1.2. Шинная структура связей

Для достижения максимальной универсальности и упрощения протоколов обмена информацией в *микропроцессорных системах* применяется так называемая шинная структура связей между отдельными устройствами, входящими в систему. Суть шинной структуры связей сводится к следующему.

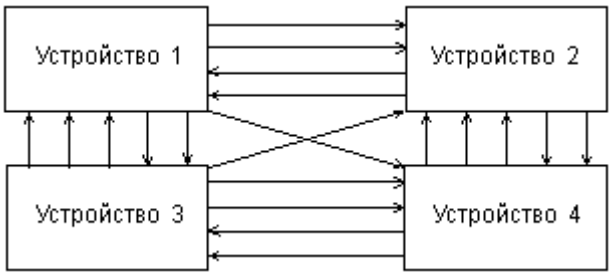

Рис. 1.5. Классическая структура связей.

При классической структуре связей (рис. 1.5) все сигналы и коды между устройствами передаются по отдельным линиям связи. Каждое устройство, входящее в систему, передает свои сигналы и коды независимо от других устройств. При этом в системе получается очень много линий связи и разных протоколов обмена информацией.

При шинной структуре связей (рис. 1.6) все сигналы между устройствами передаются по одним и тем же линиям связи, но в разное время (это называется мультиплексированной передачей). Причем передача по всем линиям связи может осуществляться в обоих направлениях (так называемая двунаправленная передача). В результате количество линий связи существенно сокращается, а правила обмена (протоколы) упрощаются. Группа линий связи, по которым передаются сигналы или коды как раз и называется ***шиной*** (англ. bus).

Понятно, что при шинной структуре связей легко осуществляется пересылка всех информационных потоков в нужном направлении, например, их можно пропустить через один процессор, что очень важно для *микропроцессорной системы*. Однако при шинной структуре связей вся информация передается по линиям связи последовательно во времени, по очереди, что снижает быстродействие системы по сравнению с классической структурой связей.

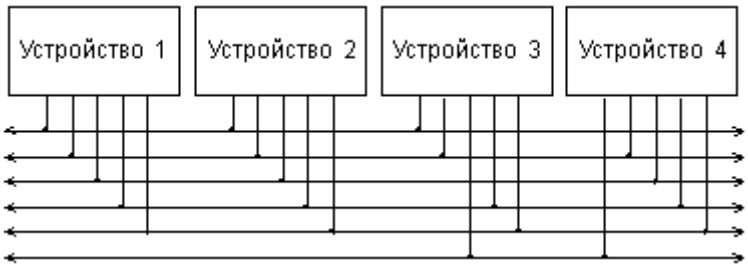

Рис. 1.6. Шинная структура связей.

Большое достоинство шинной структуры связей состоит в том, что все устройства, подключенные к *шине*, должны принимать и передавать информацию по одним и тем же правилам (протоколам обмена информацией по *шине*). Соответственно, все узлы, отвечающие за обмен с *шиной* в этих устройствах, должны быть единообразны, унифицированы.

Существенный недостаток шинной структуры связан с тем, что все устройства подключаются к каждой линии связи параллельно. Поэтому любая неисправность любого устройства может вывести из строя всю систему, если она портит линию связи. По этой же причине отладка системы с шинной структурой связей довольно сложна и обычно требует специального оборудования.

В системах с шинной структурой связей применяют все три существующие разновидности выходных каскадов цифровых микросхем:

- стандартный выход или выход с двумя состояниями (обозначается 2С, 2S, реже ТТЛ, TTL);
- выход с открытым коллектором (обозначается ОК, ОС);
- выход с тремя состояниями или (что то же самое) с возможностью отключения (обозначается 3С, 3S).

Упрощенно эти три типа выходных каскадов могут быть представлены в виде схем на рис. 1.7.

У выхода 2С два ключа замыкаются по очереди, что соответствует уровням логической единицы (верхний ключ замкнут) и логического нуля (нижний ключ замкнут). У выхода ОК замкнутый ключ формирует уровень логического нуля, разомкнутый — логической единицы. У выхода 3С ключи могут замыкаться по очереди (как в случае 2С), а могут размыкаться одновременно, образуя третье, высокоимпедансное, состояние. Переход в третье состояние (Z-состояние) управляется сигналом на специальном входе EZ.

