**САРОВСКИЙ ГОСУДАРСТВЕННЫЙ
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ**

Факультет информационных технологий и электроники

Кафедра вычислительной и информационной техники

В.А. Павлов

Система ввода-вывода ПК.
Параллельный порт.

Учебное пособие и практикум

**г. Саров
2005г.**

САРОВСКИЙ ГОСУДАРСТВЕННЫЙ
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет информационных технологий и электроники

Кафедра вычислительной и информационной техники

В.А. Павлов

Система ввода-вывода ПК.
Параллельный порт.

Учебное пособие и практикум

Утверждено:

На заседании кафедры ВИТ

_____ 2005г

Научно-методическим Советом факультета

_____ 2005г

Научно-методическим Советом СарФТИ

_____ 2005г.

Саров
СарФТИ
2005

УДК 681.3.06
П12

Одобрено Научно-методическим советом Саровского государственного физико-технического института

Павлов Виктор Александрович
Система ввода-вывода ПК. Параллельный порт. Учебно-методическое пособие и практикум по циклу "Система ввода-вывода ПК". СарФТИ, Саров, 2005. - 204 с.: ил.

Настоящее учебное пособие является дальнейшим расширением методической поддержки лабораторных практикумов цикла "Система ввода-вывода ПК". В СарФТИ этот цикл реализуется в спецкурсе "Интерфейсы периферийных устройств" и курсе специализации "Адаптеры и контроллеры ЭВМ" в рамках специальности 230101 "Вычислительные машины, комплексы, системы и сети".

В пособии рассмотрены принципы работы и программирования параллельных портов ПК в различных режимах работы (SPP, Bi-Di, EPP, ECP). Приведено описание пяти лабораторных работ, в которых рассмотрены вопросы практического использования параллельных портов для передачи и приема данных по программным каналам ввода/вывода и по каналу прямого доступа к памяти, вопросы программной реализации различных протоколов параллельного интерфейса, а также вопросы организации обмена данными между компьютерами через параллельные порты, работающие в режимах Bi-Di, ECP и EPP.

В приложении приведено описание программных средств (отладчиков DEBUG и AFD), используемых для отладки управляющих программ, реализующих алгоритмы инициализации портов и организации обмена данными. Приведено также описание системы команд процессора i8086.

ВВЕДЕНИЕ

Под системой ввода-вывода (СВВ) компьютеров понимают [1] совокупность аппаратных и программных средств, обеспечивающих обмен информацией между объектами (компонентами) внешнего мира и оперативной памятью (ОП) компьютера. Внешний мир по отношению к компьютеру представляет собой совокупность всех источников и потребителей информации. Компонентами (объектами) внешнего мира являются: человек, объекты управления, другие компьютеры, внешние коммуникационные системы (локальные вычислительные сети, телефонные сети и т.п.) а также носители информации внешней памяти (магнитные и оптические диски, магнитные ленты и т.п.). Устройства СВВ, предназначенные для взаимодействия и обмена информацией с объектами внешнего мира, называются периферийными (ПУ).

Помимо периферийных устройств в состав СВВ входит система внутренних и внешних аппаратных *интерфейсов*, *адаптеры и контроллеры* ПУ, контроллеры стандартных портов ввода-вывода, каналообразующие контроллерные и процессорные устройства (контроллеры прямого доступа к памяти, контроллеры аппаратных прерываний, процессоры ввода-вывода, центральный процессор) и ряд других вспомогательных устройств (программируемый таймер, CMOS/RTC и т.п.). Процессоры, контроллеры и вспомогательные устройства СВВ, за исключением центрального процессора, относятся к так называемой системной логике, которая в настоящее время реализуется в рамках сверхбольших интегральных схем - "чипсетов". "Чипсеты", на базе которых формируется внутренняя коммуникационная часть (среда) СВВ, размещаются на системных (материнских платах) (северный и южный мосты, хабы и т.д.). "Чипсеты", реализующие функции адаптеров и контроллеров ПУ могут размещаться на материнских платах, на устройствах расширения (видеоадаптер и т.п.) или на самих периферийных устройствах (дисковых накопителях, сетевых адаптерах и т.д.).

Под *интерфейсом* в общем случае понимают совокупность средств и правил, обеспечивающих взаимодействие устройств цифровой вычислительной системы и (или) программ. К средствам аппаратных интерфейсов относятся линии связи, по которым передаются информационные, адресные и управляющие сигналы, разъемные соединители и схемы управления работой интерфейса, реализуемые, как правило, на базе специализированных и универсальных процессорных устройств - *контроллеров.* К правилам аппаратного интерфейса можно отнести сигнальный *протокол* взаимодействия через интерфейс, поддерживаемые режимы передачи (дуплексный, полудуплексный, симплексный), поддерживаемые методы связи (синхронный, асинхронный) и т.п.

В общем случае под *протоколом* понимают совокупность правил и соглашений, определяющих работу функциональных устройств (контроллеров, адаптеров, ПУ и т.п.) и (или) процедур в процессе взаимодействия (связи). При взаимодействии вычислительной системы с объектами внешнего мира используются протоколы нескольких уровней, число которых может достигать семи. При организации передачи информации через физическую структуру СВВ используются протоколы двух нижних уровней: канального и физического. Протоколы более высоких уровней реализуются в операционных системах (ОС) в рамках логической организации их систем ввода-вывода. Работа аппаратных интерфейсов, связанная с передачей и приемом сигналов по линиям (среде) интерфейса определяется протоколами физического уровня. Взаимодействие процессора с компонентами СВВ и с некоторыми компонентами объектов внешнего мира (СВВ других компьютеров, функциональные компоненты вычислительных сетей (мосты, коммутаторы и т.п.) и телекоммуникационных систем и т.д.) определяется протоколами канального уровня. Эти протоколы используются для формирования в аппаратной среде СВВ каналов передачи данных между ОП и конкретными ПУ или портами ввода-вывода, а также для непосредственной передачи данных через программные каналы ввода-вывода. Эти протоколы реализуют, как правило, программы - драйверы соответствующих компонентов СВВ (драйвер последовательного или параллельного порта, драйвер сетевого адаптера, драйвер принтера и т.д.). В ряде

случаев функции протоколов канального уровня могут распределяться между системными программными средствами и контроллерами устройств СВВ, например, контроллером шины USB, контроллером сетевого адаптера и т.п.

К *адаптерам* относят компоненты СВВ, которые предназначены для сопряжения между собой устройств с различными способами представления данных или устройств, использующих различные виды унифицированных сопряжений (интерфейсов). Адаптеры, как правило, выполняются на базе одного или нескольких универсальных и специализированных контроллеров и поддерживают соответствующие интерфейсы на своих входах и выходах.

Центральный процессор, оперативная память и аппаратные средства СВВ составляют аппаратную платформу компьютера и определяют его архитектуру. Для практического использования (оживления) эта платформа добавляется программными и информационными средствами базовой системы ввода-вывода (BIOS - Basic Input-Output System) устанавливаемыми на материнскую плату и ряд устройств расширения (видеоадаптер) в виде микросхем постоянной памяти (ПЗУ). Это позволяет при включении питания компьютера автоматически инициализировать компоненты СВВ, обеспечить интерфейс доступа программ к средствам СВВ, а также устанавливать на аппаратной платформе компьютера операционные системы (ОС), которые, в свою очередь, являются программной средой для выполнения прикладных программ. В рамках ОС формируется логическая структура СВВ, имеющая свой программно-аппаратный интерфейс с аппаратной частью СВВ в виде программ-драйверов соответствующих устройств СВВ. Это обеспечивает прикладным программам независимый от конкретной аппаратной платформы компьютера интерфейс доступа к объектам внешнего мира, например, интерфейс пользователя, обеспечивающий интерактивный режим взаимодействия пользователя с ОС и приложениями.

Системы ввода-вывода современных компьютеров достаточно динамично развиваются, отслеживая бурное развитие процессорных устройств, что, в свою очередь, является тонусом для совершенствования программных средств (ОС, приложений и т.п.), ориентированных на эффективное использование вычислительной мощности процессоров и коммуникационных возможностей средств СВВ. Эта динамика предъявляет определенные требования к организации информационно-методического обеспечения курсов, связанных с изучением СВВ компьютеров.

Учебно-методическое пособие "Параллельный порт ПК" посвящено изучению принципов работы и программирования параллельного порта персонального компьютера (ПК), который является достаточно универсальным компонентом системы ввода/вывода ПК. Современный параллельный порт ПК поддерживает основные режимы обмена, характерные для СВВ ПК: обмен по программным каналам ввода/вывода с поддержкой аппаратных прерываний и обмен по каналу прямого доступа к памяти. Через параллельный порт можно подключать к ПК не только принтеры, но и ряд универсальных и специфических периферийных устройств, включая другие ПК. Параллельный порт имеет системную поддержку на уровне BIOS.

Принципы организации обмена данными через параллельный порт в той или иной мере характерны и для других интерфейсов СВВ ПК (последовательный и игровой порты, интерфейсы гибких и жестких дисков, интерфейсов типа USB, FireWire и др.). В связи с этим достаточно подробное знакомство с принципами работы, применения и программирования параллельного порта позволит значительно облегчить процесс освоения и других интерфейсных компонентов СВВ ПК. Эти принципы используются для разработки драйверов устройств или для реализации обмена с ПУ и объектами внешнего мира непосредственно в рамках прикладных программ.

Учебно-методическое пособие состоит из трех частей.

В первой части приведено подробное описание принципов работы и реализации параллельного порта, его системной поддержки на уровне BIOS, его программирования на уровне регистров контроллера порта и на уровне функций прерывания BIOS.

Во второй части представлены описания пяти лабораторных работ, в которых рассмотрены вопросы практического использования порта в различных режимах его работы для обмена данными с ПУ и другими компьютерами с программной и аппаратной реализацией различных протоколов параллельного интерфейса.

Третья часть представляет собой приложение, в котором приводится описание программных средств отладки программ, реализующих алгоритмы инициализации контроллера порта в различных режимах и алгоритмы обмена данными через параллельный порт с реализацией различных протоколов канального и физического уровней. В приложении приведено также описание системы команд процессора i8086.

Данное учебное пособие предназначено для расширения спектра лабораторных работ практикума цикла "Система ввода-вывода ПК", реализуемого в СарФТИ в спецкурсе "Интерфейсы периферийных устройств" и в курсе специализации "Адаптеры и контроллеры ЭВМ" в рамках специальности 230101 "Вычислительные машины, комплексы, системы и сети".

Контрольные вопросы.

1. Что понимается под системой ввода-вывода (СВВ) компьютера?
2. Что относится к объектам внешнего мира компьютера?
3. Какие компоненты СВВ относятся к периферийным устройствам?
4. Что помимо ПУ входит в состав аппаратной части СВВ?
5. Какие компоненты СВВ относятся к системной логике?
6. Что в рамках СВВ формируется на базе "Чипсетов"?
7. Где могут размещаться "Чипсеты"?
8. Что понимается под интерфейсом?
9. Что можно отнести к средствам аппаратных интерфейсов?
10. Что можно отнести к правилам аппаратных интерфейсов?
11. Что понимается под протоколом взаимодействия?
12. Сколько уровней протоколов взаимодействия может быть использовано при взаимодействии с компонентами внешнего мира?
13. Какие уровни протоколов обмена используются при организации обмена данными через физическую среду СВВ?
14. Где реализуются протоколы с уровнем выше канального?
15. Что определяют протоколы физического уровня?
16. Что определяют протоколы канального уровня?
17. Что понимается под адаптерами СВВ?
18. Что входит в состав аппаратной платформы компьютера?
19. Что такое BIOS и для чего она используется в ПК?
20. Где и для чего формируется логическая структура СВВ?

1. ПАРАЛЛЕЛЬНЫЙ ИНТЕРФЕЙС: LPT-ПОРТ

Порт параллельного интерфейса был введен в PC для подключения принтера — отсюда и пошло его название LPT-порт (Line PrinTer — построчный принтер). Традиционный, он же стандартный, LPT-порт (так называемый *SPP-порт)* ориентирован на вывод данных, хотя с некоторыми ограничениями позволяет и вводить данные. Существуют различные модификации LPT-порта — двунаправленный, EPP, ECP и другие, расширяющие его функциональные возможности, повышающие производительность и снижающие нагрузку на процессор. Поначалу они являлись фирменными решениями отдельных производителей, позднее был принят стандарт IEEE 1284.

С внешней стороны порт имеет 8-битную шину данных, 5-битную шину сигналов состояния и 4-битную шину управляющих сигналов, выведенные на разъем-розетку DB-25S. В LPT-порте используются логические уровни ТТЛ, что ограничивает допустимую длину кабеля из-за невысокой помехозащищенности ТТЛ-интерфейса. Гальваническая развязка отсутствует — схемная земля подключаемого устройства соединяется со схемной землей компьютера. Из-за этого порт является уязвимым местом компьютера, страдающим при нарушении правил подключения и заземления устройств. Поскольку порт обычно располагается на системной плате, в случае его «выжигания» зачастую выходит из строя и его ближайшее окружение, вплоть до выгорания всей системной платы.

С точки зрения программного взаимодействия LPT-порт представляет собой набор регистров, расположенных в пространстве ввода-вывода. Регистры порта адресуются относительно базового адреса порта, стандартными значениями которого являются 3BCh, 378h и 278h. Порт может использовать линию запроса аппаратного прерывания, обычно IRQ7 или IRQ5. В расширенных режимах может использоваться и канал DMA.

Порт имеет системную поддержку на уровне BIOS: поиск установленных портов во время теста POST, сервисы печати Int 17h и INT 05h и сервисы EPP BIOS.

Практически все современные системные платы (еще начиная с PCI-плат для процессоров i486) имеют встроенный адаптер LPT-порта. Существуют интерфейсные карты (устройства расширения) ISA с LPT-портом, где он чаще всего соседствует с парой COM-портов, а также с контроллерами дисковых интерфейсов (FDC+IDE). LPT-порт обычно присутствовал и на видеоадаптерах MDA и HGC. Есть и карты PCI с дополнительными LPT-портами.

К современным LPT-портам подключают принтеры, плоттеры, сканеры, коммуникационные устройства и устройства хранения данных, а также электронные ключи, программаторы и прочие устройства. Иногда параллельный интерфейс используют для связи между двумя компьютерами — получается сеть, «сделанная на коленке» (Lap Link).

1.1. Традиционный LPT-порт

Традиционный, он же стандартный, LPT-порт называется *стандартным параллельным портом* (Standard Parallel Port, SPP), или *SPP-портом,* и является однонаправленным портом. Разводка сигналов на разъеме SPP-порта приведена в табл. 1.1.

Таблица 1.1. Разъем стандартного LPT-порта

Контакт DB-25S	№ провода в кабеле	Назначение			Контакт на разъеме принтера (36 контактов)
		I/O[1]	Бит[2]	Сигнал	
1	1	O/I	CR.0\	Strobe#	1
2	3	O(I)	DR.0	Data 0 (DB0)	2
3	5	O(I)	DR.1	Data 1 (DB1)	3
4	7	O(I)	DR.2	Data 2 (DB2)	4
5	9	O(I)	DR.3	Data 3 (DB3)	5
6	11	O(I)	DR.4	Data 4 (DB4)	6
7	13	O(I)	DR.5	Data 5 (DB5)	7
8	15	O(I))	DR.6	Data 6 (DB6)	8
9	17	O(I))	DR.7	Data 7 (DB7)	9
10	19	I[3]	SR.6	Ack#	10
11	21	I	SR.7\	Busy	11
12	23	I	SR.5	PaperEnd (PE)	12
13	25	I	SR.4	Select (SLCT)	13
14	2	O/I	CR.1\	Auto LF# (AutoFeed#)	14
15	4	I	SR.3	Error#	32
16	6	O/I	CR.2	Init#	31
17	8	O/I	CR.3\	Select In#	36
18—25	10,12,14, 16, 18, 20,22,24,26	—	—	—	19-30, 33

[1] I/O задает направление передачи (вход-выход) сигнала порта. O/I обозначает выходные линии, состояние которых считывается при чтении из портов вывода; O(I) — выходные линии, состояние которых может быть считано только при особых условиях (см. ниже).

[2] Символом «\» отмечены инвертированные сигналы (1 в регистре соответствует низкому уровню линии).

[3] Вход Ack# соединен резистором (10 кОм) с питанием +5 В.

Примечание: идентификаторы сигналов с низким активным уровнем или оканчиваются знаком #, или сверху метятся штрихом, или начинаются знаком -.

Через SPP-порт программно можно реализовать большинство параллельных интерфейсов. Сервис BIOS Int 17h (драйвер принтера), программно реализует протокол обмена интерфейса *Centronics*.

1.1.1. Внутреннее устройство порта

Общая схема SPP-порта представлена на рис. 1.1. Восьмибитовые данные заносятся в DD1 во время записи в регистр с адресом *базовый адрес + 0 (BASE + 0)*. Операция осуществляется командой WRITE_DATA#.

