения, участвующего в работе ПО АРМ.

Этот уровень обеспечивает основную операционную среду, в которой функционируют все более высокие уровни. Основное правило, которому мы следовали, - использование только стандартных возможностей базовых средств и осуществление этого стандартным образом. Единственное отступление от этих правил состоит в использовании возможности работы с векторами прерываний и регистрами устройств "через голову" ДОС М400.

Ядро ПО АРМ - уровень, который несет основную нагрузку в достижении необходимой системной организации ПО АРМ. Все вопросы, решаемые ядром, условно можно разбить на три группы:

- 1) обеспечение недостающих элементов операционной среды;
- 2) расширение возможностей прикладного уровня;
- 3) поддержка мобильности компонент прикладного уровня.

Остановимся на каждой из этих групп.

1) То, что ДОС М-400 не устраивала нас во многих отношениях, выразилось в том, что ядро ПО АРМ приобрело сильную операционную направленность. Им решаются задачи

- статического выделения процессов и обеспечения механизма синхронизации;
- управления каналом связи с центральной ЭВМ;
- управления нестандартными периферийными устройствами АРМа (типа графического дисплея ЭПГ СМ);
- управления периферийными устройствами АРМа для достижения лучших диалоговых свойств системы;
- обеспечения эффективного механизма подкачки перекрытий.

2) Поскольку программирование функций ПО АРМ выполняется на прикладном уровне, а основной язык программирования этого уровня - Фортран, то на ядро ложится задача предоставления соответствующих пакетов подпрограмм по управлению всеми своими возможностями. Каждая компонента ядра обеспечивает такой пакет.

3) Мобильность, очевидно, ухудшается, если прикладной уровень использует многие нестандартные возможности. С другой стороны, без привлечения таких средств пока не удается достичь требуемых системных свойств и эффективности. Выделение ядра и унификация его функций - один из ключей, ведущих к

компромиссу. В рамках СПО ГД, в частности, ядро рассматривается как одно из средств, обеспечивающих мобильность [3]: все машинно- или операционно-зависимые элементы выносятся в ядро и унифицируются.

Наконец, для распределенных систем, помимо "просто переносимости", важно обеспечить сбор соответствующей статистической информации, поскольку вопрос переноса той или иной компоненты на другую ЭВМ, как правило, связан с решением задачи более рационального распределения программ и данных между сателлитом и главной ЭВМ. Решить такую задачу часто трудно, если система не предоставляет средств сбора статистики. Предоставление таких средств также ложится на ядро.

Прикладной уровень - содержательная часть ПО АРМ. Основными выполняемыми единицами этого уровня являются прикладные программы (ПП). Реализационно каждая ПП - это набор перекрытий, связанных по управлению. Любое перекрытие, в свою очередь, собирается из необходимых подпрограмм, написанных на Фортране или на языке макроассемблера.

В рамках ПО АРМ определены ПП двух типов:

а) Р-ПП (компоненты 2М-модуля, см. [3]) - прикладные программы, вызываемые главной ЭВМ. Каждая такая ПП реализует отдельную функцию для центральной ЭВМ. С точки зрения центральной ЭВМ каждая такая Р-ПП мыслится как некоторая подпрограмма от параметров. Взаимодействие с Р-ПП строится на основе логического протокола [6].

б) F-ПП(1М-модули, см. [3]) - прикладная программа автономной обработки, запускаемая абонентом. Это обычная диалоговая программа, написанная на Фортране с использованием возможностей ядра.

3. Процессы ПО АРМ

Как уже отмечалось выше, СПО ГД обеспечивает двухуровневую организацию диалога. Это предполагает наличие по крайней мере двух процессов [3] : центральной и автономной (или локальной) обработки.

На рис. 2 приведена принципиальная логическая модель центральной обработки в виде сети Петри [7], связывающая воедино работу абонента, АРМа и центральной ЭВМ.

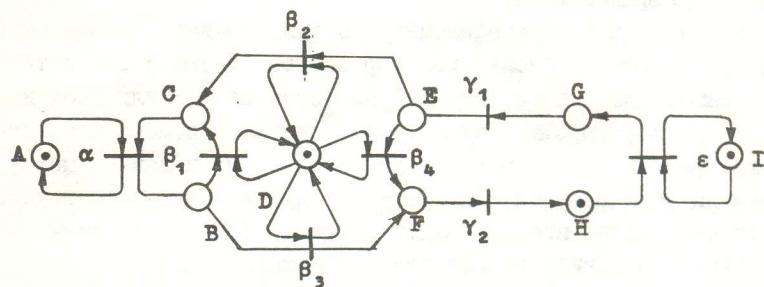


Рис. 2

Места A, D и I символизируют состояния соответственно абонента, АРМа и центральной ЭВМ. Места C и B символизируют порты связи между прикладным уровнем АРМа и драйверным уровнем, обслуживающим устройство ввода/вывода. Места E, F – это порты связи с драйверными процессами обслуживания канала связи с центральной ЭВМ. Места G, H – аналогичные порты для центральной ЭВМ. Эта модель, помимо того, что она показывает, как в принципе работает центральный процесс, вскрывает одно простое свойство процессов прикладного уровня (переходы β_1 , β_2 , β_3 , β_4 для АРМа, переход ϵ для центральной ЭВМ) – их последовательную обработку. Благодаря этому она легко укладывается в структуру управления Фортрана, а следовательно, программирование переходов может не выходить за возможности этого языка. Кроме того, логический протокол взаимодействия между прикладными программами АРМа и центральной ЭВМ также строится по аналогии с фортрановскими **CALL**, **RETURN**. Все это делается для того, чтобы распределенное программное обеспечение могло строиться человеком, знакомым лишь с возможностями Фортрана или близкими ему языками. В заключение заметим, что переходы α , γ_1 , γ_2 обеспечиваются ядром СПО ГД, а переходы β_1 , β_2 , β_3 , β_4 и ϵ программируются на прикладном уровне и, в частности, могут быть выполнены любым прикладным программистом.