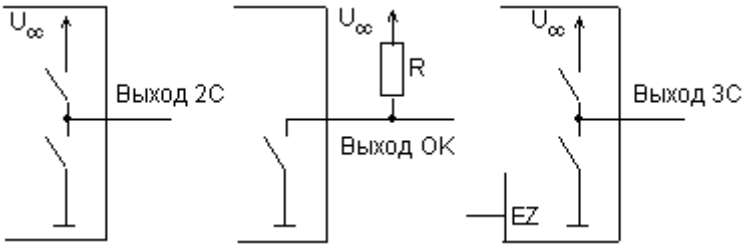

Рис. 1.7. Три типа выходов цифровых микросхем.

Выходные каскады типов 3С и ОК позволяют объединять несколько выходов микросхем для получения мультиплексированных (рис. 1.8) или двунаправленных (рис. 1.9) линий.

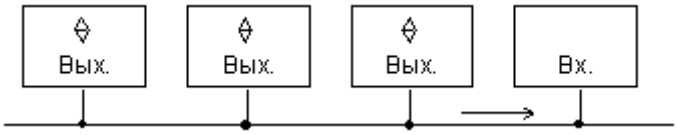

Рис. 1.8. Мультиплексированная линия.

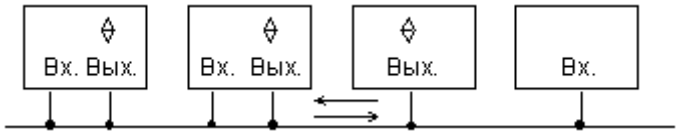

Рис. 1.9. Двунаправленная линия.

При этом в случае выходов 3С необходимо обеспечить, чтобы на линии всегда работал только один активный выход, а все остальные выходы находились бы в это время в третьем состоянии, иначе возможны конфликты. Объединенные выходы ОК могут работать все одновременно, без всяких конфликтов.

Типичная структура *микропроцессорной системы* приведена на рис. 1.10. Она включает в себя три основных типа устройств:

- процессор;
- *память*, включающую оперативную ***память*** (ОЗУ, RAM — Random Access Memory) и постоянную *память* (ПЗУ, ROM —Read Only Memory), которая служит для хранения данных и программ;
- ***устройства ввода/вывода*** (УВВ, I/O — Input/Output Devices), служащие для связи *микропроцессорной системы* с внешними устройствами, для приема (ввода, чтения, Read) входных сигналов и выдачи (вывода, записи, Write) выходных сигналов.

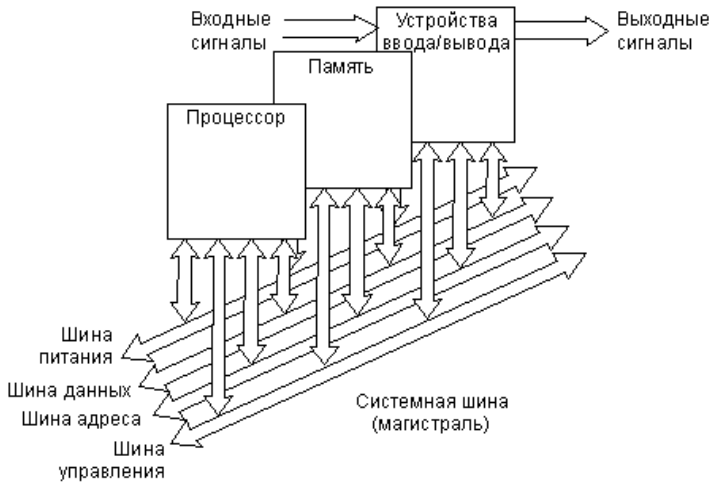

Рис. 1.10. Структура микропроцессорной системы.

Все устройства *микропроцессорной системы* объединяются общей системной ***шиной*** (она же называется еще **системной магистралью** или **каналом**). Системная магистраль включает в себя четыре основные *шины* нижнего уровня:

- *шина* адреса (Address Bus);
- *шина* данных (Data Bus);
- *шина* управления (Control Bus);
- *шина* питания (Power Bus).

Шина адреса служит для определения адреса (номера) устройства, с которым процессор обменивается информацией в данный момент. Каждому устройству (кроме процессора), каждой ячейке *памяти* в *микропроцессорной системе* присваивается собственный адрес. Когда код какого-то адреса выставляется процессором на *шине* адреса, устройство, которому этот адрес приписан, понимает, что ему предстоит обмен информацией. *Шина* адреса может быть однонаправленной или двунаправленной.