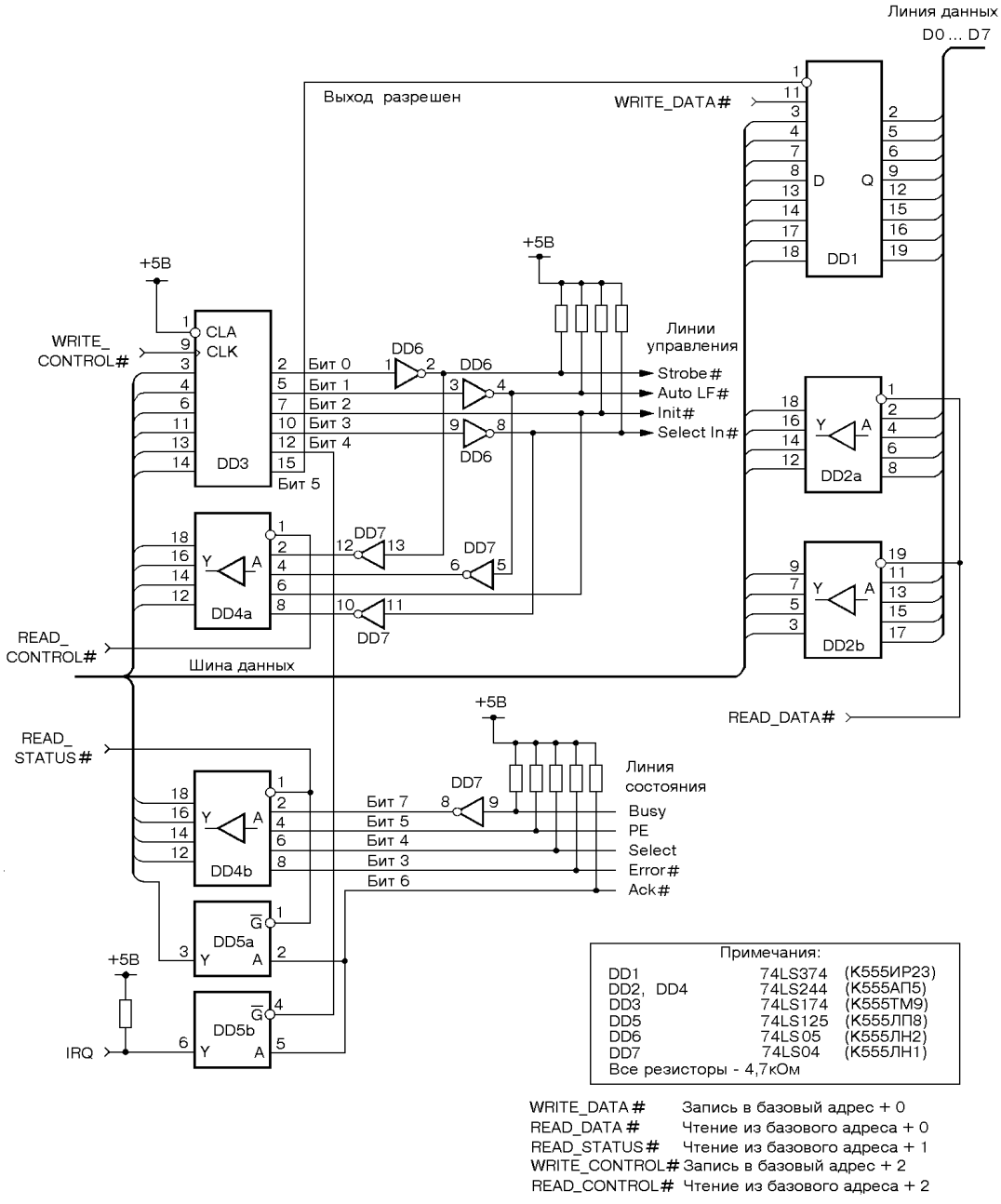


Рис. 1 1. Общая схема параллельного порта

Эти данные образуют группу. Они считываются компьютером из того же регистра через DD2 с помощью команды READ_DATA#. Шести битовое управляющее слово записывается в DD3 через регистр с

адресом *BASE + 2* при помощи команды WRITE_CONTROL#. Биты с 0 по 3 подаются на выход разъема и образуют группу управления. Некоторые биты инвертируются микросхемами с открытыми коллекторами на выходе (DD6 и DD7). Все выходные линии подключены к питанию +5 В через резисторы 4,7 кОм. Состояние этих линий считывается через регистр с адресом *BASE + 2* через DD4 посредством команды READ_CONTROL#. Четвертый бит управляющего байта разрешает прерывание, а пятый бит открывает или закрывает выход DD1. Состояние пяти контактов разъема порта (группа состояния) компьютер считывает через DD4 с помощью команды READ_STATUS# через регистр с адресом *BASE + 1*. Входы линии подключены к питанию +5 В через резисторы 4,7 кОм, один вход инвертируется.

В первых конструкциях IBM PC контакт «выход разрешен» DD1 соединялся с «землей» для постоянного открывания выходов. Это была однонаправленная версия параллельного порта. Начиная с IBM PS/2, указанный контакт соединили с пятым битом регистра управления DD3 (см. рис. 1.1.), и порт стал двунаправленным. Следует отметить, что многие параллельные порты, поставляемые со встроенными картами ввода/вывода, двунаправленные. Для любого контакта следует избегать короткого замыкания и/или соединения с шиной питания.

В этом разделе рассматривается однонаправленный параллельный порт. На рис. 1.2. представлена его логическая структура. Контакты порта образуют три группы: данных, управления и состояния.

Группа данных посылает данные от ПК на внешние устройства. Имеет восемь выходных линий и ассоциируется с байтом в адресном пространстве ввода/вывода процессоров x86. Адрес: *BASE*.

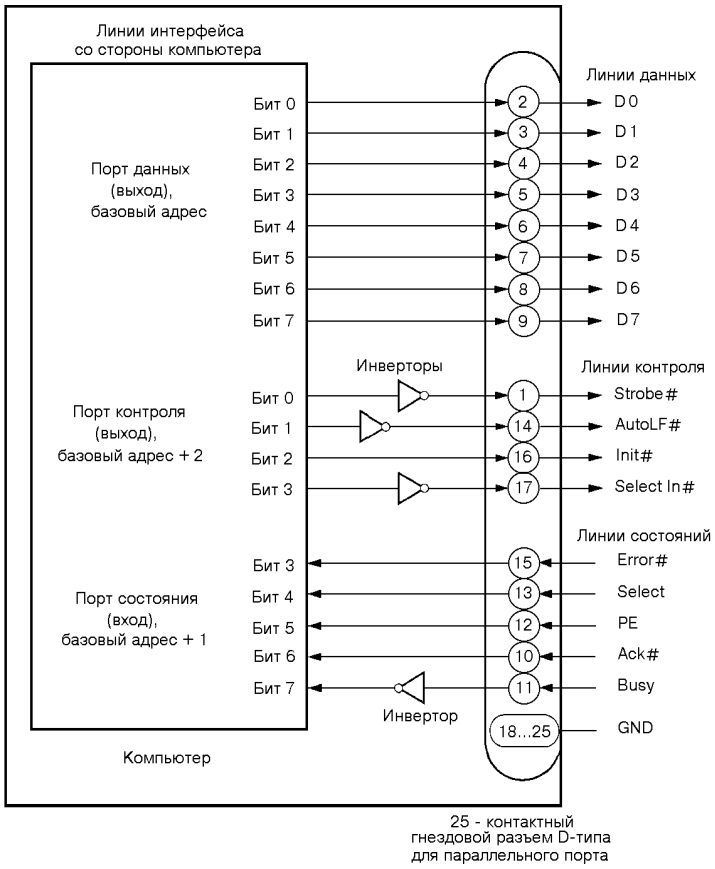


Рис. 1.2. Логическая структура параллельного порта

Группа управления контролирует внешнее устройство. Имеет четыре инвертированные выходные линии (Strobe#, Auto LF#, Select In# и Init#). Адрес группы управления: *BASE + 2*.

Группа состояния может использоваться компьютером для получения текущего состояния внешнего устройства, ее адрес: *BASE + 1*. Имеет пять линий (Error#, Select, PE, Ack# и Busy). Линии Error# и Ack# инвертированы, остальные - нет.

1.1.2. Регистры SPP-порта.

Для управления работой порта, определения состояния порта и подключенного к нему устройства а также программной реализации протокола передачи данных адаптер SPP-порта содержит три 8-битных регистра, расположенных по соседним адресам в пространстве ввода-вывода, начиная с базового адреса порта BASE (3BCh, 378h или 278h).

Data Register **(DR) — *регистр данных***, адрес: BASE. Данные, записанные в этот регистр, *выводятся* на выходные линии Data[7:0]. Данные, считанные из этого регистра, в зависимости от схемотехники адаптера соответствуют либо ранее записанным данным, либо сигналам, поступающим на вход порта со стороны

внешнего устройства.

Status Register **(SR) — *регистр состояния*** (только чтение), адрес: BASE+1. Регистр отображает *5-битный порт ввода* сигналов состояния принтера и флаг прерывания. Бит SR.7 инвертируется — низкому уровню сигнала соответствует единичное значению бита в регистре, и наоборот.

Ниже приведено назначение бит регистра состояния (в скобках даны альтернативные названия разрядов).

- SR. 7 — BUSY (nBusy) - инверсное отображение состояния линии Busy: при низком уровне на линии устанавливается единичное значения бита — разрешение на вывод очередного байта.
- SR. 6 — ACK (nAck) - отображение состояния линии Ack# .
- SR. 5 — PE (PErrог) - отображение состояния линии Paper End. Единичное значение соответствует высокому уровню линии, устанавливается при отсутствии бумаги в принтере.
- SR. 4 — Select - отображение состояния линии Select. Единичное значение соответствует высокому уровню линии — сигналу о включении принтера.
- SR. 3 — Error (nFault) - отображение состояния линии Error# . Нулевое значение соответствует низкому уровню линии — сигналу о любой ошибке принтера.
- SR. 2—PIRQ - флаг прерывания по сигналу Ack#. Бит обнуляется, если сигнал Ack# вызвал аппаратное прерывание. Единичное значение устанавливается по аппаратному сбросу и после чтения регистра состояния.
- SR [1, 0] - зарезервированы.

Control Register **(CR) — *регистр управления,*** anpec=BASE+2, допускает запись и чтение. Регистр связан с *4-битным портом вывода управляющих сигналов* (биты 0-3) для которых возможно и чтение; выходной буфер обычно имеет тип «открытый коллектор». Это позволяет корректно использовать линии данного регистра как входные при программировании их в высокий уровень. Биты 0, 1, 3 инвертируются.

Ниже приведено назначение бит регистра управления.

- CR [7:6] — зарезервированы.
- CR. 5 — Dir (direction) - бит управления направлением передачи (только для портов PS/2, см. ниже и в портах, поддерживающих стандарт IEEE 1284). Запись единицы переводит порт данных в режим ввода. При чтении регистра CR, состояние бита не определено.
- CR. 4— IRQE (ackIntEn) - разрешение прерывания. Единичное значение разрешает прерывание по спаду сигнала на линии Ack# — сигнал подтверждения приема и запроса следующего байта.
- CR. 3—SELIN (Selectin) - единичное значение бита соответствует низкому уровню на выходе Select In# - сигналу, разрешающему работу принтера по интерфейсу *Centronics.*
- CR. 2— INIT (ninit) - нулевое значение бита соответствует низкому уровню на выходе Init# - сигнал аппаратного сброса принтера.
- CR. 1 — AUTO LF (autofd) - единичное значение бита соответствует низкому уровню на выходе Auto LF# - сигналу на автоматический перевод строки (LF — Line Feed) по приему байта возврата каретки (CR). Иногда сигнал и бит называют AutoFD или AutoFDXT.
- CR. 0 — STRB (strobe) - единичное значение бита соответствует низкому уровню на выходе Strobe# — сигналу стробирования выходных данных.

Запрос аппаратного прерывания (обычно IRQ7 или IRQ5) вырабатывается по отрицательному перепаду сигнала на выводе 10 разъема интерфейса (Ack#) при установке CR. 4=1. Во избежание ложных прерываний контакт 10 соединён резистором с шиной +5 В. Прерывание вырабатывается, когда принтер подтверждает прием предыдущего байта. BIOS это прерывание не использует и не обслуживает.

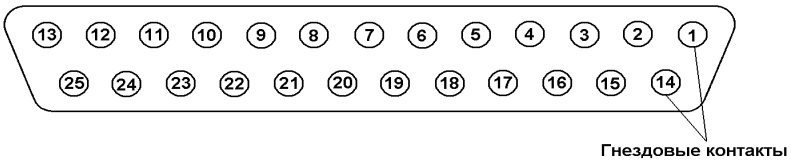
1.1.3. Интерфейс Centronics

Параллельный интерфейс Centronics ориентирован на передачу потока байт данных принтеру и прием сигналов состояния принтера. Этот интерфейс поддерживается всеми LPT-портами и большинством принтеров с параллельным интерфейсом. Его отечественным аналогом является интерфейс *ИРПР-М.* Понятие Centronics относится как к набору сигналов и протоколу взаимодействия, так и к разъемам, устанавливаемым на ПК и принтерах. Разъемы порта для компьютера и принтера отличаются друг от друга. Первый - это 25-контактная розетка D-типа (рис. 1.3а.), а второй - 36-контактная розетка параллельного типа (рис. 1.3б.). Для соединения компьютера с принтером используется интерфейсный принтерный кабель (рис. 1.4) длиной не более 5 м.

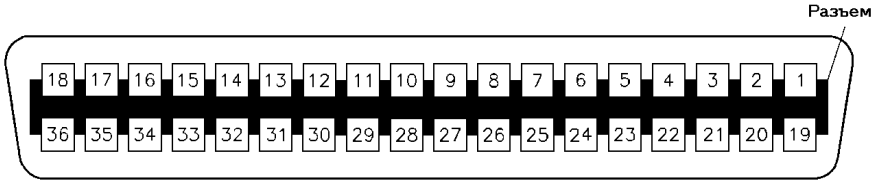
Назначение сигналов интерфейса и разводка их по контактам разъема принтера приведены в табл. 1.2, а разъема компьютера - в табл.1.1. Временные диаграммы обмена с принтером представлены на рис. 1.5.

Передача данных начинается с проверки готовности принтера — состояния линии Busy. Низкий уровень этого сигнала информирует о готовности принтера к приему байта данных. Контроллер порта выдает байт данных на шину данных и затем формирует строб-импульс данных. Длительность этого импульса должна быть больше длительности переходных процессов в линиях данных интерфейса и может составлять доли микросекунды. Порт может заканчивать его формирование не дожидаясь изменения сигнала

Busy с низкого уровня на высокий. Во время строба данные должны быть действительными. Подтверждением приема байта (символа) является низкий уровень сигнала Ack#, который вырабатывается после приема данных по заднему фронту строба через неопределенное время. Импульс Ack# является также запросом принтера на прием следующего байта, его задействуют для формирования сигнала прерывания от порта принтера. Если прерывания не используются, то сигнал Ack# игнорируется и весь обмен управляется парой сигналов Strobe# и Busy. Свое состояние принтер может сообщить порту по линиям Select, Error# и



а)



б)

Рис. 1.3. Контакты разъема параллельного порта ПК и разъема принтера.

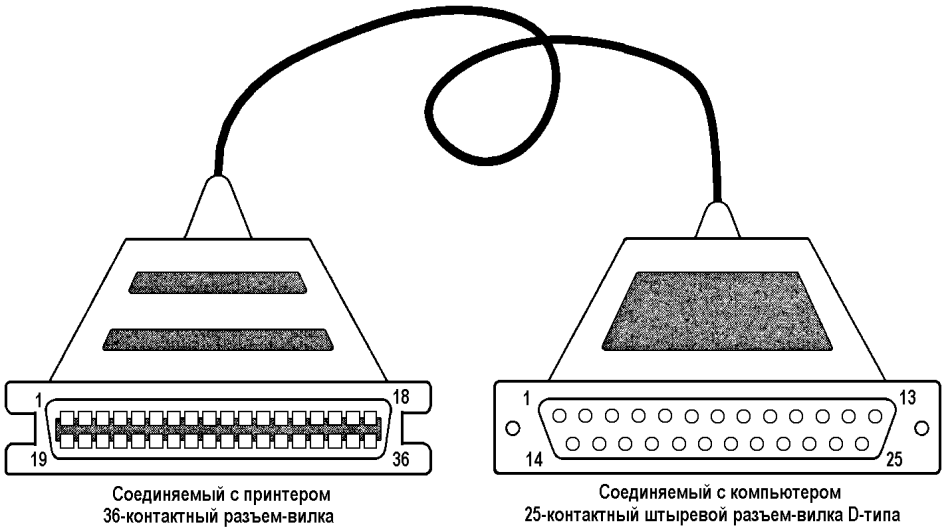


Рис. 1.4. Интерфейсный кабель принтера.

Таблица 1.2. Сигналы интерфейса Centronics на контактах разъема принтера.

Сигнал	I/O*	Контакт	Назначение
Strobe#	I	1	Строб данных. Данные фиксируются по низкому уровню или по заднему фронту сигнала
Data [0:7]	I	2-9	Линии данных. Data 0 (контакт 2) — младший бит
Ack#	O	10	Acknowledge — импульс подтверждения приема байта (запрос на прием следующего). Может использоваться для формирования запроса прерывания
Busy	O	11	Занято. Прием данных возможен только при низком уровне сигнала
PaperEnd (PE)	O	12	Высокий уровень сигнализирует о конце бумаги
Select	O	13	Сигнализирует о включении принтера (обычно в принтере соединяется резистором с цепью +5 В)
Auto LF#	I	14	Автоматический перевод строки. При низком уровне принтер, получив символ CR (Carriage Return — возврат каретки), автоматически выполняет и функцию LF (Line Feed — перевод строки)
Error#	O	32	Ошибка: конец бумаги, состояние OFF-Line или внутренняя ошибка принтера
Init#	I	31	Инициализация (сброс в режим параметров умолчания, возврат к началу строки)

Select In#	I	36	Выбор принтера (низким уровнем). При высоком уровне принтер не воспринимает остальные сигналы интерфейса
GND	-	19-30 33	Общий провод интерфейса

* I/O задает направление (вход/выход) применительно к принтеру.