Автономный процесс может быть представлен аналогичной моделью (рис. 3).

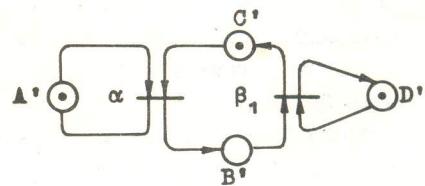


Рис. 3

Смысл мест и переходов тот же, что и для центрального процесса.

Взаимодействие между центральным и автономным процессами осуществляется за счет того, что состояния A и A' или D и D' не являются полностью независимыми.

В частности, в последнем случае состояния могут "пересекаться" по общей памяти в ОЗУ или на диске и т.п.

Организация взаимодействия между процессами, а также возможность использования ими неделимых ресурсов требуют решения задачи синхронизации. (Например, в рамках ПО АРМ для ДОС М400 неделимыми ресурсами являются:

- ДОС М400,
- исполняющая система Фортрана,
- отдельные устройства,
- поле под перекрытие,

и т.д.).

Задача синхронизации решается в рамках ПО АРМ с помощью построения дейкстровских синхронизирующих примитивов P , V по работе с двоичным семафором [8]. Обращение в рамках Фортрана выглядит следующим образом:

CALL BEGIN - операция P ,
CALL END - операция V .

В заключение отметим два момента.

А. Приведенные две модели соответствуют ситуации, когда оба процесса выполняются параллельно. Если центральный процесс отсутствует, то логическая модель автономного процесса становится тождественной модели центрального процесса. Огра-

ничение, накладываемое на автономный процесс в момент выполнения центрального, – следствие наличия только одного логического канала связи в рамках СПО ГД, реализованного для ДОС М400. Если центральный процесс присутствует – он владеет каналом в монопольном режиме. В противном случае – каналом владеет автономный процесс.

Б. Большие ограничения по памяти не позволили обеспечить параллельную работу в рамках ДОС М400 Р- и F-III. В результате в каждый момент времени выполняется (на АРМе) или центральный или автономный процессы. Однако, поскольку выполнение любой Р-III скоротечно, а запрос на запуск F-III становится в очередь и выполняется сразу же по освобождению поля перекрытия, абонент практически не ощущает больших неудобств, связанных с задержкой вызова автономных функций.

4. Оверлейный механизм

Как уже отмечалось выше, весь прикладной уровень ПО АРМ строится в виде системы перекрытий. В связи с этим важно иметь удобные операции по работе с перекрытиями. Стандартная операция **LINK**, во-первых, не обладает нужной эффективностью, поскольку подкачка каждый раз осуществляется из спискового файла и, во-вторых, не обеспечивает возврата в вызывающее перекрытие.

В рамках ПО АРМ был создан загрузчик перекрытий **CALL**, лишенный отмеченных недостатков. Обращение к нему имеет следующий вид:

CALL CALL ('имя перекрытия')

На рис. 4. приведена структура данных загрузчика **CALL**, а его алгоритм работает следующим образом. Сначала в стеке запоминается указатель на текущее перекрытие (см. рис. 4). Затем делается попытка загрузить новое перекрытие из библиотеки образов (непрерывный файл XXXXX.XXX на рис. 4) по имени, заданному в обращении. Для этого имя ищется в каталоге образов. Если оно найдено, текущий указатель устанавливается на данный элемент каталога образов, а соответствующий образ под-

качивается за один обмен с помощью макрокоманды ДОС M400 .TRAN, и управление передается на начало вызванного перекрытия. Вся необходимая информация для проведения загрузки и запуска перекрытия содержится в каталоге образов.

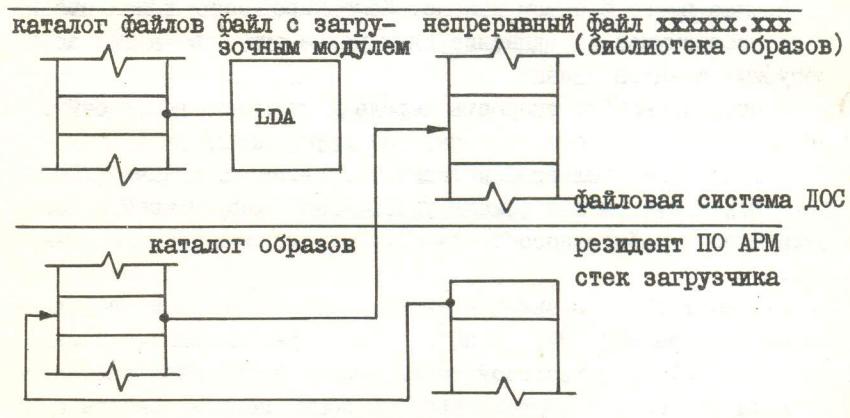


Рис. 4

Если имя не найдено, то перекрытие подкачивается стандартными средствами ДОС M400 (макрокоманда .RUN) из спискового файла .LDA (см. рис. 4), после чего оно дублируется в библиотеке образов (макрокоманда .TRAN), а каталог образов пополняется, и текущий указатель устанавливается на соответствующий элемент каталога образов. По возврату управления из перекрытия загрузчик восстанавливает прежнее перекрытие, для чего использует указатель, расположенный в вершине стека, и возвращает управление в место вызова.