Шина данных — это основная *шина*, которая используется для передачи информационных кодов между всеми устройствами *микропроцессорной системы*. Обычно в пересылке информации участвует процессор, который передает код данных в какое-то устройство или в ячейку *памяти* или же принимает код данных из какого-то устройства или из ячейки *памяти*. Но возможна также и передача информации между устройствами без участия процессора. *Шина* данных всегда двунаправленная.

Шина управления в отличие от *шины* адреса и *шины* данных состоит из отдельных управляющих сигналов. Каждый из этих сигналов во время обмена информацией имеет свою функцию. Некоторые сигналы служат для стробирования передаваемых или принимаемых данных (то есть определяют моменты времени, когда информационный код выставлен на *шину* данных). Другие управляющие сигналы могут использоваться для подтверждения приема данных, для сброса всех устройств в исходное состояние, для тактирования всех устройств и т.д. Линии *шины* управления могут быть однонаправленными или двунаправленными.

Наконец, *шина* питания предназначена не для пересылки информационных сигналов, а для питания системы. Она состоит из линий питания и общего провода. В *микропроцессорной системе* может быть один источник питания (чаще +5 В) или несколько источников питания (обычно еще –5 В, +12 В и –12 В). Каждому напряжению питания соответствует своя линия связи. Все устройства подключены к этим линиям параллельно.

Если в *микропроцессорную систему* надо ввести входной код (или входной сигнал), то процессор по *шине* адреса обращается к нужному *устройству ввода/вывода* и принимает по *шине* данных входную информацию. Если из *микропроцессорной системы* надо вывести выходной код (или выходной сигнал), то процессор обращается по *шине* адреса к нужному *устройству ввода/вывода* и передает ему по *шине* данных выходную информацию.

Если информация должна пройти сложную многоступенчатую обработку, то процессор может хранить промежуточные результаты в системной оперативной *памяти*. Для обращения к любой ячейке *памяти* процессор выставляет ее адрес на *шину* адреса и передает в нее информационный код по *шине* данных или же принимает из нее информационный код по *шине* данных. В *памяти* (оперативной и постоянной) находятся также и управляющие коды (команды выполняемой процессором программы), которые процессор также читает по *шине* данных с адресацией по *шине* адреса. Постоянная *память* используется в основном для хранения программы начального пуска *микропроцессорной системы*, которая выполняется каждый раз после включения питания. Информация в нее заносится изготовителем раз и навсегда.

Таким образом, в *микропроцессорной системе* все информационные коды и коды команд передаются по *шинам* последовательно, по очереди. Это определяет сравнительно невысокое быстродействие *микропроцессорной системы*. Оно ограничено обычно даже не быстродействием процессора (которое тоже очень важно) и не скоростью обмена по системной *шине* (магистрали), а именно последовательным характером передачи информации по системной *шине* (магистрали).

Важно учитывать, что *устройства ввода/вывода* чаще всего представляют собой устройства на «жесткой логике». На них может быть возложена часть функций, выполняемых *микропроцессорной системой*. Поэтому у разработчика всегда имеется возможность перераспределять функции системы между аппаратной и программной реализациями оптимальным образом. Аппаратная реализация ускоряет выполнение функции, но имеет недостаточную гибкость. Программная реализация значительно медленнее, но обеспечивает высокую гибкость. Аппаратная реализация функций

увеличивает стоимость системы и ее энергопотребление, программная — не увеличивает. Чаще всего применяется комбинирование аппаратных и программных функций.

Иногда *устройства ввода/вывода* имеют в своем составе процессор, то есть представляют собой небольшую специализированную *микропроцессорную систему*. Это позволяет переложить часть программных функций на *устройства ввода/вывода*, разгрузив центральный процессор системы.