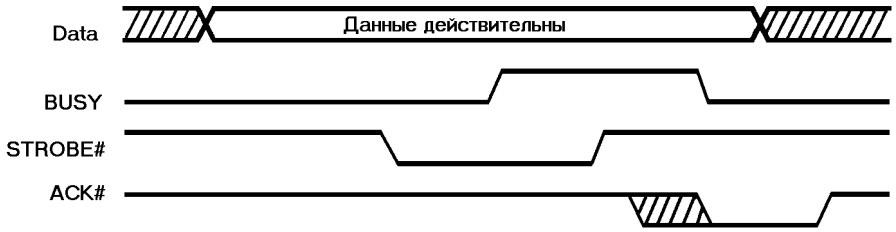


Рис. 1.5. Передача данных по протоколу Centronics

PaperEnd — по ним можно определить, включен ли принтер, исправен ли он и заправлен ли он бумагой. Формированием импульса на линии Init# производится общий сброс (инициализация) принтера (при этом принтер очистит и весь свой буфер данных). Режимом автоматического перевода строки, как правило, не используется, и сигнал AutoLF# имеет высокий уровень. Сигнал SelectIn# позволяет логически отключать принтер от интерфейса.

Перечислим шаги программной реализации процедуры вывода байта по интерфейсу Centronics с указанием требуемого количества шинных операций процессора.

1. Вывод байта в регистр данных (1 цикл IOWR#).
2. Ввод из регистра состояния и проверка готовности устройства (бит SR. 7 — сигнал Busy). Этот шаг зацикливается до получения готовности или до срабатывания программного тайм-аута (минимум 1 цикл IORD#).
3. По получению готовности выводом в регистр управления устанавливается строб данных, а следующим выводом строб снимается. Обычно, чтобы переключить только один бит (строб), регистр управления предварительно считывается, что к двум циклам IOWR# добавляет еще один цикл IORD#.

Видно, что для вывода одного байта требуется 4-5 операций ввода-вывода с регистрами порта (в лучшем случае, когда готовность обнаружена по первому чтению регистра состояния). Помимо этого параллельный порт является относительно медленным устройством, поэтому при работе с высокопроизводительными процессорами в нем предусмотрена выработка сигнала READY (на шине ISA обозначается сигналом IOCHRDY), который затягивает выполнение команд обращения к регистрам порта переводя процессор в циклы ожидания примерно на одну микросекунду. Отсюда вытекает главный недостаток вывода через стандартный порт — невысокая скорость обмена при значительной загрузке процессора. Порт удается разогнать до скоростей 100 - 150 Кбайт/с при полной загрузке процессора. Другой недостаток функциональный — сложность использования его в качестве порта ввода.

Стандартный порт асимметричен — при наличии 12 линий (и бит), нормально работающих на вывод, на ввод работает только 5 линий состояния. Если необходима симметричная двунаправленная связь, на всех стандартных портах работоспособен *режим полубайтного обмена — Nibble Mode.* В этом режиме, называемом также *Hewlett Packard Bi-tronics,* одновременно принимаются 4 бита данных, пятая линия используется для квитирования. Таким образом, каждый байт передается за два цикла, а каждый цикл требует по крайней мере 5 операций ввода-вывода.

Схемотехника выходных буферов данных LPT-портов отличается большим разнообразием. На многих старых моделях адаптеров SPP-порт данных можно использовать и для организации ввода. Если в порт данных записать байт с единицами во всех разрядах, а на выходные линии интерфейса через микросхемы с выходом типа «открытый коллектор» подать какой-либо код (или соединить ключами какие-то линии со схемной землей), то этот код может быть считан из того же регистра данных. Однако выходным цепям передатчика информации придется «бороться» с выходным током логической единицы выходных буферов адаптера. Схемотехника ТТЛ такие решения не запрещает, но если внешнее устройство выполнено на микросхемах КМОП, их мощности может не хватить для «победы» в этом шинном конфликте. Однако современные адаптеры часто имеют в выходной цепи согласующий резистор с сопротивлением до 50 Ом.

Выходной ток короткого замыкания выхода на землю обычно не превышает 30 мА. Простой расчет показывает, что даже в случае короткого замыкания контакта разъема на землю при выводе «единицы» на этом резисторе падает напряжение 1,5 В, что входной схемой приемника будет воспринято как «единица». Поэтому нельзя полагать, что такой способ ввода будет работать на всех компьютерах. На некоторых старых адаптерах портов выходной буфер отключается перемычкой на плате. Тогда порт превращается в обыкновенный порт ввода.

Относительно близким родственником интерфейса Centronics является и интерфейс *ИРПР (интерфейс радиальный, параллельный),* имеющий следующие отличия:

- Линии данных инвертированы.
- Протокол квитирования несколько иной.
- Ко всем входным линиям (на принтере) подключены пары согласующих резисторов: 220 Ом к питанию +5 В и 330 Ом к общему проводу. Это позволяет использовать длинные кабели, но перегружает большинство интерфейсных адаптеров PC.
- Сигнал ошибки (и конца бумаги) отсутствует.

Интерфейс ИРПР может быть программно реализован через обычный LPT-порт. Однако, для устранения перегрузки выходных линий, согласующие резисторы из принтера желательно удалить. Порт, перегруженный по выходу, может преподносить самые неожиданные сюрпризы (естественно, только неприятные и трудно диагностируемые).

1.2. Расширения параллельного порта

Недостатки стандартного порта частично устраняли новые типы портов, появившиеся в компьютерах PS/2.

Двунаправленный порт 1 (Type 1 parallel port) — интерфейс, введенный в PS/2. Такой порт кроме стандартного режима может работать в режиме ввода или двунаправленном режиме. Протокол обмена формируется программно, а для указания направления передачи в регистр управления порта введен специальный бит CR. 5: 0 — буфер данных работает на вывод, 1 — на ввод. Не путайте этот порт, называемый также *enhanced bi-directional*, с EPP. Данный тип порта «прижился» и в обычных компьютерах, в CMOS Setup он может называться PS/2 или Bi-Di.

Порт с прямым доступом к памяти (Type 3 DMA parallel port) применялся в PS/2 моделей 57, 90, 95. Был введен для повышения пропускной способности и разгрузки процессора при выводе на принтер. Программе, работающей с портом, требовалось только задать в памяти блок данных, подлежащих выводу, а затем вывод по протоколу Centronics производился без участия процессора.

Позже появились другие адаптеры LPT-портов, реализующие протокол обмена Centronics аппаратно, — ***Fast Centronics.*** Некоторые из них использовали FIFO-буфер данных — ***Parallel Port FIFO Mode***. Не будучи стандартизованными, такие порты разных производителей требовали наличия собственных специальных драйверов. Программы, использующие прямое управление регистрами стандартных портов, не могли задействовать их дополнительные возможности. Такие порты часто входили в состав мультикарт VLB. Существуют их варианты с шиной ISA, а также встроенные в системную плату.

Контрольные вопросы

1. Для чего в состав ПК был введен параллельный порт?
2. Какие разновидности параллельного порта используются в современных ПК?
3. Охарактеризуйте аппаратную часть параллельного порта.
4. Охарактеризуйте параллельный порт с точки зрения программного взаимодействия с ним.
5. Перечислите средства системной поддержки параллельного порта на уровне BIOS.
6. Для чего используются параллельные порты.
7. Охарактеризуйте внутренне устройство SPP-порта.
8. Охарактеризуйте логическую структуру SPP-порта.
9. Охарактеризуйте регистры SPP-порта.
10. Дайте краткую характеристику интерфейсу Centronics.
11. Охарактеризуйте процесс передачи данных через интерфейс Centronics.
12. Охарактеризуйте шаги программной реализации протокола передачи интерфейса Centronics.
13. Чем ограничена скорость выдачи данных через SPP-порт?
14. В чем заключается асимметричность SPP-порта?
15. Дайте краткую характеристику режиму обмена Nibble Mode.
16. Охарактеризуйте проблемы, возникающие при использовании SPP-порта для приема и передачи данных в байтовом формате.
17. Охарактеризуйте отличие параллельного интерфейса ИРПР от Centronics.
18. Охарактеризуйте расширения параллельного порта, используемые до введения стандарта IEEE 1284.

В данном разделе использованы материалы из [3, 4, 6].

1.3. Стандарт IEEE 1284

Стандарт на параллельный интерфейс *IEEE 1284,* принятый в 1994 году, описывает порты *SPP, EPP* и *ECP*. Стандарт определяет 5 режимов обмена данными, метод согласования режима, физический и электрический интерфейсы. Согласно IEEE 1284, возможны следующие режимы обмена данными через параллельный порт:

- *Режим совместимости (Compatibility Mode)* — однонаправленный (вывод) по протоколу Centronics. Этот режим соответствует SPP-порту.
- *Полубайтный режим (Nibble Mode)* — ввод байта в два цикла (по 4 бита), используя для приема линии состояния. Этот режим обмена подходит для любых адаптеров, поскольку задействует только возможности стандартного порта.
- *Байтный режим (Byte Mode)* — ввод байта целиком, используя для приема линии данных. Этот режим работает только на портах, допускающих чтение выходных данных *(Bi-Directional или PS/2 Type 1,* см. выше).
- *Режим EPP (EPP Mode)* — двунаправленный обмен данными (EPP означает Enhanced Parallel Port). Управляющие сигналы интерфейса генерируются аппаратно во время цикла обращения к порту. Эффективен при работе с устройствами внешней памяти и адаптерами локальных сетей.
- *Режим ECP (ECP Mode)* — двунаправленный обмен данными с возможностью аппаратного сжатия данных по методу *RLE* (Run Length Encoding) и использования FIFO-буферов и DMA (ECP означает Extended Capability Port). Управляющие сигналы интерфейса генерируются аппаратно. Эффективен для принтеров и сканеров (здесь может использоваться сжатие) и различных устройств блочного обмена.

Стандарт определяет способ, по которому ПО может определить режим, доступный и хосту (PC), и периферийному устройству (или присоединенному второму компьютеру). Режимы нестандартных портов, реализующих протокол обмена Centronics аппаратно *(Fast Centronics, Parallel Port FIFO Mode),* могут и не являться режимами IEEE 1284, несмотря на наличие в них черт EPP и ECP.

В компьютерах с LPT-портом на системной плате режим — SPP, EPP, ECP или их комбинация — задается в BIOS Setup. Режим совместимости полностью соответствует SPP-порту. Остальные режимы подробно рассмотрены ниже.

При описании режимов обмена фигурируют следующие понятия:

- *хост* — компьютер, обладающий параллельным портом;
- *ПУ*— периферийное устройство, подключаемое к этому порту;
- *Per* — в названиях сигналов обозначает передающее ПУ;
- *прямой канал* — канал вывода данных от хоста в ПУ;
- *обратный канал* — канал ввода данных в хост из ПУ.

1.3.1. Полубайтный режим ввода — Nibble Mode

Полубайтный режим предназначен для двунаправленного обмена и может работать на всех стандартных портах. Порты имеют *5 линий ввода* состояния, используя которые ПУ может посылать в хост байт тетрадами (nibble - полубайт, 4 бита) за два приема. Сигнал Ack#, вызывающий прерывание, которое может использоваться в данном режиме, соответствует биту 6 регистра состояния, что усложняет программные манипуляции с битами при сборке байта. Сигналы порта приведены в табл. 1.3, временные диаграммы - на рис. 1.6.

Таблица 1.3. Сигналы LPT-порта в полубайтном режиме ввода

Контакт	Сигнал SPP	I/O	Бит	Описание
14	AutoFeed#	O	CR.1\	HostBusy - сигнал квитирования. Низкий уровень означает готовность к приему тетрады, высокий подтверждает прием тетрады
17	SelectIn#	O	CR.3\	Высокий уровень указывает на обмен в режиме IEEE 1284 (в режиме SPP уровень низкий)
10	Ack#	I	SR.6	PerClk. Низкий уровень означает готовность тетрады, высокий - ответ на сигнал HostBusy
11	Busy	I	SR.7	Прием бита данных 3, затем бита 7
12	PE	I	SR.5	Прием бита данных 2, затем бита 6
13	Select	I	SR.4	Прием бита данных 1, затем бита 5
15	Error#	I	SR.3	Прием бита данных 0, затем бита 4

Прием байта данных в полубайтном режиме состоит из следующих фаз:

1. Хост сигнализирует о готовности приема данных установкой низкого уровня на линии HostBusy.
2. ПУ в ответ помещает тетраду на входные линии состояния.
3. ПУ сигнализирует о готовности тетрады установкой низкого уровня на линии PerClk.
4. Хост устанавливает высокий уровень на линии HostBusy, указывая на занятость приемом и

обработкой тетрады.
5. ПУ отвечает установкой высокого уровня на линии PerClk.
6. Шаги 1-5 повторяются для второй тетрады.

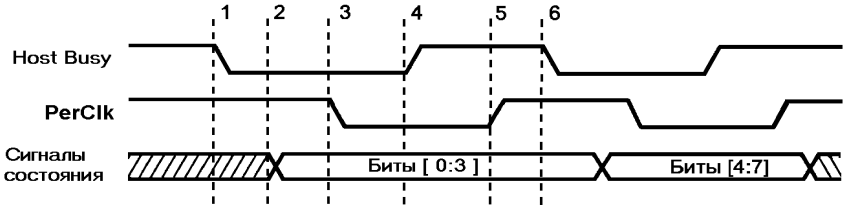


Рис. 1.6. Прием данных в полубайтном режиме

Полубайтный режим сильно нагружает процессор, и поднять скорость обмена выше 50 Кбайт/с не удается. Безусловное его преимущество в том, что он работает *на всех портах*. Его применяют в тех случаях, когда поток данных невелик (например, для связи с принтерами). Однако при связи с адаптерами локальных сетей, внешними дисковыми накопителями и CD-ROM прием больших объемов данных требует изрядного терпения со стороны пользователя.

1.3.2. Двунаправленный байтный режим — Byte Mode

В этом режиме данные принимаются с использованием двунаправленного порта, у которого выходной буфер данных может отключаться установкой бита CR. 5-1. Как и предыдущие, режим является программно-управляемым — все сигналы квитирования анализируются и устанавливаются драйвером. Сигналы порта представлены в табл. 1.4, временные диаграммы — на рис. 1.7.

Таблица 1.4. Сигналы LPT-порта в байтном режиме ввода-вывода

Контакт	Сигнал SPP	Имя в байтном режиме	I/O	Бит	Описание
1	Strobe#	HostClk	O	CR.0\	Импульс (низкого уровня) подтверждает прием байта в конце каждого цикла
14	AutoFeed#	HostBusy	O	CR.1\	Сигнал квитирования. Низкий уровень означает готовность хоста принять байт; высокий уровень устанавливается по приему байта
17	SelectIn#	1284Active	O	CR.3\	Высокий уровень указывает на обмен в режиме IEEE 1284 (в режиме SPP уровень низкий)
16	Init#	Init#	O	CR.2	Не используется; установлен высокий уровень
10	Ack#	PerClk	I	SR.6	Устанавливается в низкий уровень для индикации действительности данных на линиях данных Data [0:7] в ответ на сигнал HostBusy
11	Busy	PerBusy	I	SR.7	Состояние занятости прямого канала
12	PE	AckDataReq[1]	I	SR.5	Устанавливается ПУ для указания на наличие обратного канала передачи
13	Select	Xflag[1]	I	SR.4	Флаг расширяемости
15	Error#	DataAvail#[1]	I	SR.3	Устанавливается ПУ для указания на наличие обратного канала передачи
2-9	Data [0:7]	Data [0:7] I/O	I/O	DR[0:7]	Двунаправленный (прямой и обратный) канал данных

[1] Сигналы действуют в последовательности согласования (см. ниже).

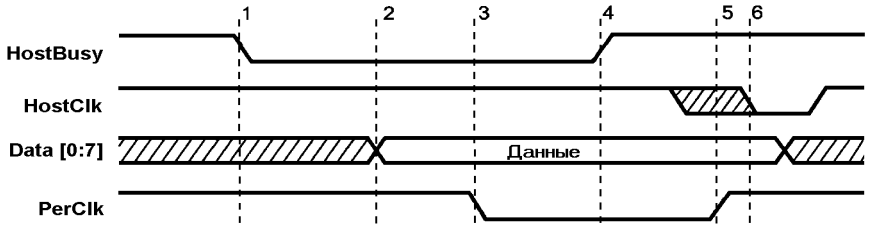


Рис. 1.7. Прием данных в байтном режиме

Фазы приема байта данных перечислены ниже.

1. Хост сигнализирует о готовности приема данных установкой низкого уровня на линии HostBusy.
2. ПУ в ответ помещает байт данных на линии Data [0:7].
3. ПУ сигнализирует о действительности байта установкой низкого уровня на линии PerClk.
4. Хост устанавливает высокий уровень на линии HostBusy, указывая на занятость приемом и обработкой байта.

5. ПУ отвечает установкой высокого уровня на линии PerClk.
6. Хост подтверждает прием байта импульсом HostClk.

Шаги 1-6 повторяются для каждого следующего байта. Квитирование осуществляется парой сигналов HostBusy и PerClk; ПУ может и не использовать сигнал HostClk (это приглашение к выдаче следующего байта, напоминающее сигнал Ack# в интерфейсе Centronics). Побайтный режим позволяет поднять скорость обратного канала до скорости прямого канала в стандартном режиме. Однако он способен работать *только на двунаправленных портах,* которые раньше применялись в основном на малораспространенных машинах PS/2, но практически все современные порты можно сконфигурировать на двунаправленный режим (в настройках BIOS Setup - Bi-Di или PS/2).