Конфликтов из-за нехватки памяти в каталоге образов или библиотеке образов не возникает, поскольку их размеры согласованы с существующим числом перекрытий.

5. Управление связью с большой ЭВМ

Построение ПО связи в рамках СПО ГД является одним из на-

иболее отработанных элементов, что связано с частыми переходами на новую аппаратуру сопряжения между ЭВМ. В этом разделе мы опишем комбинированный канал связи для комплекса АРМ - ЕСМ-6, который наиболее полно вобрал в себя все возможности принятых решений.

Прежде всего о самом канале. Комбинированный канал связи есть программное объединение двух ранее существовавших аппаратурных каналов связи:

- быстрый канал; скорость передачи не менее 100 к байт, используется для передачи основной информации;
- медленный (телеграфный)канал; скорость передачи 1200 бод, используется для обмена управляющей информацией и восстановления работоспособности быстрого канала при его отказах.

Для построения взаимодействия между программами, выполняющимися на разных ЭВМ, в СПО ГД используется техника портов [9] (рис. 2). С логической точки зрения порт можно рассматривать как специально организованную общую память, работа с которой строится на основе специальных операций (примитивов) типа: ПОСЛАТЬ, ПОЛУЧИТЬ. Для работы с портами на обеих ЭВМ в рамках СПО ГД прикладной программе представляются наборы операций [5], эквивалентные следующим:

ПОСЛАТЬ - БАЙТ (БАЙТ)
ПОСЛАТЬ - ЦЕЛОЕ (ЦЕЛОЕ)
ПОСЛАТЬ - ВЕЩЕСТВЕННОЕ (ВЕЩЕСТВЕННОЕ)

• • •

ПОЛУЧИТЬ - БАЙТ (БАЙТ)
ПОЛУЧИТЬ - ЦЕЛОЕ (ЦЕЛОЕ)
ПОЛУЧИТЬ - ВЕЩЕСТВЕННОЕ (ВЕЩЕСТВЕННОЕ)

• • •

Эти примитивы учитывают основные типы данных Фортрана - языка, для которого разработана СПО ГД. Задача ПО связи СПО ГД и состоит, преимущественно, в реализации указанной системы портов. В коммуникационном ПО СПО ГД выделяются следующие уровни на ЕСМ-6:

- программный,
- логический транспортный,

- физический транспортный,
- отражатель временного синхронизатора,
- фильтр,
- ОС ДИСПАК

и на СМ-3:

- фильтр,
- формирователь временного синхронизатора или сообщения,
- физический транспортный,
- логический транспортный,
- программный.

Кратко поясним функциональную нагрузку каждого уровня.

Программный уровень - это программа, написанная на Фортране, выполняемая в рамках СПО ГД и взаимодействующая с аналогичной программой на другой ЭВМ. Основные положения такого взаимодействия следующие:

1. Программа взаимодействует "непосредственно" с аналогичной программой на другой ЭВМ и ничего не знает о более низких уровнях (прозрачность).

2. Программа взаимодействует в терминах Фортрана (т.е. манипулирует данными, имеющими стандартный тип: байт, целое, вещественное и т.п.).

3. Взаимодействие строится с учетом логического протокола [6].

Третье положение является основным, отличающим данный уровень от логического транспортного. Соблюдение логического протокола означает, что данные посылаются в канал не произвольно, а согласно некоторым синтаксическим и семантическим договоренностям, благодаря которым удается обеспечить нужную операционную среду. В частности, операционная среда позволяет

- осуществить вызов ПП из библиотеки на другой ЭВМ;
- передать ПП на другой ЭВМ необходимые параметры;
- получить результаты счета ПП на другой ЭВМ;
- согласовать работу ПП на разных ЭВМ.

В свою очередь, программный уровень также может делиться на некоторое число подуровней. В этом случае к коммуникационному ПО следует относить лишь те, которые обеспечивают соблюдение

ние логического протокола (2М-модули в [3]). Резюмируя скажанное, определим программный уровень коммуникационного ПО:

Программный уровень коммуникационного ПО – это та часть программного обеспечения, которую необходимо "поправить" при смене логического протокола взаимодействия между программами разных ЭВМ, принятого в СПО ГД.

Логический транспортный уровень обеспечивает связь между ЭВМ в терминах портов. Основным отличием этого уровня от программного является отсутствие требования работать в рамках логического протокола. Это означает, что можно изменить логический протокол, не меняя спецификаций этого уровня.

Для программного уровня логический транспортный уровень предоставляет набор операций типа ПОСЛАТЬ, ПОЛУЧИТЬ, основными среди которых являются:

НАЧАТЬ-ПЕРЕДАЧУ-СООБЩЕНИЯ
ПОСЛАТЬ-БАЙТ (БАЙТ)
КОНЧИТЬ-ПЕРЕДАЧУ-СООБЩЕНИЯ
НАЧАТЬ-ПРИЕМ-СООБЩЕНИЯ
ПОЛУЧИТЬ-БАЙТ (БАЙТ)
КОНЧИТЬ-ПРИЕМ-СООБЩЕНИЯ.