1.3. Режимы работы микропроцессорной системы

Как уже отмечалось, *микропроцессорная система* обеспечивает большую гибкость работы, она способна настраиваться на любую задачу. Гибкость эта обусловлена прежде всего тем, что функции, выполняемые системой, определяются программой (программным обеспечением, software), которую выполняет процессор. Аппаратура (аппаратное обеспечение, hardware) остается неизменной при любой задаче. Записывая в *память* системы программу, можно заставить *микропроцессорную систему* выполнять любую задачу, поддерживаемую данной аппаратурой. К тому же шинная организация связей *микропроцессорной системы* позволяет довольно легко заменять аппаратные модули, например, заменять *память* на новую большего объема или более высокого быстродействия, добавлять или модернизировать *устройства ввода/вывода*, наконец, заменять процессор на более мощный. Это также позволяет увеличить гибкость системы, продлить ее жизнь при любом изменении требований к ней.

Но гибкость *микропроцессорной системы* определяется не только этим. Настраиваться на задачу помогает еще и выбор режима работы системы, то есть режима обмена информацией по системной магистрали (*шине*).

Практически любая развитая *микропроцессорная система* (в том числе и компьютер) поддерживает три основных режима обмена по магистрали:

- программный обмен информацией;
- обмен с использованием прерываний (Interrupts);
- обмен с использованием прямого доступа к *памяти* (ПДП, DMA — Direct Memory Access).

Программный обмен информацией является основным в любой *микропроцессорной системе*. Он предусмотрен всегда, без него невозможны другие режимы обмена. В этом режиме процессор является единоличным хозяином (или задатчиком, Master) системной магистрали. Все операции (циклы) обмена информацией в данном случае инициируются только процессором, все они выполняются строго в порядке, предписанном исполняемой программой.

Процессор читает (выбирает) из *памяти* коды команд и исполняет их, читая данные из *памяти* или из *устройства ввода/вывода*, обрабатывая их, записывая данные в *память* или передавая их в *устройство ввода/вывода*. Путь процессора по программе может быть линейным, циклическим, может содержать переходы (прыжки), но он всегда непрерывен и полностью находится под контролем процессора. Ни на какие внешние события, не связанные с программой, процессор не реагирует (рис. 1.11). Все сигналы на магистрали в данном случае контролируются процессором.

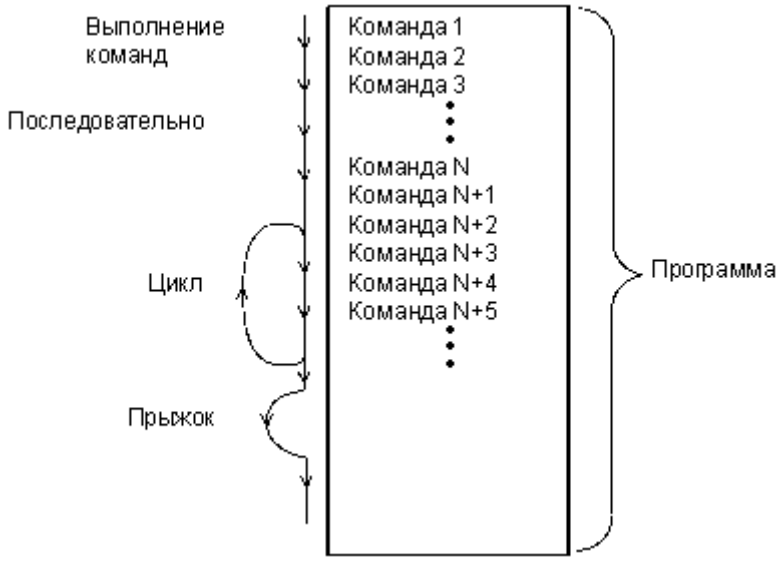

Рис. 1.11. Программный обмен информацией.

Обмен по прерываниям используется тогда, когда необходима реакция *микропроцессорной системы* на какое-то внешнее событие, на приход внешнего сигнала. В случае компьютера внешним событием может быть, например, нажатие на клавишу клавиатуры или приход по локальной сети пакета данных. Компьютер должен реагировать на это, соответственно, выводом символа на экран или же чтением и обработкой принятого по сети пакета.

В общем случае организовать реакцию на внешнее событие можно тремя различными путями:

- с помощью постоянного программного контроля факта наступления события (так называемый метод опроса флага или polling);
- с помощью прерывания, то есть насильственного перевода процессора с выполнения текущей программы на выполнение экстренно необходимой программы;
- с помощью прямого доступа к *памяти*, то есть без участия процессора при его отключении от системной магистрали.