1.3.3. Режим EPP

Порот *EPP* (Enhanced Parallel Port — улучшенный параллельный порт) был разработан компаниями Intel, Xircom и Zenith Data Systems задолго до принятия стандарта IEEE 1284. Этот порт предназначен для повышения производительности обмена по параллельному интерфейсу со сканерами и внешними дисководами. Он впервые был реализован в чипсете Intel 386SL (микросхема 82360) и впоследствии принят множеством компаний как дополнительный протокол параллельного порта. Стандарт EPP предусматривает возможность соединения подключаемых к параллельному порту устройств в цепочки, для чего EPP - устройство снабжается двумя 25-контактными разъемами (входным и выходным) и специальным коммутатором, делающим устройство «невидимым» для компьютера, когда оно не используется. В одну цепочку можно включить до 8 EPP - устройств; кроме того, в конце цепочки может присутствовать устройство, не поддерживающее стандарт EPP (например, принтер, работающий только в режимах SPP и ECP). Цепочку устройств можно подключить к компьютеру напрямую или через мультиплексор. Мультиплексор может иметь до восьми выходных разъемов, так что общее количество подключенных устройств может достигать 72 (восемь цепочек по девять устройств).

Версии протокола EPP-порта, реализованные до принятия IEEE 1284, отличаются от нынешнего стандарта (см. ниже). Протокол EPP обеспечивает четыре типа циклов обмена:

- запись данных;
- чтение данных;
- запись адреса;
- чтение адреса.

Назначение циклов записи и чтения данных очевидно. Адресные циклы используются для передачи адресной, канальной и управляющей информации. Циклы обмена данными отличаются от адресных циклов применяемыми стробирующими сигналами. Назначение сигналов порта EPP и их связь с сигналами SPP объясняются в табл. 1.5.

Таблица 1.5. Сигналы LPT-порта в режиме ввода-вывода EPP

Контакт	Сигнал SPP	Имя в EPP	I/O	Описание
1	Strobe#	Write#	O	Низкий уровень — цикл записи, высокий — цикл чтения
14	AutoLF#	DataStb#	O	Строб данных. Низкий уровень устанавливается в циклах передачи данных
17	Selectln#	AddrStb#	O	Строб адреса. Низкий уровень устанавливается в адресных циклах
16	Init#	Reset#	O	Сброс ПУ (низким уровнем)
10	Ack#	INTR#	I	Прерывание от ПУ
11	Busy	Wait#	I	Сигнал квитирования. Низкий уровень разрешает начало цикла (установку строба в низкий уровень), переход в высокий — разрешает завершение цикла (снятие строба)
2-9	Data [0:7]	AD[0:7]	I/O	Двунаправленная шина адреса/данных
12	PaperEnd	AckDataReq[1]	I	Используется по усмотрению разработчика периферии
13	Select	Xflag[1]	I	Используется по усмотрению разработчика периферии
15	Error#	DataAvail#[1]	I	Используется по усмотрению разработчика периферии

[1] Сигналы действуют в последовательности согласования (см. ниже).

EPP-порт *имеет расширенный набор регистров* (табл. 1.6), который занимает в пространстве ввода-вывода 5-8 смежных байт. В отличие от программно-управляемых режимов, рассмотренных выше, внешние сигналы EPP-порта для каждого цикла обмена формируются аппаратно по одной операции записи или чтения в регистр порта. На рис. 1.8 приведена диаграмма *цикла записи* данных, иллюстрирующая внешний цикл обмена, *вложенный* в цикл записи системной шины процессора (иногда эти циклы называют *связанными).* Адресный цикл записи отличается от цикла данных только стробом внешнего интерфейса.

Таблица 1.6. Регистры EPP-порта

Имя регистра	Смещение	Режим	R/W	Описание
SPP Data Port	+0	SPP/EPP	W	*Регистр данных SPP*
SPP Status Port	+1	SPP/EPP	R	*Регистр состояния SPP*
SPP Control Port	+2	SPP/EPP	W	*Регистр управления SPP*
EPP Address Port	+3	EPP	R/W	*Регистр адреса EPP.* Чтение или запись в него генерирует связанный цикл чтения или записи адреса EPP
EPP Data Port	+4	EPP	R/W	*Регистр данных EPP.* Чтение (запись) генерирует связанный цикл чтения (записи) данных EPP
Not Defined	+5...+7	EPP	N/A	В некоторых контроллерах могут использоваться для 16-32-битных операций ввода-вывода

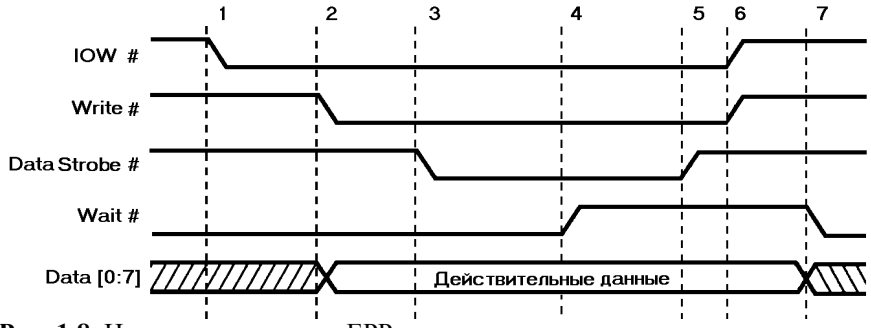


Рис. 1.8. Цикл записи данных EPP

Цикл записи данных состоит из следующих фаз.

1. Программа выполняет цикл вывода (IOWR#) в порт Base+4 (EPP Data Port).
2. Адаптер устанавливает сигнал Write# (низкий уровень), и данные помещаются на выходную шину LPT-порта.
3. При низком уровне Wait# устанавливается строб данных.
4. Порт ждет подтверждения от ПУ (перевода Wait# в высокий уровень).
5. Снимается строб данных — внешний EPP-цикл завершается.
6. Завершается процессорный цикл вывода.
7. ПУ устанавливает низкий уровень Wait#, указывая на возможность начала следующего цикла.

Пример адресного *цикла чтения* приведен на рис. 1.9. Цикл чтения данных отличается только применением другого стробирующего сигнала.

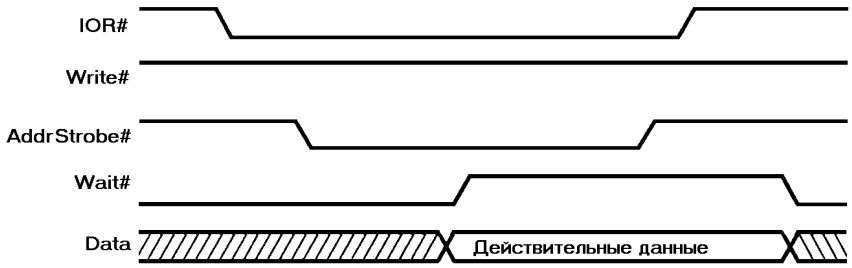


Рис. 1.9. Адресный цикл чтения EPP

Главной отличительной чертой EPP является выполнение внешней передачи во время одного процессорного цикла ввода-вывода. Это позволяет достигать высоких скоростей обмена (0,5-2 Мбайт/с). ПУ, подключенное к параллельному порту EPP, может работать со скоростью устройства, подключаемого через слот ISA.

Протокол блокированного квитирования *(interlocked handshakes)* позволяет автоматически настраиваться на скорость обмена, доступную и хосту, и ПУ. ПУ может регулировать длительность всех фаз обмена с помощью всего лишь одного сигнала Wait#. Протокол автоматически подстраивается под длину кабеля — вносимые задержки приведут только к удлинению цикла. Поскольку кабели, соответствующие стандарту IEEE 1284 (см. выше), имеют одинаковые волновые свойства для разных линий, нарушения передачи, связанного с «состязаниями» сигналов, происходить не должно. При подключении сетевых адаптеров или внешних дисков к EPP-порту можно наблюдать непривычное явление: снижение производительности по мере удлинения интерфейсного кабеля.

Естественно, ПУ не должно «подвешивать» процессор на шинном цикле обмена. Это гарантирует механизм тайм-аутов PC, который принудительно завершает любой цикл обмена, длящийся более 15 мкс. В ряде реализации EPP за тайм-аутом интерфейса следит сам адаптер — если ПУ не отвечает в течение определенного времени (5 мкс), цикл прекращается и в дополнительном (не стандартизованном) регистре

состояния адаптера фиксируется ошибка.

Устройства с интерфейсом EPP, разработанные до принятия IEEE 1284, отличаются началом цикла: строб DataStb# или AddrStb# устанавливается независимо от состояния WAIT#. Это означает, что ПУ не может задержать начало следующего цикла (хотя может растянуть его на требуемое время). Такая спецификация называется EPP 1.7 (предложена Xircom). Именно она применялась в контроллере 82360. Периферия, совместимая с IEEE 1284 EPP, будет нормально работать с контроллером EPP 1.7, но ПУ в стандарте EPP 1.7 может отказаться работать с контроллером EPP 1284.

С *программной* точки зрения контроллер EPP-порта выглядит просто (см. табл. 6). К трем регистрам стандартного порта, имеющим смещение 0, 1 и 2 относительно базового адреса порта, добавлены два регистра (EPP Address Port и EPP Data Port), чтение и запись в которые вызывает генерацию связанных внешних циклов.

Назначение регистров стандартного порта сохранено для совместимости EPP-порта с ПУ и ПО, рассчитанными на применение программно-управляемого обмена. Поскольку сигналы квитирования адаптером вырабатываются аппаратно, при записи в регистр управления CR биты 0, 1 и 3, соответствующие сигналам Strobe#, AutoFeed# и SelectIn# должны иметь нулевые значения. Программное вмешательство могло бы нарушить последовательность квитирования. Некоторые адаптеры имеют специальные средства защиты *(EPP Protect),* при включении которых программная модификация этих бит блокируется. Использование регистра данных EPP позволяет осуществлять передачу блока данных с помощью одной инструкции REP INSB или REP OUTSB. Некоторые адаптеры допускают *16/32-битное обращение к* регистру данных EPP. При этом адаптер просто дешифрует адрес со смещением в диапазоне 4-7 как адрес регистра данных EPP, но процессору сообщает о разрядности 8 бит. Тогда 16- или 32-битное обращение по адресу регистра данных EPP приведет к автоматической генерации двух или четырех шинных циклов по нарастающим адресам, начиная со смещения 4. Эти циклы будут выполняться быстрее, чем то же количество одиночных циклов. Более «продвинутые» адаптеры для адреса регистра данных EPP сообщают разрядность 32 бит и для них до 4 байт может быть передано за один цикл обращения процессора. Таким образом обеспечивается производительность до 2 Мбайт/с, достаточная для адаптеров локальных сетей, внешних дисков, стримеров и CD-ROM. Адресные циклы EPP всегда выполняются только в однобайтном режиме.

Важной чертой EPP является то, что обращение процессора к ПУ осуществляется в реальном времени — нет буферизации. Драйвер способен отслеживать состояние и подавать команды в точно известные моменты времени. Циклы чтения и записи могут чередоваться в произвольном порядке или идти блоками. Такой тип обмена удобен для *регистро-ориентированных ПУ* или ПУ, работающих в *реальном времени,* например устройств сбора информации и управления. Этот режим пригоден и для устройств хранения данных, сетевых адаптеров, принтеров, сканеров и т. п.

EPP-порт имеет системную поддержку в виде EPP BIOS, однако она была разработана со значительным опозданием и до сих пор не используется значительным числом изготовителей системных плат. Помимо этого режим EPP поддерживается не всеми портами — он отсутствует, к примеру, в ряде блокнотных ПК. Так что при разработке собственных устройств ради большей совместимости с компьютерами приходится ориентироваться на режим ECP.

1.3.4. Режим ECP

Параллельный порт *ECP* (Extended Capability Port — порт с расширенными возможностями) был предложен Hewlett Packard и Microsoft для связи с ПУ типа принтеров или сканеров. Как и EPP, данный порт обеспечивает высокопроизводительный двунаправленный обмен данными хоста с ПУ.

1.3.4.1. Протокол ECP

Протокол ECP-порта в обоих направлениях обеспечивает два типа циклов:

- циклы записи и чтения данных;
- командные циклы записи и чтения.

Командные циклы подразделяются на два типа: передача канальных адресов и передача счетчика *RLC* (Run-Length Count).

В отличие от EPP вместе с протоколом ECP сразу появился стандарт на программную (регистровую) модель его адаптера, изложенный в документе «The IEEE 1284 Extended Capabilities Port Protocol and ISA Interface Standard» компании Microsoft. Этот документ определяет свойства протокола, не заданные стандартом IEEE 1284:

- компрессия данных хост-адаптером по методу RLE;
- буферизация FIFO для прямого и обратного каналов;
- применение DMA и программного ввода-вывода.

Компрессия в реальном времени по методу *RLE* (Run-Length Encoding) позволяет достичь коэффициента сжатия 64:1 при передаче растровых изображений, которые имеют длинные строки повторяющихся байт. Компрессию можно использовать, только если ее поддерживают и хост, и ПУ.

Канальная адресация ECP применяется для адресации множества логических устройств, входящих в одно физическое. Например, в комбинированном устройстве факс/принтер/модем, подключаемом только к

одному параллельному порту, возможен одновременный прием факса и печать на принтере. В режиме SPP, если принтер установит сигнал занятости, канал будет занят данными, пока принтер их не примет. В режиме ECP программный драйвер просто адресуется к другому *логическому каналу* того же порта.

Протокол ECP переопределяет сигналы SPP (табл. 1.7).

Таблица 1.7. Сигналы LPT-порта в режиме ввода-вывода ECP

Контакт	Сигнал SPP	Имя в ECP	I/O	Описание
1	Strobe# (Strobe)[2]	HostClk	O	Строб данных, используется в паре с PeriphAck для передачи в прямом направлении (вывод)
14	AutoLF# (Autofd)[2]	HostAck	O	Указывает тип цикла (команда/данные) при передаче в прямом направлении. Используется как сигнал подтверждения в паре с PeriphClk для передачи *в* обратном направлении
17	Selectin# (Selectin)[2]	1284Active	O	Высокий уровень указывает на обмен в режиме IEEE 1284 (в режиме SPP уровень низкий)
16	Init# (ninit)[2]	ReverseRequest#	O	Запрос реверса.Низкий уровень сигнализирует о переключении канала на передачу в обратном направлении
10	Ack# (nAck)[2]	PeriphClk	I	Строб данных, используется в паре с HostAck для передачи в обратном направлении
11	Busy (nBusy)[2]	PeriphAck	I	Используется как сигнал подтверждения в паре с HostClk для передачи в прямом направлении. Индицирует тип команда/ данные при передаче в обратном направлении
12	PaperEnd (PError)[2]	AckReverse#	I	Подтверждение реверса. Переводится в низкий уровень в ответ на ReverseRequest#
13	Select	Xflag[1]	I	Флаг расширяемости
15	Error# (nFault)[2]	PeriphRequest#[1]	I	Устанавливается ПУ для указания на доступность (наличие) обратного канала передачи[1]
2-9	Data [0:7]	Data [0:7]	I/O	Двунаправленный канал данных

[1] Сигналы действуют в последовательности согласования (см. ниже);

2 Обозначение сигналов в разделе 1.9.

Адаптер ECP тоже генерирует внешние протокольные сигналы квитирования аппаратно, но его работа существенно отличается от режима EPP.

На рис. 1.10, *а* приведена диаграмма двух циклов прямой передачи: за циклом данных следует командный цикл. *Тип цикла* задается уровнем на линии HostAck: в цикле данных — высокий, в командном цикле — низкий. В командном цикле байт может содержать *канальный адрес* или счетчик RLE. Отличительным признаком является бит 7 (старший): если он нулевой, то биты 0-6 содержат счетчик *RLE* (0-127), если единичный — то канальный адрес. На рис. 1.10, б показана пара циклов обратной передачи.

В отличие от диаграмм обмена EPP, на рис. 1.10 не приведены сигналы циклов системной шины процессора. В данном режиме обмен программы с ПУ разбивается на два относительно независимых процесса, которые связаны через FIFO-буфер. Обмен драйвера с FIFO-буфером может осуществляться как с использованием DMA, так и программного ввода-вывода. Обмен ПУ с буфером аппаратно выполняет контроллер ECP-порта. Драйвер в режиме ECP не имеет информации о точном состоянии процесса обмена, но обычно важно только то, завершен он или нет.

Прямая передача данных на внешнем интерфейсе состоит из следующих шагов:

1. Хост помещает данные на шину канала и устанавливает признак цикла данных (высокий уровень) или команды (низкий уровень) на линии HostAck.
2. Хост устанавливает низкий уровень на линии HostClk, указывая на действительность данных.
3. ПУ отвечает установкой высокого уровня на линии PeriphAck.
4. Хост устанавливает высокий уровень линии HostClk, и этот перепад может использоваться для фиксации данных в ПУ.
5. ПУ устанавливает низкий уровень на линии PeriphAck для указания на готовность к приему следующего байта.

Поскольку передача в ECP происходит через FIFO-буферы, которые могут присутствовать на обеих сторонах интерфейса, важно понимать, на каком этапе данные можно считать переданными. Данные считаются *переданными* на шаге 4, когда линия HostClk переходит в высокий уровень. В этот момент модифицируются счетчики переданных и принятых байт. В протоколе ECP есть условия, вызывающие прекращение обмена между шагами 3 и 4. Тогда эти данные не должны рассматриваться как переданные.