Передача более сложных типов данных строится на основе байтовой передачи (здесь же, если требуется, делается перекодировка). Передаваемые или принимаемые сообщения могут быть любой длины. Данный уровень является прозрачным в отношении байтовой передачи: посланный байт на одной ЭВМ тождествен принятому байту на другой ЭВМ.

В то же время логический транспортный уровень осуществляется частичную адаптацию процесса приема/передачи к реальной операционной среде. "Частичность" адаптации заключается в том, что этот уровень работает в условиях некоторой виртуализованной операционной обстановки, основными положениями которой являются:

- отсутствие совмещения приема/передачи;
- обмен осуществляется порциями по k байт, где k – заранее определенное число.

Первое положение принято, чтобы упростить требования к опера-

ционным возможностям обеих ЭВМ. Второе положение собственно и обеспечивает приближение к реальному механизмы обмена.

Таким образом, от более низких уровней требуется выполнение лишь двух основных операций:

ПОСЛАТЬ - ПОРЦИЮ (A) (*)

ПРИНЯТЬ - ПОРЦИЮ (A)

где A - байтовый массив информации.

Выделение логического транспортного уровня позволяет значительно уменьшить объем ПО, сменяемого при замене аппаратуры сопряжения.

Физический транспортный уровень. В его задачу входит адаптация логического уровня к конкретной операционной среде. Для этого необходимо (и достаточно) обеспечить две основные операции (*). Взаимодействие между физическими уровнями на разных ЭВМ строится на основе транспортного протокола, который предусматривает небольшой диалог по обмену каждой порцией. Физический уровень - это та часть канального ПО, которая меняется при смене аппаратуры сопряжения между ЭВМ.

Отражатель временного синхронизатора. Если, оформив запрос на прием информации от АРМа, операционная система БЭСМ-6 не получает ответ в течение некоторого отрезка времени (для телеграфного канала - 5 мин для быстрого канала - 10 с), она прекращает обмен и сообщает об этом в программу математика. Чтобы не возникало таких разрывов связи, программное обеспечение на АРМе формирует специальное сообщение (порцию), которое мы называем времененным синхронизатором. В задачу описываемого уровня на БЭСМ-6 входит "отражение" этого синхронизатора обратно в АРМ, после чего начинается новый цикл ожидания ответа.

Фильр (отбраковка/повторение) - это часть физического транспортного уровня, выполняющая следующие функции:

1) проверку правильности принятой порции от другой ЭВМ, и, если она неправильна, посылку бракующего сообщения;

2) если сообщение правильно и является бракующим, повторяется последняя передававшаяся порция, отличная от бракующего сообщения.

3) проведение восстановления, если обнаружена остановка канала.

ОС ДИСПАК обеспечивает ряд экстракодов для работы с обыми каналами связи. В случае медленного канала - это стандартный экстракод 7I. Для быстрого канала необходимые экстракоды были разработаны в ВЦ СО АН СССР [10].

Формирователь временного синхронизатора или сообщения - один из драйверных процессов ПО АРМ. В активном состоянии процесс следит за расходом **кванта** времени ожидания БЭСМ-6 и появлением готовой порции от коммуникационного ПО АРМа. В зависимости от того, какое событие происходит раньше, он формирует синхронизатор или сообщение-порцию. Естественно, что работа этого процесса согласована с работой остального ПО связи АРМа. Полный перечень операций по работе с каналом связи, обеспечиваемый в рамках ПО АРМ прикладному программисту, приведен в [3].

6. Управление вводом-выводом

В этом разделе опишем подсистему ПО АРМ, решающую задачи управления простейшими устройствами ввода/вывода. В первую очередь, мы относим сюда все символьные устройства. Это ввод с буквенно-цифровых клавиатур или перфолент и вывод на экран видеотона. Подход, выполненный в рамках ПО АРМ, напоминает подход **UNIX** [11], где в каждый момент у программы определен один стандартный поток ввода и один стандартный поток вывода. В нашем случае также определены стандартные потоки, но имеются следующие отличия:

1) вывод жестко закреплен за двумя устройствами и осуществляется синхронно: на видеотон и эмулятор видеотона на экране ЭПГ СМ (при переносе на АРМ-М последний был убран); эмулятор видеотона может отключаться абонентом;

2) для операторов ввода определен тип вводимой информации, например,

CALL INI(1) - ввод целого,

CALL INR(R) - ввод вещественного,
CALL INS(STRING, N, K) - ввод строки

и т.д.

У абонента имеется возможность закрепить в любое время ввода за любым типом одно из возможных устройств ввода [2];

3) в качестве возможных устройств в схему включены графические устройства ввода: сколка, маркер, крест (последние два определены только для АРМ-Р и моделируются программно); для них ввод интерпретируется как пара вещественных чисел.

Некоторая "недоразвитость" возможностей вывода определяется тем, что, наряду с предоставлением этих возможностей, были сохранены все стандартные возможности Фортрана в работе с операторами READ, WRITE.

Кроме этого следует отметить, что дальнейшее развитие системы ввода-вывода, по-видимому, следует осуществлять в комплексе с созданием системы управления памятью (СУП), статистики и, самое главное, оболочки (см. ниже), позволяющей в ряде случаев существенно повысить эффективность работы абонента. Качественное решение всех этих вопросов зависит как от имеющихся ресурсов АРМа, так и от выбора операционной системы.

7. Управление графическими устройствами

В этом разделе мы остановимся на задаче подключения графического векторного дисплея в рамках ПО АРМ. Как известно, к графическому ПО предъявляются два противоречивых требования:

- оно должно быть унифицировано (т.е. универсально, в каком-то смысле);

- должно позволять максимально использовать сильные стороны дисплея (т.е. учитывать особенности каждого из них).