Проиллюстрировать эти три способа можно следующим простым примером. Допустим, вы готовите себе завтрак, поставив на плиту кипятиться молоко. Естественно, на закипание молока надо реагировать, причем срочно. Как это организовать? Первый путь — постоянно следить за молоком, но тогда вы ничего другого не сможете делать. Правильнее будет регулярно поглядывать на молоко, делая одновременно что-то другое. Это программный режим с опросом флага. Второй путь — установить на кастрюлю с молоком датчик, который подаст звуковой сигнал при закипании молока, и спокойно заниматься другими делами. Услышав сигнал, вы выключите молоко. Правда, возможно, вам придется сначала закончить то, что вы начали делать, так что ваша реакция будет медленнее, чем в первом случае. Наконец, третий путь состоит в том, чтобы соединить датчик на кастрюле с управлением плитой так, чтобы при закипании молока горелка была выключена без вашего участия (правда, аналогия с ПДП здесь не очень точная, так как в данном случае на момент выполнения действия вас не отвлекают от работы).

Первый случай с опросом флага реализуется в *микропроцессорной системе* постоянным чтением информации процессором из *устройства ввода/вывода*, связанного с тем внешним устройством, на поведение которого необходимо срочно реагировать.

Во втором случае в режиме прерывания процессор, получив запрос прерывания от внешнего устройства (часто называемый IRQ — Interrupt ReQuest), заканчивает выполнение текущей команды и переходит к программе обработки прерывания. Закончив выполнение программы обработки прерывания, он возвращается к прерванной программе с той точки, где его прервали (рис. 1.12).

Здесь важно то, что вся работа, как и в случае программного режима, осуществляется самим процессором, внешнее событие просто временно отвлекает его. Реакция на внешнее событие по прерыванию в общем случае медленнее, чем при программном режиме. Как и в случае программного обмена, здесь все сигналы на магистрали выставляются процессором, то есть он полностью контролирует магистраль. Для обслуживания прерываний в систему иногда вводится специальный модуль контроллера прерываний, но он в обмене информацией не участвует. Его задача состоит в том, чтобы упростить работу процессора с внешними запросами прерываний. Этот контроллер обычно программно управляется процессором по системной магистрали

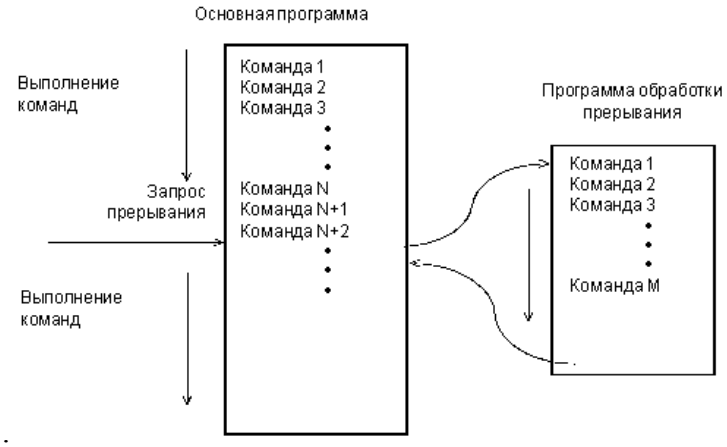

Рис. 1.12. Обслуживание прерывания.

Естественно, никакого ускорения работы системы прерывание не дает. Его применение позволяет только отказаться от постоянного опроса флага внешнего события и временно, до наступления внешнего события, занять процессор выполнением каких-то других задач.

Прямой доступ к памяти (ПДП, DMA) — это режим, принципиально отличающийся от двух ранее рассмотренных режимов тем, что обмен по системной *шине* идет без участия процессора. Внешнее устройство, требующее обслуживания, сигнализирует процессору, что режим ПДП необходим, в ответ на это процессор заканчивает выполнение текущей команды и отключается от всех *шин*, сигнализируя запросившему устройству, что обмен в режиме ПДП можно начинать.

Операция ПДП сводится к пересылке информации из *устройства ввода/вывода* в *память* или же из *памяти* в *устройство ввода/вывода*. Когда пересылка информации будет закончена, процессор вновь возвращается к прерванной программе, продолжая ее с той точки, где его прервали (рис. 1.13). Это похоже на режим обслуживания прерываний, но в данном случае процессор не участвует в обмене. Как и в случае прерываний, реакция на внешнее событие при ПДП существенно медленнее, чем при программном режиме.