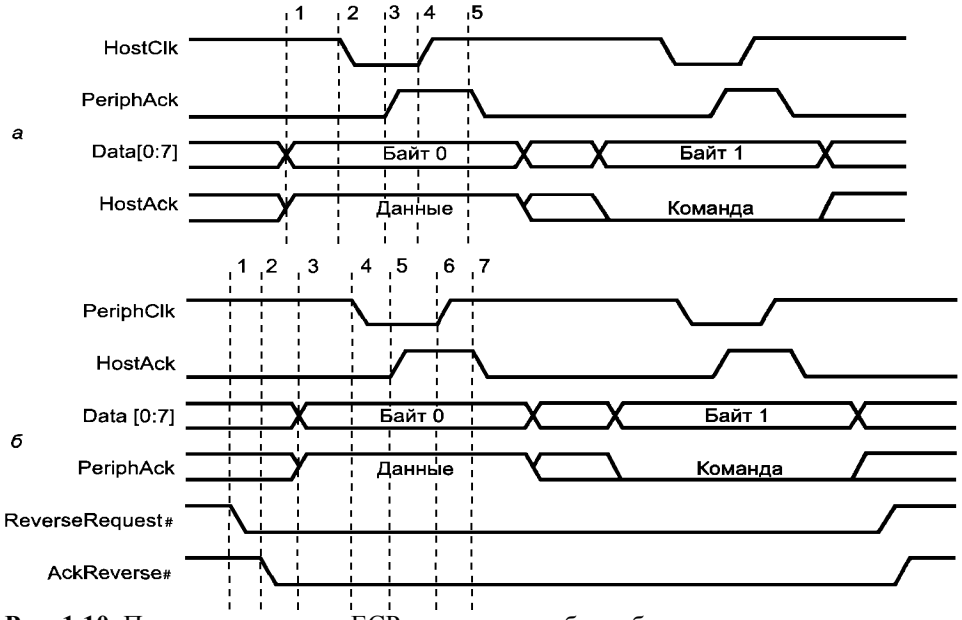


Рис. 1.10. Передача в режиме ECP: а — прямая, б — обратная

Из рис. 1.10 видно и другое отличие ECP от EPP. Протокол EPP позволяет драйверу чередовать циклы прямой и обратной передачи, не запрашивая подтверждения на смену направления. В ECP *смена направления* должна быть согласована: хост запрашивает реверс установкой ReverseRequest#, после чего он должен дождаться подтверждения сигналом AckReverse#. Поскольку предыдущий цикл мог выполняться по прямому доступу, драйвер должен дождаться завершения прямого доступа или прервать его, выгрузить буфер FIFO, определив точное значение счетчика переданных байт, и только после этого запрашивать реверс.

Обратная передача данных состоит из следующих шагов:

1. Хост запрашивает изменение направления канала, устанавливая низкий уровень на линии ReverseRequest#.
2. ПУ разрешает смену направления установкой низкого уровня на линии AckReverse#.
3. ПУ помещает данные на шину канала и устанавливает признак цикла данных (высокий уровень) или команды (низкий уровень) на линии PeriphAck.
4. ПУ устанавливает низкий уровень на линии PeriphClk, указывая на действительность данных.
5. Хост отвечает установкой высокого уровня на линии HostAck.
6. ПУ устанавливает высокий уровень линии PeriphClk; этот перепад может использоваться для фиксации данных хостом.
7. Хост устанавливает низкий уровень на линии HostAck для указания на готовность к приему следующего байта.

1.3.4.2. Режимы и регистры ECP-порта

Программный интерфейс и регистры ECP для адаптеров IEEE 1284 определяет спецификация Microsoft. Порт ECP может работать в различных режимах, приведенных в табл. 1.8, где код соответствует полю Mode регистра ECR (биты [7:5]).

Таблица 1.8. Режимы ECP-порта

Код	Режим
000	*SPP mode,* стандартный (традиционный) режим
001	*Bi-directional mode,* двунаправленный порт (тип 1 для PS/2)
010	*Fast Centronics,* однонаправленный с использованием FIFO и DMA
011	*ECP Parallel Port mode,* собственно режим *ECP*
100	*EPP Parallel Port mode,* режим EPP[1]
101	Зарезервировано
110	*Test mode,* тестирование работы FIFO и прерываний
111	*Configuration mode,* доступ к конфигурационным регистрам

[1] Этот режим не входит в спецификацию Microsoft, но трактуется как EPP многими адаптерами портов, если в CMOS Setup установлен режим ECP+EPP.

Режим 000 (SPP) - режим совместимости, в этом режиме порт работает как однонаправленный программно-управляемый SPP-порт.

Режим 001 (Bi-Di PS/2) - режим двунаправленной передачи, в этом режиме порт работает как двунаправленный порт PS/2 типа 1. От режима 000 отличается возможностью реверса канала данных по биту

CR. 5.

Режим 010 (Fast Centronics) предназначен только для высокопроизводительного вывода через FIFO-буфер с использованием DMA. Сигналы квитирования по протоколу Centronics вырабатываются аппаратно. Сигнал запроса прерывания вырабатывается по состоянию FIFO-буфера, но не по сигналу Ack# (запрос одиночного байта «не интересует» драйвер быстрого блочного вывода).

Режим 011 является собственно режимом ECP, рассмотренным выше. Поток данных и команд, передаваемых в ПУ, помещается в FIFO-буфер через регистры ECPDFIFO и ECPAFIFO соответственно. Из FIFO они выводятся с соответствующим признаком цикла (состояние линии HostAck). Принимаемый поток данных от ПУ извлекается из FIFO-буфера через регистр ECPDFIFO. Получение адреса в командном цикле от ПУ не предусматривается. Обмен с регистром ECPDFIFO может производиться и по каналу DMA. *Компрессия по методу RLE* при передаче выполняется программно. Для передачи подряд более двух одинаковых байт данных в регистр ECPAFIFO записывается байт, у которого младшие 7 бит содержат счетчик *RLC* (значение RLC= 127 соответствует 128 повторам), а старший бит нулевой. После этого в ECPDFIFO записывается сам байт. Принимая эту пару байт (командный байт и байт данных), ПУ осуществляет декомпрессию. При приеме потока от ПУ адаптер ECP декомпрессию осуществляет аппаратно и в FIFO-буфер помещает уже декомпрессированные данные. Из выше изложенного видно, что вывод данных с одновременным использованием компрессии и DMA невозможен, а ввод - возможен.

Режим 100 (EPP) — один из способов включения режима EPP (если таковой поддерживается адаптером и разрешен в CMOS Setup).

Режим 110 (Test Mode) предназначен для тестирования взаимодействия FIFO и прерываний. Данные могут передаваться в регистр TFIFO и из него с помощью DMA или программным способом. На внешний интерфейс обмен не воздействует. Адаптер отрабатывает операции вхолостую на максимальной скорости интерфейса (как будто сигналы квитирования приходят без задержек). Адаптер следит за состоянием буфера и по мере необходимости вырабатывает сигналы запроса прерывания. Таким образом программа может определить максимальную пропускную способность канала.

Режим 111 (Configuration mode) предназначен для доступа к конфигурационным регистрам. Выделение режима защищает адаптер и протокол от некорректных изменений конфигурации в процессе обмена.

Регистровая модель адаптера ECP (табл. 1.9) использует свойства архитектуры стандартной шины и адаптеров ISA, где для дешифрации адресов портов ввода-вывода задействуются только 10 младших линий шины адреса. Поэтому, например, обращения по адресам Port, Port+400h, Port+800h... будут восприниматься как обращения к адресу Port, лежащему в диапазоне 0-3FFh. Современные PC и адаптеры декодируют большее количество адресных бит, поэтому обращения по адресам 0378h и 0778h будут адресованы двум различным регистрам. Помещение дополнительных регистров ECP «за спину» регистров стандартного порта (смещение 400h-402h) преследует две цели. Во-первых, эти адреса никогда не использовались традиционными адаптерами и их драйверами, и их применение в ECP не приведет к сужению доступного адресного пространства ввода-вывода. Во-вторых, этим обеспечивается совместимость со старыми адаптерами на уровне режимов 000-001 и возможность определения факта присутствия ECP-адаптера посредством обращения к его расширенным регистрам.

Каждому режиму ECP соответствуют (и доступны) свои функциональные регистры. Переключение режимов осуществляется записью в регистр ECR. «Дежурными» режимами, включаемыми по умолчанию, являются 000 или 001. В любом из них работает полубайтный режим ввода. Из этих режимов всегда можно переключиться в любой другой, но из старших режимов (010-111) переключение возможно только в 000 или 001. Для корректной работы интерфейса перед выходом из старших режимов необходимо дождаться завершения обмена по прямому доступу и очистки FIFO-буфера.

Как уже упоминалось, каждому режиму ECP соответствуют свои функциональные регистры (табл. 1.9).

Таблица 1.9. Регистры ECP

Смещение	Имя	Альтернативное имя	R/W	Режимы ECP[1]	Название
000	DR	Data	R/W	000-001	Data Register
000	ECPAFIFO	EcpAFifo	R/W	011	ECP Address FIFO
001	SR	Dsr	R/W	Все	Status Register
002	CR	Dcr	R/W	Все	Control Register
400	SDFIFO	CFifo	R/W	010	Parallel Port Data FIFO
400	ECPDFIFO	EcpDFifo	R/W	011	ECP Data FIFO
400	TFIFO	TFifo	R/W	110	Test FIFO
400	ECPCFGA	CnfgA	R	111	Configuration Register A
401	ECPCFGB	CnfgB	R/W	111	Configuration Register B
402	ECR	Ecr	R/W	Все	Extended Control Register

[1] Регистры доступны только в данных режимах (указаны значения бит 7-5 регистра ECR).

Регистр данных DR (Data) используется для передачи данных только в программно-управляемых режимах (000 и 001).

Регистр состояния SR (Dsr) передает значение сигналов на соответствующих линиях (как в SPP).

Регистр управления CR(Dcr) имеет назначение бит, совпадающее с SPP. В режимах 010,011 запись в биты 0,1 (сигналы AutoLF# и Strobe#) игнорируется.

Регистр ECPAFIFO (EcpAFifo) служит для помещения информации командных циклов (канального адреса или счетчика RLE, в зависимости от бита 7) в FIFO-буфер. Из буфера информация будет выдана в командном цикле вывода.

Регистр SDFIFO (CFifo) используется для передачи данных в режиме 010. Данные, записанные в регистр (или посланные по каналу DMA), передаются через буфер FIFO по реализованному аппаратно протоколу Centronics. При этом должно быть задано прямое направление передачи (бит CR 5=0).

Регистр ECPDFIFO (EcpDFifo) используется для обмена данными в режиме 011 (ECP). Данные, записанные в регистр или считанные из него (или переданные по каналу DMA), передаются через буфер FIFO по протоколу ECP.

Регистр TFIFO (Tfifo) обеспечивает механизм тестирования FIFO-буфера в режиме 110.

Регистр ECPCFGA (CnfgA) позволяет считывать информацию об адаптере (идентификационный код в битах [7:4]).

Регистр ECPCFGB (CnfgB) позволят хранить любую информацию, необходимую драйверу. Запись в регистр не влияет на работу порта.

Регистр ECR (Ecr) — главный управляющий регистр ECP. Его биты имеют следующее назначение (в скобках приводится альтернативное обозначение):

- ECR [7: 5] - ECP MODE - задают режим ECP (см. табл. 1.9);
- ECR. 4 — ERRINTREN# (nErrIntEn) — (Error Interrupt Disable) запрещает прерывания по сигналу Error# (при нулевом значении бита по отрицательному перепаду на этой линии вырабатывается запрос прерывания);
- ECR. 3 — DMAEN (dmaEn) — (DMA Enable) разрешает обмен по каналу DMA;
- ECR. 2 — SERVICEINTR (serviceIntr) — (Service Interrupt) запрещает сервисные прерывания, которые вырабатываются по окончании цикла DMA (если он разрешен), по порогу заполнения/опустошения FIFO-буфера (если не используется DMA) и по ошибке переполнения буфера сверху или снизу;
- ECR. 1 — FIFOFS (full) — (FIFO Full Status) сигнализирует о заполнении буфера; при FIFOFS=1 в буфере нет ни одного свободного байта;
- ECR. 0 — FIFOES (empty) — (FIFO Empty Status) указывает на полное опустошение буфера; комбинация FIFOFS = FIFOES = 1 означает ошибку работы с FIFO (переполнение сверху или снизу).

Когда порт находится в стандартном или двунаправленном режимах (000 или 001), первые три регистра полностью совпадают с регистрами стандартного порта. Так обеспечивается совместимость драйвера со старыми адаптерами и старых драйверов с новыми адаптерами.

По интерфейсу с программой ECP-порт напоминает EPP: после установки режима (записи кода в регистр ECR) обмен данными с устройством сводится к чтению или записи в соответствующие регистры. За состоянием FIFO-буфера наблюдают либо по регистру ECR, либо по обслуживанию сервисных прерываний от порта. Весь протокол квитирования генерируется адаптером аппаратно. Обмен данными с ECP-портом (кроме явного программного) возможен и по прямому доступу к памяти (каналу DMA), что эффективно при передаче больших блоков данных.

1.3.5. Согласование режимов IEEE 1284

ПУ в стандарте IEEE 1284 обычно не требуют от контроллера реализации всех предусмотренных этим стандартом режимов. Для определения режимов и методов управления конкретным устройством стандарт предусматривает *последовательность согласования (negotiation sequence).* Последовательность построена так, что старые устройства, не поддерживающие IEEE 1284, на нее не ответят, и контроллер останется в стандартном режиме. Периферия IEEE 1284 может сообщить о своих возможностях, и контроллер установит режим, удовлетворяющий и хост, и ПУ.

Во время фазы согласования контроллер выставляет на линии данных *байт расширяемости* (extensibility byte), запрашивая подтверждение на перевод интерфейса в требуемый режим или прием идентификатора ПУ (табл. 1.10). Идентификатор передается контроллеру в запрошенном режиме (любой режим обратного канала, кроме EPP). ПУ использует сигнал Xflag (Select в терминах SPP) для подтверждения запрошенного режима обратного канала, кроме полубайтного, который поддерживается всеми устройствами IEEE 1284. Бит Extensibility Link request послужит для определения дополнительных режимов в будущих расширениях стандарта.

Последовательность согласования (рис. 1.11) состоит из следующих шагов.

1. Хост выводит байт расширяемости на линии данных.
2. Хост устанавливает высокий уровень сигнала SelectIn# и низкий — AutoFeed#, что означает начало последовательности согласования.
3. ПУ отвечает установкой низкого уровня сигнала Ack# и высокого — Error#, PaperEnd и Select. Устройство, «не понимающее» стандарта 1284, ответа не даст, и дальнейшие шаги не выполнятся.
4. Хост устанавливает низкий уровень сигнала Strobe# для записи байта расширяемости в ПУ.

5. Хост устанавливает высокий уровень сигналов Strobe# и AutoLF#.
6. ПУ отвечает установкой в низкий уровень сигналов PaperEnd и Error#, если ПУ имеет обратный канал передачи данных. Если запрошенный режим поддерживается устройством, на линии Select устанавливается высокий уровень, если не поддерживается — низкий.
7. ПУ устанавливает высокий уровень на линии Ack# для указания на завершение последовательности согласования, после чего контроллер задает требуемый режим работы.

Таблица 1.10. Биты в байте расширяемости

Бит	Описание	Допустимые комбинации бит [7:0]
7	Request Extensibility Link — зарезервирован	1000 0000
6	Запрос режима EPP	0100 0000
5	Запрос режима ECP с RLE	0011 0000
4	Запрос режима ECP без RLE	0001 0000
3	Зарезервировано	0000 1000
2	Запрос идентификатора устройства с ответом в режиме: полубайтный байтный ECP без RLE ECP с RLE	 0000 0100 0000 0101 0001 0100 0011 0100
1	Зарезервировано	0000 0010
0	Запрос полубайтного режима	0000 0001
none	Запрос байтного режима	0000 0000

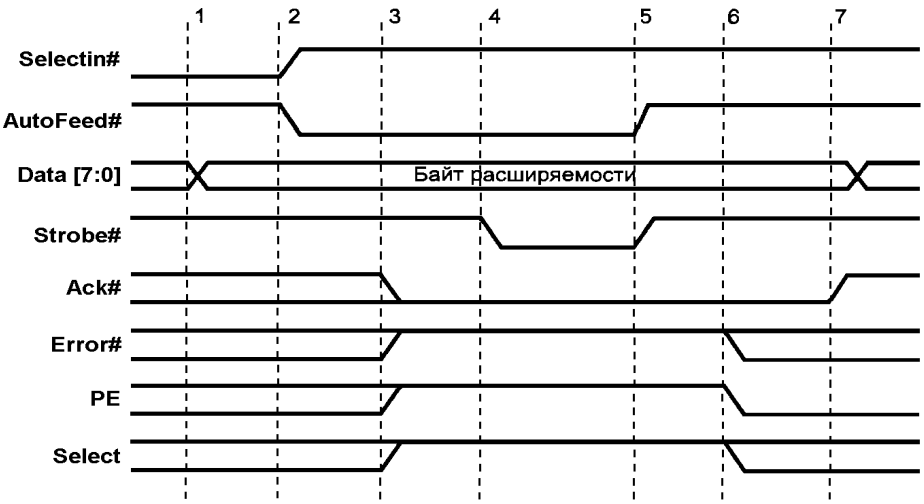


Рис. 1.11. Последовательность согласования режимов IEEE 1284

1.3.6. Физический и электрический интерфейсы

Стандарт IEEE 1284 определяет физические характеристики приемников и передатчиков сигналов, которые по уровням совместимы с ТТЛ. Спецификации стандартного порта не задавали типов выходных схем, предельных значений величин нагрузочных резисторов и емкости, вносимой цепями и проводниками. На относительно невысоких скоростях обмена разброс этих параметров не вызывал проблем совместимости. Однако расширенные (функционально и по скорости передачи) режимы требуют четких спецификаций. IEEE 1284 определяет два уровня интерфейсной совместимости. *Первый уровень* (Level I) определен для устройств медленных, но использующих смену направления передачи данных. *Второй уровень* (Level II) определен для устройств, работающих в расширенных режимах с высокими скоростями и длинными кабелями. К *передатчикам* предъявляются следующие требования.