Однако, учитывая, что существующие дисплеи сильно непохожи один на другой (например, ЭПГ СМ и УПИ), по-видимому, эти два требования качественно удовлетворить нельзя без программного моделирования многих функций. Из-за острого дефицита памяти мы решили ограничиться лишь унификацией уровня данных. В рамках СПО ГД такой уровень представлен понятиями вир-

туального графического сегмента и архива графических сегментов, единими для всех ЭВМ комплекса. Задача отображения и редактирования графических сегментов решается для каждого конкретного дисплея наиболее адекватными средствами. Здесь мы опишем средства программирования, предоставляемые прикладному программисту для работы с ЭПГ СМ в рамках ПО АРМ-Р.

С точки зрения прикладного уровня дисплей отождествляется с понятием архива графических сегментов. Элементами этого архива являются графические сегменты. Каждый сегмент обладает рядом атрибутов: подсветка, чувствительность, привязка и т.п. Изменение атрибутов вызывает изменения изображения на уровне сегментов, установка атрибута "подсветка" вызовет подкачку сегмента в дисплейный файл и включение его в цикл регенерации. Обратная операция вызовет обратный эффект: сегмент будет откачен на диск. Изменение атрибута привязки вызовет сдвиг изображения на экране дисплея, если он выключен, и т.д. С другой стороны, графический сегмент представляет собой линейный массив графических элементов [2]. Редактирование этого массива вызовет соответствующие изменения изображения.

8. Оболочка

Оболочка для ДОС М-400 не была реализована, поскольку для ее построения необходимо иметь более совершенную системную организацию, чем та, которая достигнута в рамках ПО АРМ ДОС М400.

Под оболочкой мы понимаем механизм макроввода, занимающий промежуточное положение между драйверным уровнем и системой ввода-вывода (рис. 5).

Оболочка может выключаться из работы – тогда мы приходим к обычному варианту системы.

Когда оболочка включена, абонент может использовать макрокоманды, позволяющие существенно экономить время, затрачиваемое на ввод. Библиотека макроопределений в простейшем случае представляет собой именованные "последовательности действий" абонента, выполненных на устройствах АРМа. Всякий раз, когда ему необходимо повторить какую-либо именованную последователь-

ность, достаточно указать имя, все остальное сделает оболочка. Таким образом, оболочку можно сравнить с программируемым "автопилотом", подменяющим абонента всякий раз, когда надо выполнить стандартную рутинную работу по нажатию большого числа кнопок АРМа.

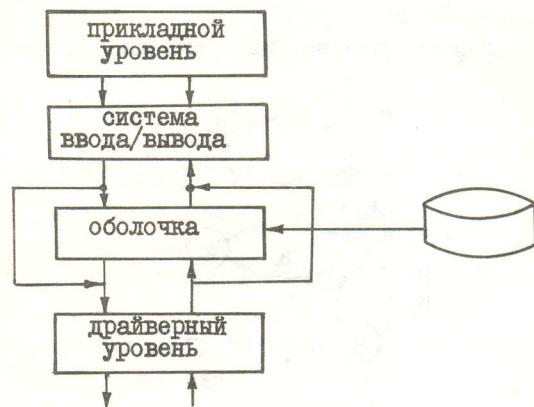


Рис. 5

Идея оболочки не нова. Ее истоки лежат в пакетных системах, использующих макрогенераторы. Примером интерактивной оболочки может служить оболочка **UNIX** [12]. Отличия нашего подхода, по-видимому, будут касаться

- языка,
- включения графических устройств,
- реализации.

В заключение можно привести простую модель, иллюстрирующую эффект от включения оболочки в систему.

Систему без оболочки можно представить в виде простейшей сети массового обслуживания, состоящей из двух обрабатывающих узлов с очередями (рис. 6).

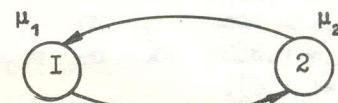


Рис. 6

Первый узел обозначает абонента, второй – систему. Полагаем, что система работает в режиме запрос–ответ, поэтому можно считать, что в ней циркулирует одно задание и сеть является замкнутой. Предположим, что все узлы имеют показательный закон времени обработки; μ_1, μ_2 – соответствующие интенсивности обработки.

Расширение системы оболочкой приводит к следующей модели:

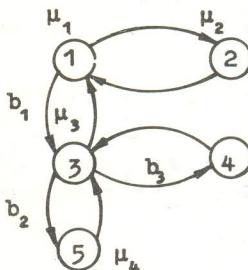


Рис. 7

1 – абонент; 2 – система; 3 – оболочка; 4 – система;
5 – диски.

Предположим, что параметры модели имеют следующие значения:

$\mu_1 = 0.2$ (обдумывание одного ответа $\sim 5\text{c}$),
 $\mu_2 = 2$ (выполнение одного запроса $\sim 0.5\text{c}$),
 $\mu_3 = 100$ (формирование ввода оболочкой $\sim 0.01\text{c}$),
 $\mu_5 = 10$ (обмен с диском $\sim 100\text{мс}$),
 $b_1 = 0.5$ (в половине случаев используем макрокоманду),

$b_2 = 1/12$ } считаем, что в среднем одна макрокоманда
 $b_3 = 10/12$ } содержит 10 действий и требуется один обмен с диском на ее выполнение.