Понятно, что в этом случае требуется введение в систему дополнительного устройства (контроллера ПДП), которое будет осуществлять полноценный обмен по системной магистрали без всякого участия процессора. Причем процессор предварительно должен сообщить этому контроллеру ПДП, откуда ему следует брать информацию и/или куда ее следует помещать. Контроллер ПДП может считаться специализированным процессором, который отличается тем, что сам не участвует в обмене, не принимает в себя информацию и не выдает ее (рис. 1.14).

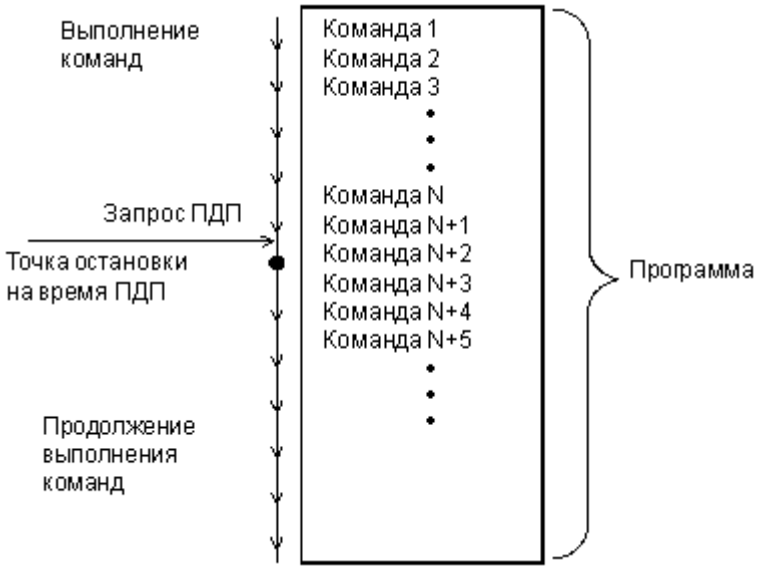

Рис. 1.13. Обслуживание ПДП.

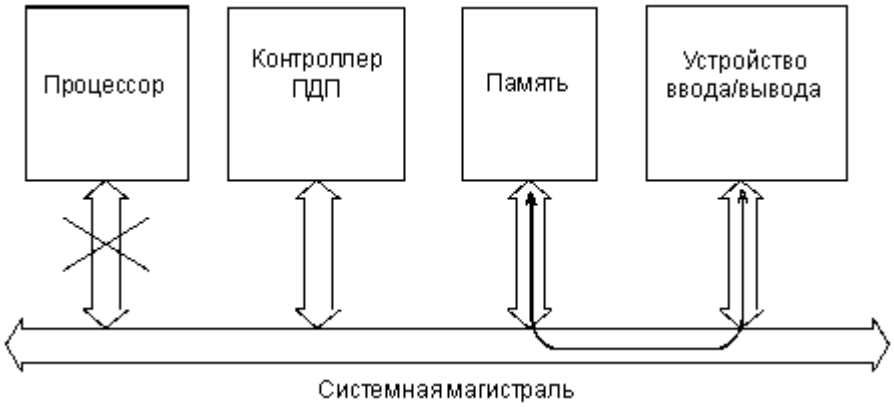

Рис. 1.14. Информационные потоки в режиме ПДП.

В принципе контроллер ПДП может входить в состав *устройства ввода/вывода*, которому необходим режим ПДП или даже в состав нескольких *устройств ввода/вывода*. Теоретически обмен с помощью прямого доступа к *памяти* может обеспечить более высокую скорость передачи информации, чем программный обмен, так как процессор передает данные медленнее, чем специализированный контроллер ПДП. Однако на практике это преимущество реализуется далеко не всегда. Скорость

обмена в режиме ПДП обычно ограничена возможностями магистрали. К тому же необходимость программного задания режимов контроллера ПДП может свести на нет выигрыш от более высокой скорости пересылки данных в режиме ПДП. Поэтому режим ПДП применяется редко.

Если в системе уже имеется самостоятельный контроллер ПДП, то это может в ряде случаев существенно упростить аппаратуру *устройств ввода/вывода*, работающих в режиме ПДП. В этом, пожалуй, состоит единственное бесспорное преимущество режима ПДП.