- Уровни сигналов без нагрузки не должны выходить за пределы -0,5... +5,5 В.
- Уровни сигналов при токе нагрузки 14 мА должны быть не ниже +2,4 В для высокого уровня (V_{OH}) и не выше +0,4 В для низкого уровня (V_{OL}) на постоянном токе.
- Выходной импеданс R_O, измеренный на разъеме, должен составлять 50±5 Ом на уровне V_{OH} - V_{OL}. Для обеспечения заданного импеданса используют последовательные резисторы в выходных цепях передатчика. Согласование импеданса передатчика и кабеля снижает уровень импульсных помех.
- Скорость нарастания (спада) импульса должна находиться в пределах 0,05-0,4 В/нс.

Ниже перечислены требования к *приемникам*.

- Допустимые пиковые значения сигналов —2,0...+7,0 В.
- Пороги срабатывания должны быть не выше 2,0 В (V_{OH}) для высокого уровня и не ниже 0,8 В (V_{IL}) для низкого.
- Приемник должен иметь гистерезис в пределах 0,2-1,2 В (гистерезисом обладают специальные микросхемы — триггеры Шмитта).
- Входной ток микросхемы (втекающий и вытекающий) не должен превышать 20 мкА, входные линии соединяются с шиной питания +5 В резистором 1,2 кОм.
- Входная емкость не должна превышать 50 пФ.

Когда появилась спецификация ECP, компания Microsoft рекомендовала применение динамических терминаторов на каждую линию интерфейса. Однако в настоящее время следуют спецификации IEEE 1284, в которой динамические терминаторы не применяются. Рекомендованные схемы входных, выходных и двунаправленных цепей приведены на рис. 1.12.

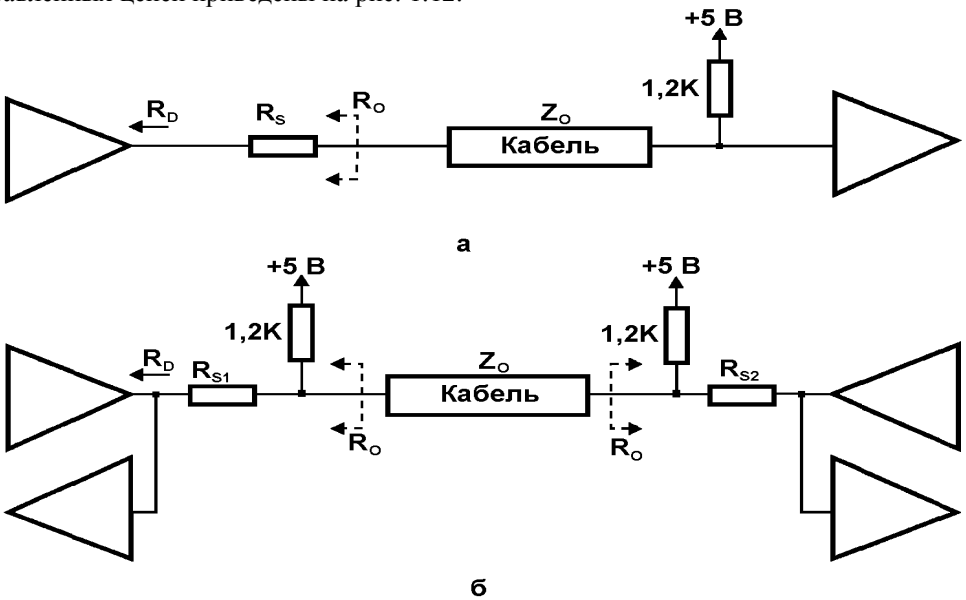


Рис. 1.12. Оконечные цепи линий интерфейса IEEE 1284: а — однонаправленные линии, б — двунаправленные

Стандарт IEEE 1284 определяет три типа используемых *разъемов.* Типы *A (DB-25)* и *B (Centronics-36)* характерны для традиционных кабелей подключения принтера, тип *C* — новый малогабаритный 36-контактный разъем.

Традиционные интерфейсные кабели имеют от 18 до 25 проводов, в зависимости от числа проводников цепи GND. Эти проводники могут быть как перевитыми, так и нет. К экранированию кабеля жестких требований не предъявлялось. Такие кабели вряд ли будут надежно работать на скорости передачи 2 Мбайт/с и при длине более 2 м.

Стандарт IEEE 1284 регламентирует *свойства кабелей.*

- Все сигнальные линии должны быть перевитыми с отдельными обратными (общими) проводами.
- Каждая пара должна иметь импеданс 62±6 Ом в частотном диапазоне 4-16 МГц.
- Уровень перекрестных помех между парами не должен превышать 10 %.
- Кабель должен иметь экран (фольгу), покрывающий не менее 85 % внешней поверхности. На концах кабеля экран должен быть окольцован и соединен с контактом разъема.

Кабели, удовлетворяющие этим требованиям, маркируются надписью «IEEE Std 1284-1994 Compliant». Они могут иметь длину до 10 метров, обозначения типов приведены в табл. 1.11.

Таблица 1.11. Типы кабелей IEEE 1284

Тип	Расшифровка	Разъем 1	Разъем 2
AMAM	Type A Male — Type A Male	A(вилка)	A(вилка)
AMAF	Type A Male — Type A Female	A(вилка)	A(розетка)
AB	Type A Male — Type B Plug — стандартный кабель к принтеру	A(вилка)	B
AC	Type A Male — Type C Plug — новый кабель к принтеру	A(вилка)	C
BC	Type B Plug — Type C Plug	B	C
CC	Type C Plug — Type C Plug	C	C

1.3.7. Развитие стандарта IEEE 1284

Ниже перечислены некоторые дополнения основного стандарта IEEE 1284.

- *IEEE P1284.1* «Standard for Information Technology for Transport Independent Printer/Scanner Interface (TIP/SI)». Этот стандарт разрабатывается для управления и обслуживания сканеров и принтеров на основе протокола NPAP (Network Printing Alliance Protocol).
- *IEEE P1284.2* «Standard for Test, Measurement and Conformance to IEEE Std. 1284» — стандарт для тестирования портов, кабелей и устройств на совместимость с IEEE 1284.
- *IEEE P1284.3* «Standard for Interface and Protocol Extensions to IEEE Std. 1284 Compliant Peripheral and Host Adapter Ports» — стандарт на драйверы и использование устройств прикладным программным обеспечением (ПО). Приняты спецификации BIOS для использования EPP драйверами DOS. Прорабатывается стандарт на разделяемое использование одного порта цепочкой устройств или группой устройств, подключаемых через мультиплексор.
- *IEEE P1284.4* «Standard for Data Delivery and Logical Channels for IEEE Std. 1284 Interfaces» направлен на реализацию пакетного протокола достоверной передачи данных через параллельный порт. Основой служит протокол MLC (Multiple Logical Channels) фирмы Hewlett-Packard, однако совместимость с ним в окончательной версии стандарта не гарантируется.

Контрольные вопросы

1. Когда появился стандарт IEEE 1284 и что он определяет?
2. Кратко охарактеризуйте режимы обмена через параллельный порт , определяемые стандартом IEEE 1284.
3. Как изначально могут задаваться режимы работы параллельного порта, предусмотренные стандартом IEEE 1284?
4. Какие понятия предусмотрены при описании режимов обмена в IEEE 1284?
5. Охарактеризуйте полубайтный режим ввода - Nibble Mode.
6. Охарактеризуйте двунаправленный байтовый режим - Byte Mode.
7. Кем и для чего был разработан порт EPP? Какие циклы обмена поддерживает EPP порт?
8. Охарактеризуйте назначение сигналов EPP порта и их связь с сигналами SPP порта.
9. Охарактеризуйте регистровую архитектуру EPP порта.
10. Охарактеризуйте цикл записи данных EPP порта и адресный цикл чтения EPP порта.
11. В чем суть связанных циклов обмена через EPP порт?
12. Что понимается под протоколом блокированного квитирования?
13. Как реализуется механизм тайм-аута в EPP порту?
14. В чем особенность контроллера EPP-1.7?
15. Охарактеризуйте особенности аппаратной организации протокола обмена EPP порта.
16. Как достигается скорость обмена 2 Мбайт/с через EPP порт?
17. В чем суть обмена в реальном времени?
18. Кем был разработан порт ECP?
19. Какие типы циклов обмена обеспечивает ECP порт?
20. Каковы особенности программной модели ECP порта?
21. Охарактеризуйте компрессию по методу RLE.
22. Что понимается под канальной адресацией ECP?
23. Охарактеризуйте переопределение сигналов ECP по отношению к сигналам SPP.
24. Каковы особенности циклов обмена ECP по сравнению с циклами EPP?
25. Охарактеризуйте шаги прямой передачи данных через ECP порт.
26. Охарактеризуйте шаги обратной передачи через ECP порт.
27. Охарактеризуйте режимы ECP порта.
28. В чем особенность регистровой модели адаптера ECP?
29. Охарактеризуйте регистры контроллера ECP порта.
30. Для чего нужно согласование режимов IEEE 1284?
31. Охарактеризуйте шаги последовательности согласования режимов.
32. Охарактеризуйте допустимые комбинации бит в байте расширяемости.
33. Охарактеризуйте первый и второй уровни интерфейсной совместимости IEEE 1284.
34. Какие требования предъявляются к передатчикам стандарта IEEE 1284?
35. Какие требования предъявляются к приемникам стандарта IEEE 1284?
36. Охарактеризуйте схемы входных, выходных и двунаправленных цепей IEEE 1284.
37. Какие типы разъемов определяет стандарт IEEE 1284?
38. Какие свойства кабелей и какие типы кабелей регламентирует стандарт IEEE 1284?
39. Охарактеризуйте дополнения е стандарту IEEE 1284.

В данном разделе использованы материалы из [2 - 9].

1.4. Системная поддержка LPT-порта

Системная поддержка BIOS LPT-порта включает идентификацию (поиск) установленных портов на стадии POST (Power On Self Test - тест начального включения), сервисы печати SPP-порта (Int 17h - драйвер принтера и Int 05h - печать копии экрана) и сервис расширенных режимов порта - EPP BIOS.

1.4.1. Идентификация LPT-портов на стадии POST.

В процессе начального тестирования POST BIOS проверяет наличие параллельных портов по адресам 3BCh, 378h и 278h и помещает базовые адреса обнаруженных портов в ячейки области данных BIOS (BIOS Data Area) 0:0408h, 0:040Ah, 0:040Ch. Эти ячейки хранят адреса портов LPT1-LPT3, нулевое значение адреса является признаком отсутствия порта с данным номером.. BIOS поддерживает также и LPT4, однако на стадии POST он не ищется. Поэтому, если пользователь желает работать с этим портом он должен сам занести его адрес в ячейку памяти 0:040Eh. BIOS предусматривает также функции тайм-аута портов, длительность которых может программироваться пользователем. Для этого используются адреса 0:0478h, 0:0479h, 0:047Ah и 0:047Bh. Значения тайм-аута устанавливаются POST на 20 и могут быть изменены в диапазоне от 1 до 255. Каждая единица соответствует примерно одной секунде. В BIOS Data Area резервируется также ячейка 0:0500h в которую помещается информация о состоянии функции "Печать копии экрана" (прерывание Int 05h).

Поиск портов обычно ведется достаточно примитивно — по базовому адресу (в регистр данных предполагаемого порта) выводится тестовый байт (AAh или 55h), затем производится ввод по тому же адресу. Если считанный байт совпал с записанным, предполагается, что найден LPT-порт; его адрес помещается в ячейку BIOS Data Area. Базовые адреса портов могут быть впоследствии изменены программно. Адрес порта LPT4 система BIOS самостоятельно установить не может, поскольку в списке стандартных адресов поиска имеются только три вышеуказанных.

Обнаруженные порты *инициализируются:* записью в регистр управления формируется и снимается сигнал Init#, после чего записывается значение 0Ch, соответствующее исходному состоянию сигналов интерфейса. В некоторых случаях сигнал Init# активен с момента аппаратного сброса до инициализации порта при загрузке ОС. Это можно заметить по поведению включенного принтера во время перезагрузки компьютера — у принтера надолго гаснет индикатор On-Line. Следствие этого явления — невозможность распечатки экранов (например, параметров BIOS Setup) по нажатию клавиши Print Screen до загрузки ОС.

1.4.2. Печать копии экрана (INT 05h)

Возможны два случая:

- INT 05h - SW (программой), печать копии экрана,
- INT 05h - CPU, (центральным процессором) нарушение адресной границы.

INT 05h вызывается путем выполнения команды INT 05h (для печати копии экрана). Это достигается нажатием клавиши PrintScreen. Данное прерывание используется также CPU для генерирования неисправности, когда превышены пределы, указанные в команде BOUND.

При нажатии клавиши PrintScreen обработчик клавиатуры (драйвер клавиатуры в ПЗУ BIOS) обращается к программе прерываний. INT 05h, которая использует команду INT10h, чтобы считывать информацию из видеобуфера, и INT 17h для передачи символов на принтер. Символы из каждой позиции экрана посылаются на принтер без подавления следов перемещения курсора. При обращении к INT 05h сохраняется текущая позиция курсора. Эта позиция восстанавливается по завершении печати. Состояние печати копии экрана хранится по адресу 0040:0100h, где 00h - печати копии экрана нет (или успешное завершение вызова печати копии экрана), 01h - идет печать копии экрана. Значение FFh возвращается при обнаружении тайм-аута. Если во время печати копии экрана производится попытка еще одного обращения по команде INT 05h, оно игнорируется. Все регистры сохраняются. INT 05h выполняется при разрешенных прерываниях.

Прикладная программа может использовать данное прерывание для запроса документальной копии экрана. Этот вектор обычно изменяется при расширении функций экрана, которые оперируют с графическим экраном, специальными принтерами или переадресовывают выходные данные в файлы.

1.4.3. Драйвер принтера INT 17h

Для работы через параллельный SPP-порт с принтером предназначена группа функций BIOS, вызываемых по прерыванию Int 17h. После выполнения любой из функций данной группы в регистре AH будет возвращен код состояния принтера, формат которого показан на рис. 1.13.

Биты кода состояния имеют следующее значение:

- бит 0 — признак тайм-аута (0 — нормальное состояние, 1 — ошибка тайм-аута, то есть принтер не отвечает);
- биты 1 и 2 — не используются, установлены в 0;
- бит 3 — признак ошибки ввода-вывода (0 — ошибка, 1 — нет ошибки);
- бит 4 — признак выбора принтера (0 — принтер в автономном режиме, 1 — принтер в режиме

подключения);

- бит 5 — контроль наличия бумаги (0 — бумага вставлена, 1 — нет бумаги);
- бит 6 — подтверждение приема (0 — подтверждение приема символа, 1 — обычное состояние);
- бит 7 — признак занятости принтера (0 — принтер занят, 1 — принтер свободен).

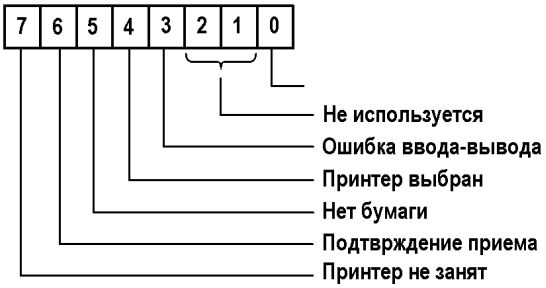


Рис. 1.13. Формат байта кода состояния

1.4.3.1. Прерывание Int 17h, функция 00h: вывести символ на принтер

Функция предназначена для выполнения побайтного вывода информации. Это основная функция данной группы — она обеспечивает посылку команд и данных на принтер. Время выполнения функции примерно 15 мкс.

Перед вызовом прерывания требуется записать в регистры следующую информацию:

- в AH — значение 00h;
- в AL — код выводимого символа;
- в DX - номер порта принтера (0 - LPT1, 1 - LPT2, 2 - LPT3, 3 - LPT4).

1.4.3.2. Прерывание Int 17h, функция 01h: инициализировать порт

Функция предназначена для выполнения инициализации (сброса) интерфейса принтера. Вызывают данную функцию после обнаружения серьезных сбоев в работе принтера. Функция устанавливает устанавлиавет линию Init в состояние высокого уровня на время около 20 мкс. Время выполнения функции примерно 70 мс.

Перед вызовом прерывания требуется записать в регистры следующую информацию:

- в AH — значение 01h;
- в DX — номер порта принтера (0 - LPT1, 1 - LPT2, 2 - LPT3, 3 - LPT4).

1.4.3.3. Прерывание Int 17h, функция 02h: получить состояние принтера

Функция возвращает текущее состояние принтера. Обычно ее вызывают перед началом печати очередной строки или нового листа, чтобы проверить готовность принтера к печати: включен ли принтер и заправлена ли в него бумага. Время выполнения функции примерно 5 мкс.