Тогда с оболочкой доля времени работы абонента $\approx 0.64I$,
доля времени системы ≈ 0.3527 , оболочки ≈ 0.0063 (сюда включена доля диска ≈ 0.005).

Без оболочки доля времени работы абонента ≈ 0.909 ,
системы ≈ 0.091 .

Таким образом, мы видим, что для приведенной модели эффект введения оболочки с отмеченными свойствами состоит в следующем.

Загруженность ЭВМ возросла в ≈ 3.876 раз, а значит, можно предположить, что и работа, произведенная ею, будет во столько же раз больше.

Среднее время отклика увеличилось с 0.5с до $\sim 2.8\text{с}$, т.е. ухудшилось примерно в 5.6 раза.

Из этого можно сделать следующие выводы:

1. Оболочку лучше применять для тех задач, где время реакции мало, например, в редакторах (в текстовом редакторе время обработки одного символа $\ll 1\text{с}$).

2. Оболочку можно применять как средство пакетирования для сложных задач.

3. Число терминалов, обрабатываемых системой с оболочкой, существенно меньше, чем число терминалов, обрабатываемых той же системой без оболочки. По Клейнроку [13], например, для приведенного случая M^* без оболочки равно 11 терминалам, M^* с оболочкой – 2.79 терминала (здесь M^* – мера насыщения системы).

9. Прикладной уровень ПО АРМ

Как мы уже отмечали, прикладной уровень ПО АРМ строится из прикладных программ двух типов: Р-ПП, реализующих функции сателлита для центральной ЭВМ, и Ф-ПП – автономных диалоговых программ, обслуживающих абонента, находящегося за пультом АРМа.

Кратко резюмируем возможности, предоставляемые прикладному программисту при создании этих ПП в рамках ПО АРМ ДОС М400.

1. Язык программирования – Фортран. Сохранение практически всех его возможностей.

2. Предоставление пакета подпрограмм для связи с ПП на другой ЭВМ.

3. Предоставление пакета подпрограмм для управления простейшими устройствами ввода/вывода.
4. Предоставление пакета подпрограмм по управлению графическим дисплеем.
5. Обеспечение "структурной" работы с перекрытиями.
6. Предоставление средств синхронизации. (Несмотря на то, что работа F-III и P-III разделена, остаются конфликты с драйверными процессами, которые также используют неделимые ресурсы).

Кроме предоставления возможностей, конечно, существуют и ограничения. Прежде всего они касаются размеров перекрытия. В рамках ПО APM оно может достигать 4K в самых "стесненных" условиях.

Заключение

Предложенный подход имеет ряд достоинств:

1. Обеспечен программируемый сателлит. Как показывает опыт двухлетней эксплуатации, новые функции, даже сложные, подключаются быстро, и их работа достаточно эффективна.
2. Обеспечен двухуровневый режим работы: абонент может в любой момент заняться формированием ввода или редакцией графической информации на APMe. Центральная ЭВМ при этом может продолжать работу.
3. Система достаточно проста и компенсирует усилия, потраченные на ее изучение в стадии создания нового ПО.
4. Предложен гибкий ввод данных, превосходящий стандартные возможности Фортрана.
5. Обеспечены высокие адаптационные свойства. В частности, это выразилось в простоте смены каналов связи между ЭВМ и легкости переноса системы с APM-P на APM-M.
6. Сателлитная часть совершенно не зависит от особенностей центральной ЭВМ.

Временным недостатком системы является отсутствие ее реализации для других операционных систем, скажем, для ОС РВ.

В заключение автору приятно выразить благодарность А.М.Марокину за постоянный интерес, поддержку и обсуждения, В.А.Деникину

белову, у которого автор многому научился и в общении с которым родилось немало идей, нашедших воплощение в ПО АРМ, а также В.И.Торшину и Р.Н.Вильданову, принявших активное участие в реализации системы.

Л и т е р а т у р а

1. Дебелов В.А., Мацокин А.М. Структура программного обеспечения графических дисплеев. - Автометрия, 1978, № 5, с. 85-86.
2. Дебелов В.А., Мацокин А.М., Чубарев А.И. СПО ГД - система программного обеспечения графического диалога. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 64-71.
3. Дебелов В.А., Мацокин А.М., Чубарев А.И. Подход к разработке распределенных систем для двухмашинных комплексов Настоящий сб., с. 5-22.
4. Van Dam A., Stabler G.M. Intelligent satellites for interactive graphics. - In: Proc. Nat. Comput. Conf., 1973, p. 229-238.
5. Graphical Kernel System (GKS). Functional description. Version:6.6. - Draft International standard ISO:ISO TC 97/SC5/WG 2. 1981 - May - 25.
6. Дебелов В.А., Торшин В.И., Чубарев А.И. Использование АРМ-Р в комплексе с БЭСМ-6. - В кн.: Интерактивные системы: Тез. докл. сообщ. II школы-семинара. Боржоми, 15-21 марта 1981 г. Тбилиси, 1981, кн. 2, с. 206-207.
7. Котов В.Е. Формальные модели параллельных вычислений: Препринт № 165. Новосибирск, 1979. - 56 с.-В надзаг.: ВЦ СО АН СССР.
8. Дијкстра Э., Взаимодействие параллельных процессов. - В сб.: Языки программирования. М., 1972. с. 9-86.
9. Цикритзис. Д., Бернстайн Ф. Операционные системы. - М.: Мир, 1977.

- I0. Васильева Л.Ф. Системные программные средства диалога для комплекса ЭВМ БЭСМ-6 - терминальная станция на базе малой ЭВМ типа М400, СМ-3. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 28-31.
- II. Thomoson K. UNIX Implementation. - Bell Syst. Techn. J. 1978, v. 57, no. 6, p. I93I-I946.
- I2. Bourne S.R. The Unix Shell. - Bell Syst. Techn. J., 1978, v. 57, no.6, p. I97I-I989.
- I3. Клейнрок Л., Вычислительные системы с очередями. - М.: Мир, 1979.

В.И.Торшин

ПРОГРАММНОЕ УПРАВЛЕНИЕ ГРАФИЧЕСКИМИ УСТРОЙСТВАМИ АВТОМАТИЗИРОВАННЫХ РАБОЧИХ МЕСТ

Разработчики систем автоматизированного проектирования (САПР) в своей работе каждый раз сталкиваются с серьезнейшей методологической проблемой организации взаимодействия человека и ЭВМ.

Каждая автоматизированная система решает эту проблему по-своему, в зависимости от поставленной задачи и аппаратуры, которую может (или нужно) использовать. Фактически в настоящее время решение проблемы взаимодействия в системах автоматизированного проектирования сводится к организации управления устройствами аппаратурной среды САПР и, с их помощью, организации ввода и вывода каким-либо (обычно проблемно-ориентированным) образом структурированной информации. В качестве аппаратурной среды САПР самых разных направлений (например, машиностроения, радиоэлектроники и т.п.) в последнее время широко используются аппаратурно-программные комплексы АРМ (автоматизированные рабочие места), построенные на базе мини-ЭВМ. В комплектацию таких комплексов АРМ входят: мини-ЭВМ (типа СМ-4), алфавитно-цифровой дисплей, графический дисплей, кодировщик графической информации, гра-фопостроитель, а также устройство сопряжения с вычислительными машинами. Последнее устройство позволяет АРМ работать в комплексе с другой ЭВМ (например, из сети ЭВМ). Таким образом, все перечисленное оборудование можно взять за аппаратурную среду для построения САПР. В качестве первоочеред-

ной на этом пути стоит задача ввода, хранения, обработки и вывода информации для аппаратурного комплекса САПР. А ее подзадачей является работа со специальной графической информацией.

В ВЦ СО АН СССР в течение ряда последних лет разрабатывалось программное обеспечение для аппаратурного комплекса САПР, состоящего из двух частей: АРМ и Центральная ЭВМ. Это обеспечение получило название системы программного обеспечения графического диалога (СПО ГД) [1]. В СПО ГД можно выделить две составляющие части: БПО – часть программного обеспечения, расположенного на центральной ЭВМ (ЦЭВМ), и ПО АРМ – часть программного обеспечения, расположенного на мини-ЭВМ комплекса АРМ. ПО АРМ решает следующие основные задачи:

- организацию взаимодействия двух ЭВМ комплекса САПР;
- ввод, хранение, вывод графической информации.

Для решения этих задач ПО АРМ содержит в своем составе:

- графическую систему (ГС), которая организует вывод графической информации на графические устройства: графический дисплей (ГД), графопостроитель (ГП);
- систему ввода (СВ), которая обеспечивает работу с любым устройством ввода комплекса АРМ по единым правилам (алфавитно-цифровая клавиатура дисплея УТ-340, графического кодировщика, графического дисплея; устройство ввода с перфоленты);
- базу данных (БД), которая обеспечивает хранение графической, текстовой и числовой информации; функции БД в настоящей версии ПО АРМ выполняет файловая система ДОС 400;
- графический редактор (ГРЕД), который организует ввод и редактирование графической информации;
- систему взаимодействия с ЦЭВМ [1], которая обеспечивает связь ПО АРМ с прикладной программой на ЦЭВМ и асинхронный ввод на АРМ.

Кроме перечисленных систем, составляющих ПО АРМ, для обеспечения более тесного контакта пользователя АРМ (а значит, и построенных на этой базе САПР) с его оборудованием, другими словами, для организации более эффективного взаимодействия человека с ЭВМ, в САПР необходимо включить в состав ПО АРМ следующие системы:

1. Текстовой редактор, который обеспечивает подготовку и редактирование текстовой информации.
2. Арифметический калькулятор, который обеспечивает оперативные расчеты числовых величин по формулам.
3. Графический интерпретатор, который обеспечивает различные графические представления (изображения) числового массива.
4. Графический калькулятор, который обеспечивает оперативные расчеты некоторых характеристик графических объектов.

5. Документатор, который обеспечивает выпуск технической и прочей документации с помощью оборудования АРМ и/или ЦЭВМ.

Все перечисленные части ПО АРМ можно использовать как в комплексе – в СПО ГД, так и отдельно как независимые системы.

Остановимся более подробно на трех системах из ПО АРМ: системе ввода (СВ), графическом редакторе (ГРЕД), графической системе (ГС). В основу ПО АРМ был заложен ряд принципов, которым предполагалось следовать разработчикам каждой из систем ПО. Эти принципы в основном были выдержаны для всего ПО АРМ. Перечислим их:

- независимость устройств (т.е. устройства, в принципе, могут работать одновременно и друг другу мешать не должны);
- принцип максимального умолчания (т.е., если пользователю не требуется какие-то необычные возможности по вводу-выводу, то ему достаточно знакомства с ПО АРМ в минимальном объеме);
- взаимозаменяемость его устройств (т.е. возможна интерпретация средств одного устройства с помощью другого);
- использование программных устройств, расширяющих аппаратные возможности комплекса АРМ (т.е. включение в ПО АРМ программно-реализованных устройств, не имеющих аппаратного воплощения);
- контроль за допустимостью вводимой-выводимой информации (т.е. проверка на возможность представления информации в ограничениях конкретного устройства);
- недетерминированность форматов вводимой информации (т.е. возможность вводить информацию, вообще говоря, в произвольном формате – все необходимые преобразования выполняет программное обеспечение);
- единые (унифицированные) правила работы с устройствами ввода-вывода АРМ.