Перед вызовом прерывания требуется записать в регистры следующую информацию:

- в AH — значение 02h;
- в DX — номер порта принтера (0 - LPT1, 1 - LPT2, 2 - LPT3, 3 - LPT4).

1.4.4. Функции EPP BIOS

Дополнительный набор функций BIOS, предназначенный для обслуживания новых режимов работы параллельного порта, получил наименование EPP BIOS. При работе с функциями EPP BIOS частично используется прерывание Int17h, т. е то же прерывание, что и при работе со стандартным параллельным портом (драйвер принтера). Рассмотрим дополнительный набор функций BIOS, предназначенный для обслуживания новых режимов работы параллельного порта и получивший наименование EPP BIOS.

1.4.4.1.Прерывание Int 17h, функция 02h: проверить наличие EPP BIOS

Функция проверяет наличие EPP BIOS. В случае если EPP BIOS поддерживается системой, функция возвращает вектор (“точку входа”) для вызова функций EPP BIOS.

Перед вызовом прерывания требуется записать в регистры следующую информацию:

- в AH — значение 02h;
- в DX — номер параллельного порта (0 - LPT1, 1 - LPT2, 2 - LPT3);
- в AL — значение 0;
- в CH — значение 45h (символ E);
- в BL — значение 50h (символ P);
- в BH — значение 50h (символ P).

В случае успешного завершения операции функция возвращает:

- в AH — значение 0;

- в AL — значение 45h;
- в CX — значение 5050h;
- в паре регистров DX: BX — вектор EPP (точку входа в EPP BIOS).

Для вызова всех остальных функций EPP BIOS применяется вектор точки входа EPP BIOS, возвращаемый данной функцией.

ПРИМЕЧАНИЕ: Кроме регистров, используемых для передачи параметров при вызове функций EPP и для возврата результатов, в функциях используется также регистр BX, содержимое которого не сохраняется (теряется после вызова функции).

1.4.4.2. Переход по вектору EPP, функция 00h: определить конфигурацию и возможности порта

Функция Query Config позволяет определить текущую конфигурацию параллельного порта с заданным номером.

Перед вызовом данной функции по вектору EPP требуется записать в регистры следующую информацию:

- в AH — значение 00h;
- в DL — номер параллельного порта (0 - LPT1,1 - LPT2,2 - LPT3).

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — уровень прерывания от EPP порта (может принимать значения в диапазоне от 0 до 15; код FFh означает, что прерывания портом не поддерживаются);
- в BH — номер версии EPP BIOS;
- в BL — возможности параллельного порта (бит 0 — признак наличия мультиплексора; бит 1 — признак наличия поддержки двунаправленного режима PS/2; бит 2 — признак наличия поддержки режима EPP 1.9; бит 3 — признак наличия поддержки режима ECP; бит 4 — зарезервирован; бит 5 — признак наличия поддержки Centronics FIFO; бит 6 — признак наличия поддержки режима EPP 1.7; бит 7 — зарезервирован);
- в CX — базовый адрес группы регистров порта при работе в режиме SPP;
- в паре регистров ES:DI — указатель на ограниченную нулем текстовую строку, содержащую информацию о разработчике данной версии EPP BIOS.

1.4.4.3. Переход по вектору EPP, функция 01h: установить режим работы порта

Функция Set Mode позволяет установить режим работы параллельного порта с заданным номером. Вызов данной функции разрешен только в том случае, если процессор работает в «реальном» режиме.

Перед вызовом данной функции требуется записать в регистры следующую информацию:

- в AH — значение 01h;
- в DL — номер параллельного порта;
- в AL — код устанавливаемого режима (бит 0 — установить «режим совместимости» SPP, бит 1 — установить двунаправленный режим SPP, бит 2 — установить режим EPP, бит 3 — установить режим ECP, бит 4 — зарезервирован, бит 5 — установить режим Centronics FIFO, бит 6 — установить режим EPP 1.7, бит 7 — зарезервирован).

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.4. Переход по вектору EPP, функция 02h: определить режим работы порта

Функция Get Mode позволяет определить текущий режим работы параллельного порта с заданным номером. Вызов данной функции разрешен только в том случае, если процессор работает в «реальном» режиме.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 02h;
- в DL — номер параллельного порта.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — код режима работы порта (бит 0 — признак режима совместимости, бит 1 — признак двунаправленного режима, бит 2 — признак режима EPP, бит 3 — признак режима ECP, бит 4 — зарезервирован, бит 5 — признак режима Centronics FIFO, бит 6 — признак режима EPP 1.7, бит 7 — признак того, что разрешена обработка прерываний в режиме EPP).

1.4.4.5. Переход по вектору EPP, функция 03h: управление прерываниями

Функция Interrupt Control позволяет включать и отключать обработку прерывания, связанного с заданным параллельным портом. Вызов данной функции разрешен только в том случае, если процессор работает в «реальном» режиме.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 03h;
- в DL — номер параллельного порта;
- в AH — управляющий код (0 — запретить прерывания от порта EPP, 1 — разрешить прерывания).

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.6. Переход по вектору EPP, функция 04h: инициализация

Функция EPP Reset позволяет осуществить инициализацию («сброс») 1 устройства, подключенного к заданному параллельному порту.

Перед вызовом данной функции требуется записать в регистры следующую информацию:

- в AH — значение 04h;
- в DL — номер параллельного порта.

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.7. Переход по вектору EPP, функция 05h: запись адреса

Функция Address Write выполняет цикл записи адреса устройства.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 05h;
- в DL — номер параллельного порта;
- в AL — адрес устройства.

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.8. Переход по вектору EPP, функция 06h: считывание адреса

Функция Address Read выполняет цикл считывания адреса активного устройства.

Перед вызовом данной функции требуется записать в регистры следующую информацию:

- в AH — значение 06h;
- в DL — номер параллельного порта.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — адрес устройства и дополнительные данные.

1.4.4.9. Переход по вектору EPP, функция 07h: запись байта

Функция Write Byte выполняет вывод одного байта данных через порт данных EPP.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 07h;
- в DL — номер параллельного порта;
- в AL — байт данных.

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.10. Переход по вектору EPP, функция 08h: запись блока данных

Функция Write Block выполняет вывод блока данных из заданного буфера через порт данных EPP.

Перед вызовом функции требуется поместить в регистры следующую информацию:

- в AH — значение 08h;
- в DL — номер параллельного порта;
- в CX — размер передаваемого блока в байтах (значение 0 в данном регистре соответствует размеру блока, равному 64 Кбайт);
- в пару регистров ES:SI — указатель на область памяти (буфер), содержащую передаваемый блок данных.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в CX — значение 0 в случае успешного выполнения передачи блока или количество не переданных байт в случае возникновения ошибки (сбоя) в процессе передачи.

ПРИМЕЧАНИЕ: Задавать значение 0 в регистре CX при вызове данной функции не рекомендуется: могут возникать проблемы совместимости, связанные с различными реализациями EPP BIOS.

1.4.4.11. Переход по вектору EPP, функция 09h: считывание байта данных

Функция Read Byte выполняет считывание одного байта данных из порта данных EPP.
Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 09h;
- в DL — номер параллельного порта.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — принятый байт данных.

1.4.4.12. Переход по вектору EPP, функция 90Ah: считывание блока данных

Функция Read Block выполняет считывание блока данных в заданный буфер через порт данных EPP.
Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 0Ah;
- в DL — номер параллельного порта;
- в CX — размер принимаемого блока в байтах (значение 0 в данном регистре соответствует размеру блока, равному 64 Кбайт);
- в пару регистров ES:DI — указатель на область памяти, предназначенную для размещения принимаемого блока данных.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в CX — значение 0 в случае успешного завершения операции или количество не переданных байт в случае возникновения ошибки.

ПРИМЕЧАНИЕ: Задавать значение 0 в регистре CX при вызове данной функции не рекомендуется: могут возникать проблемы совместимости, связанные с различными реализациями EPP BIOS.

1.4.4.13. Переход по вектору EPP, функция 0Bh: запись адреса и считывание байта

Функция Address/Byte Read выполняет комбинированную операцию: устанавливает адрес устройства, а затем принимает от него байт данных. Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 0Bh;
- в DL — номер параллельного порта;
- в AL — адрес устройства.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — принятый байт данных.

1.4.4.14. Переход по вектору EPP, функция 0Ch: запись адреса и байта данных

Функция Address/Byte Write выполняет комбинированную операцию: устанавливает адрес устройства, а затем передает этому устройству байт данных.
Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 0Ch;
- в DL — номер параллельного порта;
- в AL — адрес устройства;
- в DH — передаваемый байт данных.

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.15. Переход по вектору EPP, функция 0Dh: запись адреса и считывание блока данных

Функция Address/Block Read выполняет комбинированную операцию: устанавливает адрес устройства, а затем принимает от него блок данных.
Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 0Dh;
- в DL — номер параллельного порта;
- в AL — адрес устройства;

- в CX — размер принимаемого блока в байтах (значение 0 в данном регистре соответствует размеру блока, равному 64 Кбайт);
- в паре регистров ES:DI — указатель на область памяти, предназначенную для размещения принимаемого блока данных

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в CX — значение 0 в случае успешного завершения операции; количество не переданных байт в случае возникновения ошибки.

ПРИМЕЧАНИЕ: Задавать значение 0 в регистре CX при вызове данной функции не рекомендуется: могут возникать проблемы совместимости, связанные с различными реализациями EPP BIOS.

1.4.4.16. Переход по вектору EPP, функция 0Eh: запись адреса и блока данных

Функция Address/Block Write выполняет комбинированную операцию: устанавливает адрес устройства, а затем передает ему блок данных.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 0Eh;
- в DL — номер параллельного порта;
- в AL — адрес устройства;
- в CX — размер передаваемого блока в байтах (значение 0 в данном регистре соответствует размеру блока, равному 64 Кбайт);
- в пару регистров ES:SI — указатель на область памяти, содержащую передаваемый блок данных.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в CX — значение 0 в случае успешного завершения операции или количество не переданных байт в случае возникновения ошибки.

ПРИМЕЧАНИЕ: Задавать значение 0 в регистре CX при вызове данной функции не рекомендуется: могут возникать проблемы совместимости, связанные с различными реализациями EPP BIOS.

1.4.4.17. Переход по вектору EPP, функция 0Fh: захватить порт

Функция Lock Port позволяет упорядочить ввод и вывод данных через EPP - порт. Данная функция применяется для выбора порта конкретного устройства, если это устройство входит в состав цепочке устройств или подключено через мультиплексор.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 0Fh;
- в DL — номер параллельного порта;
- в BL — адрес устройства (биты 0-3 задают номер порта .мультиплексора, биты 4-7 — номер устройства в цепочке).

После выполнения функция возвращает в регистре AH код ошибки.

Номер порта мультиплексора может принимать значения в диапазоне от 1 до 8, номер устройства в цепочке — также от 1 до 8. При использовании в системе драйвера для работы с мультиплексором или цепочкой устройств вызов функции Lock Port завершается успешно только в том случае, если порт еще не захвачен.

Захват порта должен быть выполнен перед началом работы с устройством. Пока порт не захвачен, допускается выполнение следующих операций:

Device Interrupt	Query Config
Installation Check	Query Daisy Chain
Real time Mode	Query Device Port
Rescan Daisy Chain	Query Mux
Set Product ID	

1.4.4.18. Переход по вектору EPP, функция 10h: освободить порт

Функция Unlock Port освобождает EPP - порт и позволяет его использовать драйверам других устройств.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 10h;
- в DL — номер параллельного порта;
- в BL — адрес устройства (биты 0-3 задают номер порта мультиплексора, биты 4-7 — номер устройства в цепочке).

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.19. Переход по вектору EPP, функция 11h: установить обработчик прерываний

Функция Device Interrupt позволяет драйверу EPP-устройства установить собственный обработчик прерывания для параллельного порта с заданным номером. Вызов данной функции разрешен только в том случае, если процессор работает в “реальном” режиме.

Перед вызовом функции требуется поместить в регистры следующую информацию:

- в AH — значение 11h;
- в DL — номер параллельного порта;
- в AL — код выполняемой операции (0 — запретить обработку прерываний, 1 — разрешить обработку прерываний, 2 — удалить обработчик прерываний);
- в пару регистров ES:DI — дальний указатель на обработчик прерываний;
- в BL — адрес устройства (биты 0-3 задают номер порта мультиплексора, биты 4-7 — номер устройства в цепочке).

После выполнения функция возвращает в регистре AH код ошибки.

Перед вызовом обработчика запрещаются прерывания. Обработчик не должен выполнять захват устройства, так как эта операция уже осуществлена драйвером. При выходе из обработчика нужно отправить инструкцию EOI контроллеру прерываний. Выход должен осуществляться при помощи инструкции IRET.

1.4.4.20. Переход по вектору EPP, функция 12h: режим реального времени

Функция Real Time Mode настраивает драйвер для работы в режиме реального времени. С помощью данной функции можно определить наличие устройств, требующих использования режима реального времени: если таких устройств нет, драйвер может передавать данные большими блоками; в противном случае должны использоваться маленькие блоки.

Перед вызовом функции требуется поместить в регистры следующую информацию:

- в AH — значение 12h;
- в AL — код выполняемой операции (0 — проверить наличие устройств, работающих в режиме реального времени; 1 — проинформировать драйвер о наличии устройства реального времени; 2 — сбросить флаг реального времени).

После выполнения функция возвращает в регистре AH код ошибки. При выполнении поиска устройств, работающих в режиме реального времени (код операции 0) в регистре AL будет возращен результат поиска (0 — устройства реального времени не обнаружены, 1 — найдено одно или несколько устройств).

1.4.4.21. Переход по вектору EPP, функция 40h: опросить мультиплексор

Функция Query Mux позволяет получить информацию о мультиплексоре, подключенном к указанному порту.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 40h;
- в DL — номер параллельного порта.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — флаги состояния (бит 0 — захват канала, бит 1 — наличие обработчика прерываний);
- в BL — номер выбранного (активного) порта мультиплексора;
- в BH — номер версии драйвера мультиплексора;
- в паре регистров ES:DI — указатель на ASCIIZ-строку, идентифицирующую разработчика драйвера.

1.4.4.22. Переход по вектору EPP, функция 41h: опросить устройство

Функция Query Device Port позволяет получить информацию об устройстве с заданным адресом, подключенном к указанному порту.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 41h;
- в DL — номер параллельного порта;
- в BL — адрес устройства (биты 0-3 задают номер порта мультиплексора, биты 4-7 — номер устройства в цепочке).

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;

- в AL — флаги состояния (бит 0 — выбор порта, бит 1 — захват порта, бит 2 — прерывания разрешены, бит 3 — наличие обработчика прерываний);
- в CX — идентификатор устройства (ноль, если устройство не определено).

1.4.4.23. Переход по вектору EPP, функция 42h: задать идентификатор устройства

Функция Set Product ID позволяет задать идентификатор для устройства, которое не поддерживает циклы чтения адреса или не способно выдать собственный идентификатор.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 42h;
- в DL — номер параллельного порта;
- в BL — адрес устройства (биты 0-3 задают номер порта мультиплексора, биты 4-7 — номер устройства в цепочке);
- в CX — идентификатор для устройства.

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.24. Переход по вектору EPP, функция 50h: повторное сканирование цепочки устройств

Функция Rescan Daisy Chain Product используется для динамического перераспределения номеров портов между устройствами, соединенными в цепочку.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 50h;
- в DL — номер параллельного порта;
- в BL — номер порта мультиплексора (допускается использование номеров от 1 до 8; ноль означает отсутствие мультиплексора).

После выполнения функция возвращает в регистре AH код ошибки.

1.4.4.25. Переход по вектору EPP, функция 51h: задать идентификатор устройства

Функция Query Daisy Chain позволяет получить информацию о цепочке устройств, подключенной к указанному порту.

Перед вызовом функции требуется записать в регистры следующую информацию:

- в AH — значение 51h;
- в DL — номер параллельного порта.

После выполнения функции в регистрах находится следующая информация:

- в AH — код ошибки;
- в AL — флаги состояния (бит 0 - признак захвата канала, 1 - признак наличия обработчика прерываний);
- в BL — номер выбранного устройства;
- в CL — количество устройств в цепочке (0 - нет цепочки);
- в паре регистров ES:DI - указатель на ASCIIZ-строку, идентифицирующую разработчика драйвера.

1.4.4.26. Коды ошибок EPP BIOS

Значение кода ошибки, возвращаемого функциями EPP BIOS в регистре AH, расшифровывается следующим образом:

00h — успешное завершение операции, ошибок нет;
01h — тайм-аут;
02h — команда или операция не поддерживаются EPP BIOS;
03h — некорректный адрес порта устройства;
04h — EPP BIOS занята (BIOS не является реентерабельным);
05h — некорректный параметр;
10h — мультиплексор уже заблокирован (захвачен);
20h — мультиплексор отсутствует;
40h — программа обслуживания (менеджер) цепочки или мультиплексора не установлена;
41h — порт устройства заблокирован (захвачен);
42h — порт устройства не был заблокирован;
43h — ошибка блокировки (некорректный адрес порта).

1.5. Параллельный порт и PnP

Большинство современных периферийных устройств, подключаемых к LPT-порту, поддерживает стандарт 1284 и функции PnP. Для поддержки этих функций компьютером с аппаратной точки зрения

достаточно иметь контроллер интерфейса, соответствующий стандарту 1284. Если подключаемое устройство поддерживает PnP, оно по протоколу согласования режимов 1284 способно «договориться» с портом, представляющим «интересы» компьютера, о возможных режимах обмена. Далее, для работы PnP подключенное устройство должно сообщить операционной системе все необходимые сведения о себе. Как минимум это идентификаторы производителя, модели и набор поддерживаемых команд. Более развернутая информация об устройстве может содержать идентификатор класса, подробное описание и идентификатор устройства, с которым обеспечивается совместимость. В соответствии с принятой информацией для поддержки данного устройства операционная система может предпринять действия по установке требуемого программного обеспечения.