Практически все перечисленные принципы нашли свое отражение в системе ввода (СВ), которая фактически является фундаментом всего ПО АРМ. СВ работает в терминах типов данных, принятых в Фортране, а именно: целое, вещественное и символьная (байтовая) строка. Ввод одного из типов данных производится при обращении к соответствующей подпрограмме СВ, причем выбор конкретного устройства ввода осуществляется пользователем, в зависимости от его потребностей и привычек. Ввод информации с устройства можно прервать в произвольный момент и продолжить на любом выбранном устройстве АРМа. Так, например, с помощью функциональной клавиатуры (ФК) можно ввести код нажатой клавиши, этот же ввод можно произвести и с алфавитно-цифровой клавиатуры (АЦК), причем для набора кода ФК можно использовать АЦК графического и алфавитно-цифрового дисплея, а также графического кодировщика поочередно, например, по одной цифре на каждом из устройств.

Следующая система ПО АРМ, которой дадим краткую характеристику, – это графическая система (ГС). ГС решает задачу отображения графической информации (описания графической информации – ее модели) на графические устройства вывода – ГД, ГП. Вышеперечисленные принципы построения ПО АРМ нашли свое отражение и в ГС. В частности, единые правила работы с устройствами графического вывода позволяют в принципе не зависеть от конкретной аппаратурной конфигурации САПР, другими словами, можно, например, обойтись без графического дисплея, не уменьшая принципиальных возможностей системы, ГС работает с некоторой моделью графического изображения. Предполагается, что все изображение размещено на некоторой плоскости – так называемой поверхности изображения [2]. С помощью ГС поверхность изображения, точнее, ее содержимое, можно просматривать на графических устройствах вывода – ГД, ГП. Просмотр осуществляется фрагментами, которые выделяются на поверхности изображения с помощью окна прямоугольной формы. Положение, размер и ориентация (угол поворота вокруг своего центра) этого окна определяют изображение, которое выводится в окно на ГД или на ГП; таким образом, управляя перемещением окна на поверхности изображения, можно просмотреть все изображение, а задавая некоторые характеристики окна – фильтры, можно просматривать

только выделенные (пропущенные фильтром) элементы изображения. Важно отметить, что при визуализации фрагмента изображения на ГД устанавливаются связи между изображением (его дисплейным представлением) и моделью графической информации на поверхности изображения. Благодаря этой связи вся дальнейшая обработка фрагмента изображения может производиться одновременно с изображением на экране ГД и с моделью этого изображения, что весьма существенно при организации работы графического редактора.

Итак, перейдем к описанию графического редактора (ГРЕД). ГРЕД решает задачу ввода и редактирования графической информации, работает с моделью графического изображения, основной структурной единицей которой является графический сегмент. Графический сегмент должен содержать дескриптор сегмента, который определяет общие характеристики сегмента (подсветка, максимальные размеры изображения, система координат и т.п.), кроме того, может содержать примитивы графические (ПГ), атрибутивные (ПА), структурные (ПС). С помощью примитивов можно, с одной стороны, строить достаточно сложные изображения с простой структурой (например, определяя как примитив графический все изображение), а с другой – с помощью примитивов структуры можно построить любой сложности иерархическую структуру внутри сегмента. Ввод графической информации в ПО АРМ осуществляется в терминах примитивов с помощью системы ввода (СВ), а также с помощью базы данных в терминах графических сегментов. С помощью ГРЕД в ПО АРМ осуществляется редактирование графической информации, фрагмент которой визуализирован на экране ГД. Редактирование ведется над изображением в терминах примитивов и других структурных единиц, которые можно каким-либо образом выделить в изображении фрагмента, вообще говоря, независимо от графического сегмента, частями которого, возможно, являются выделенные элементы. Так, выделение может быть пяти уровней:

- элемента изображения (отрезок, точка, символ и т.п.);
- примитивов графических;
- структурных построений внутри сегмента;
- графических сегментов;
- всего изображения.

Выделяя таким образом нужные части изображения, можно производить с ними следующие операции: удаление, замену, удаление дополнения, дублирование, смену атрибутов, аффинные преобразования. Все проводимые преобразования с изображением сохраняются в виде протокола изменений, который затем можно, в свою очередь, отредактировать и применить (благодаря наличию связи между дисплейным представлением графической информации и ее моделью) к модели графического изображения.

В заключение приведем некоторые характеристики СПО ГД и состояние реализации. К настоящему времени система СПО ГД реализована для следующих конфигураций ЭВМ: БЭСМ-6 - АРМ-Р, БЭСМ-6 - АРМ-М, ЕС - АРМ-Р, ЕС - АРМ-М. Операционная среда для АРМов - ДОС-400, для ЕС - ОС 4.0 и выше, для БЭСМ-6 - ОС ДИСПАК.

Л и т е р а т у р а

1. Дебелов В.А., Мацокин А.М., Чубарев А.И. СПО ГД - система программного обеспечения графического диалога. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 64-71.
2. Status Report of Graphic Standards Planning Committee. - Comput. Graph., 1979, v.13, no.3.