Устройства с поддержкой PnP распознаются ОС на этапе ее загрузки, если, конечно же, они подключены к порту интерфейсным кабелем и у них включено питание. Если ОС Windows обнаруживает подключенное устройство PnP, отличающееся от того, что прописано в ее реестре для данного порта (или просто новое устройство), она пытается установить требуемые для устройства драйверы из дистрибутива ОС или из комплекта поставки нового устройства. Если Windows не желает замечать вновь подключенного устройства PnP, это может свидетельствовать о неисправности порта или кабеля. Система PnP не работает, если устройство подключается дешевым «не двунаправленным» кабелем, у которого отсутствует связь по линии SelectIn# (контакт 17 порта LPT и контакт 36 разъема Centronics).

1.6. Применение LPT-порта

Обычно LPT-порт используют для подключения принтера, однако этим его применение не исчерпывается.

Для связи двух компьютеров по параллельному интерфейсу применяются различные кабели в зависимости от режимов используемых портов. Самый простой и медленный — полубайтный режим, работающий на *всех* портах. Для этого режима в кабеле достаточно иметь 10 сигнальных и один общий провод. Распайка разъемов кабеля приведена в табл. 1.12. Связь двух PC данным кабелем поддерживается стандартным ПО типа Interlink из MS-DOS или Norton Commander. Заметим, что здесь применяется свой протокол, отличный от рассмотренного в п. 1.3.1.

Таблица 1.12. Кабель связи PC-PC (4-битный)

X1, разъем PC#1		X2, разъем PC#2	
Бит	Контакт	Контакт	Бит
DR.0	2	15	SR.3
DR.1	3	13	SR.4
DR.2	4	12	SR.5
DR.3	5	10	SR.6
DR.4	6	11	SR.7
SR.6	10	5	DR.3
SR.7	11	6	DR.4
SR.5	12	4	DR.2
SR.4	13	3	DR.1
SR.3	15	2	DR.0
GND	18-25	18-25	GND

Разъемы XI и X2 - DB25-P (вилки).

Высокоскоростная связь двух компьютеров может выполняться и в режиме ECP (режим EPP неудобен, поскольку требует синхронизации шинных циклов ввода-вывода двух компьютеров).

В табл. 1.13 приведена разводка кабеля. Из всех сигналов в кабеле не используется лишь PeriphRequest#

Таблица 1.13. Кабель связи PC-PC в режиме ECP и байтном режиме

Разъем X1		Разъем X2	
Контакт	Имя в ECP	Имя в ECP	Контакт
1	HostClk	PeriphClk	10
14	HostAck	PeriphAck	11
17	1284Active	Xflag	13
16	ReverseRequest#	AckReverse#	12
10	PeriphClk	HostClk	1
11	PeriphAck	HostAck	14
12	AckReverse#	ReverseRequest#	16
13	Xflag	1284Active	17
2,3...9	Data [0:7]	Data [0:7]	2,3...9

(контакт 15). В цепи линий данных рекомендуется вставить последовательные резисторы (0,5-1 кОм),

препятствующие протеканию слишком больших токов, когда порты данных обоих компьютеров находятся в режиме вывода. Эта ситуация возникает, когда коммуникационное ПО компьютеров еще не запущено. Связь в режиме ECP поддерживается Windows 9x, в комплект поставки этих ОС входит драйвер PARALINK.VxD, но из-за внутренней ошибки он неработоспособен. «Заплатку» на этот драйвер, а также тестовую утилиту и необходимые описания можно найти в сети (www.lpt.com, www.lvr.com/parport.htm).

Подключение *сканера к* LPT-порту эффективно, только если порт обеспечивает хотя бы двунаправленный режим *(Bi-Di),* поскольку основной поток — ввод. Лучше использовать порт ECP, если этот режим поддерживается сканером (или EPP, что маловероятно).

Подключение *внешних накопителей* (Iomega Zip Drive, CD-ROM и др.), *адаптеров ЛВС* и других симметричных устройств ввода-вывода имеет свою специфику. В режиме SPP наряду с замедлением работы устройства заметна принципиальная асимметрия этого режима: *чтение данных* происходит *в два раза медленнее, чем* (весьма небыстрая) *запись.* Применение *двунаправленного* режима *(Bi-Di* или *PS/2 Type 1)* устранит эту асимметрию — *скорости сравняются.* Только перейдя на EPP или ECP, можно получить *нормальную* скорость работы. В режиме EPP или ECP подключение к LPT-порту почти не уступает по скорости подключению через ISA-контроллер. Это справедливо и при подключении устройств со стандартным интерфейсом шин к LPT-портам через преобразователи интерфейсов (например, LPT - IDE, LPT - SCSI, LPT - PCMCIA). Заметим, что винчестер IDE, подключенный через адаптер к LPT-порту, для системы может быть представлен как устройство SCSI (это логичнее с программной точки зрения).

В табл. 1.14 описано назначение выводов разъема LPT-порта в различных режимах и их соответствие битам регистров стандартного порта.

Таблица 1.14. Назначение выводов разъема LPT-порта и бит регистров в режимах SPP, ECP и EPP

Контакт	I/O	Бит[1]	SPP	Bi-Di	ECP	EPP
1	O	CR.0\	Strobe#	HostClk	HostClk	Write#
2	I/O	DR.0	Data0	Data0	Data 0	Data0
3	I/O	DR.1	Data 1	Data 1	Data 1	Data1
4	I/O	DR.2	Data 2	Data 2	Data 2	Data 2
5	I/O	DR.3	Data 3	Data 3	Data 3	Data 3
6	I/O	DR.4	Data 4	Data 4	Data 4	Data 4
7	I/O	DR.5	Data 5	Data 5	Data 5	Data 5
8	I/O	DR.6	Data 6	Data 6	Data 6	Data 6
9	I/O	DR.7	Data 7	Data 7	Data 7	Data 7
10	I	SR.6	Ack#	PerClk	PeriphClk	INTR#
11	I	SR.7\	Busy	PerBusy	PeriphAck	Wait#
12	I	SR.5	PaperEnd	AckDataReq[1]	AckReverse#	-[2]
13	I	SR.4	Select	Xflag[1]	Xflag	-[2]
14	O	CR.1\	Auto LF#	HostBusy	HostAck	DataStb#
15	I	SR.3	Error#	DataAvail#[1]	PeriphRequest#	-[2]
16	O	CR.2	Init	Init	ReverseRequest#	Reset#
17	O	CR.3\	Select In#	1284Active	1284Active	AddrStb#

[1] Символом « \ » отмечены инвертированные сигналы (1 в регистре соответствует низкому уровню линии).

[2] Определяется пользователем.

1.7. Конфигурирование LPT-портов

Управление параллельным портом разделяется на два этапа — *предварительное конфигурирование* (Setup) аппаратных средств порта и *текущее* (оперативное) *переключение* режимов работы прикладным или системным ПО. Оперативное переключение возможно только в пределах режимов, разрешенных при конфигурировании. Этим обеспечивается возможность согласования аппаратуры с ПО и блокирования ложных переключений, вызванных некорректными действиями программы.

Конфигурирование LPT-порта зависит от его исполнения. Порт, расположенный на плате расширения (мультикарте), устанавливаемой в слот ISA или ISA+VLB, конфигурируется джамперами (или переключателями) на самой плате. Порт на системной плате конфигурируется через BIOS Setup.

Ниже перечислены параметры, подлежащие конфигурированию.

- *Базовый адрес* — 3BCh, 378h или 278h. При инициализации BIOS проверяет наличие портов по адресам именно в этом порядке и, соответственно, присваивает обнаруженным портам логические имена LPT1, LPT2, LPT3. Адрес 3BCh имеет адаптер порта, расположенный на плате MDA или HGC. Большинство портов по умолчанию конфигурируется на адрес 378h и может переключаться на 278h.
- Используемая *линия запроса прерывания:* для LPT1 — IRQ7, для LPT2 — IRQ5. Традиционно прерывания от принтера не задействуются, и этот дефицитный ресурс можно сэкономить. Однако при использование скоростных режимов ECP (или Fast Centronics) работа через прерывания может заметно

повысить производительность и снизить загрузку процессора.

- Использование *канала DMA* для режимов ECP и Fast Centronics — разрешение и номер канала DMA.
- Режимы работы порта:
  - *SPP* — порт работает только в стандартном однонаправленном программно-управляемом режиме;
  - *PS/2,* он же *bi-directional* — отличается от SPP возможностью реверса канала (установкой CR. 5=1);
  - *Fast Centronics* — аппаратное формирование протокола Centronics с использованием FIFO-буфера и, возможно, DMA;
  - *EPP* — в зависимости от использования регистров порт работает в режиме SPP или EPP;
  - *ECP*— по умолчанию включается в режим SPP или PS/2, записью в ECR может переводиться в любой режим ECP, но перевод в EPP записью в ECR кода 100 не гарантируется;
  - *ECP+EPP*— то же, что и ECP, но запись в ECR кода режима 100 переводит порт в EPP.

Выбор режима EPP, ECP или Fast Centronics сам по себе не приводит к повышению быстродействия обмена с подключенными ПУ, а только дает возможность драйверу и ПУ установить оптимальный режим в пределах их «разумения». Большинство современных драйверов и приложений пытаются использовать эффективные режимы, поэтому «подрезать им крылья» установкой простых режимов без веских на то оснований не стоит.

Принтеры и *сканеры* могут пожелать режима ECP. Windows (3.x, 9x и NT) имеет системные драйверы для этого режима. В среде DOS печать через ECP поддерживается только специальным загружаемым драйвером.

Сетевые адаптеры, внешние диски и *CD-ROM,* подключаемые к параллельному порту, могут использовать режим EPP. Для этого режима специальный драйвер пока еще не применяется; поддержка EPP включается в драйвер самого подключаемого устройства.

1.8. Неисправности и тестирование параллельных портов

Тестирование параллельных портов разумно начинать с *проверки их наличия* в системе. Список адресов установленных портов появляется в таблице, выводимой BIOS на экран перед загрузкой ОС. Список можно посмотреть и с помощью тестовых программ или прямо в BIOS Data Area с помощью отладчика.

Если BIOS обнаруживает меньше портов, чем установлено физически, скорее всего, двум портам присвоен один и тот же адрес. При этом работоспособность ни одного из конфликтующих портов не гарантируется: они будут одновременно выводить сигналы, но при чтении регистра состояния конфликт на шине, скорее всего, приведет к искажению данных. Программное тестирование порта без диагностической заглушки (Loop Back) не покажет ошибок, поскольку при этом читаются данные выходных регистров, а они у всех конфликтующих (по отдельности исправных портов) совпадут. Именно такое тестирование производит BIOS при проверке на наличие портов. Разбираться с такой ситуацией следует, последовательно устанавливая порты и наблюдая за адресами, появляющимися в списке.

Если физически установлен только один порт, а BIOS его не обнаруживает, то либо порт отключен при конфигурирование, либо он вышел из строя (скорее всего из-за нарушений правил подключения). Если вам везет, неисправность устраняется «передергиванием» платы в слоте — там иногда возникают проблемы с контактами.

Наблюдаются и такие «чудеса» — при «теплой» перезагрузке DOS после Windows 95 порт не виден (и приложения не могут печатать из MS-DOS). Однако после повторной перезагрузки DOS порт оказывается на своем месте. С этим явлением легче смириться, чем бороться.

Тестирование портов с помощью диагностических программ позволяет проверить выходные регистры, а при использовании специальных заглушек — и входные линии. Поскольку количество выходных линий порта (12) и входных (5) различно, то полная проверка порта с помощью пассивной заглушки принципиально невозможна. Разные программы тестирования требуют применения разных заглушек (рис. 1.14).

Большинство неприятностей при работе с LPT-портами доставляют *разъемы и кабели.* Для проверки порта, кабеля и принтера можно воспользоваться специальными тестами из популярных диагностических программ (Checkit, PC Check и т. п.). Можно попытаться просто вывести на принтер какой-либо символьный файл.

- Если вывод файла с точки зрения DOS проходит (копирование файла на устройство с именем LPTn или PRN совершается быстро и успешно), а принтер (исправный) не напечатал ни одного символа — скорее всего, это обрыв (неконтакт в разъеме) цепи Strobe#.
- Если принтер находится в состоянии *On Line,* а появляется сообщение о его неготовности, причину следует искать в линии Busy.

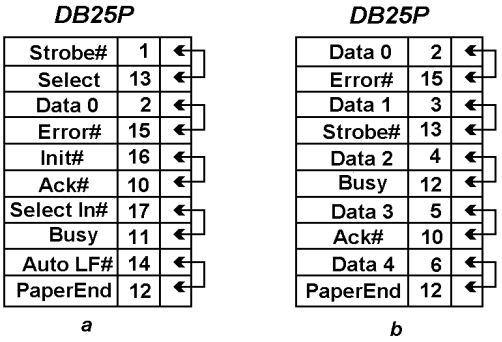


Рис. 1.14. Схема заглушки для тестирования LPT-порта: а — для Checkit, б — для Norton Diagnostics

- Если принтер, подключенный к порту, в стандартном режиме (SPP) печатает нормально, а при переходе в режим ECP начинаются сбои, следует проверить кабель — соответствует ли он требованиям IEEE 1284 (см. выше). Дешевые кабели с не перевитыми проводами нормально работают на скоростях 50-100 Кбайт/с, но при скорости 1-2 Мбайт/с, обеспечиваемой ECP, имеют полное право не работать, особенно при длине более 2 м.
- Если при установке драйвера PnP-принтера появилось сообщение о необходимости применения «двунаправленного кабеля», проверьте наличие связи контакта 17 разъема DB-25 с контактом 36 разъема Centronics. Хотя эта связь изначально предусматривалась, в ряде кабелей она отсутствует.

Аппаратные прерывания от LPT-порта используются не всегда. Даже DOS-программа фоновой печати PRINT работает с портом по опросу состояния, а ее обслуживающий процесс запускается по прерыванию от таймера. Поэтому неисправности, связанные с цепью прерывания от порта, проявляются не часто. Однако по-настоящему многозадачные ОС (например, NetWare) стараются работать с портом по прерываниям. Протестировать линию прерывания можно, только подключив к порту ПУ или заглушку. Если к порту с неисправным каналом прерывания подключить адаптер локальной сети, то он, возможно, будет работать, но с очень низкой скоростью: на любой запрос ответ будет приходить с задержкой в десятки секунд — принимаемый из адаптера пакет будет приниматься не по прерыванию (сразу по приходу), а по внешнему тайм-ауту.

Контрольные вопросы

1. Охарактеризуйте процедуру идентификации LPT-портов на стадии POST.
2. Охарактеризуйте процедуру печати копии экрана (Int 05h).
3. Охарактеризуйте функции драйвера принтера (Int 17h).
4. Как можно проверить наличие EPP BIOS?
5. Охарактеризуйте функцию 00h EPP BIOS.
6. Охарактеризуйте функцию 01h EPP BIOS.
7. Охарактеризуйте функцию 02h EPP BIOS.
8. Охарактеризуйте коды ошибок EPP BIOS.
9. Охарактеризуйте поддержку параллельным портом функций PnP.
10. Охарактеризуйте особенности обмена данными между двумя компьютерами через параллельные порты в полубайтном режиме.
11. Охарактеризуйте обмен данными между двумя компьютерами через LPT-порты в режиме ECP и байтном режиме.
12. Охарактеризуйте особенности подключения сканера через параллельный порт.
13. Охарактеризуйте особенности подключения через LPT-порт "симметричных" устройств ввода/вывода.
14. Охарактеризуйте назначение выводов разъема LPT-порта и биты регистров в режимах SPP, ECP и EPP.
15. На какие этапы разбиваются процедуры управления параллельными портами?
16. Как осуществляется конфигурирование LPT-порта, и какие его параметры подлежат конфигурированию?
17. Охарактеризуйте особенности применения скоростных режимов ECP, EPP и Fast Centronics?
18. С чего начинается тестирование LPT-портов?
19. Что можно проверить с помощью тестирования LPT-портов диагностирующими программами?
20. Как одновременно можно проверить порт, разъем порта и кабель подключения принтера?
21. В чем заключаются особенности проверки прерываний LPT-портов?

В данном разделе использованы материалы из [3, 4, 5, 7].

1.9. Программирование параллельного порта

Порт в режиме совместимости управляется тремя регистрами ввода/вывода, и адреса этих регистров различны для каждого порта (см. раздел 1.1.2.). Как уже упоминалось (раздел 1.4.1), область данных BIOS содержит базовые адреса (Base) регистров подключенных и поддерживаемых BIOS портов.

Регистр выходных данных, адрес которого является базовым для порта — это тот адрес порта, через который проходит каждый байт данных, посылаемый во внешнее устройство. Регистр статуса (состояния) с адресом (Base+1) сообщает различную информацию о состоянии внешнего устройства; процессор может постоянно опрашивать его, чтобы распознать момент, когда можно посылать данные внешнему устройству. Регистр управления с адресом (Base+2) инициализирует адаптер, внешнее устройство и управляет выводом данных. Он может также разрешить порту выработку запросов на прерывание, чтобы внешнее устройство могло посылать эти запросы процессору, когда оно готово к приему очередного символа, тем самым освобождая процессор от необходимости постоянно опрашивать регистр состояния и оставляя его свободным для других дел. Ниже приведены значения битов регистров состояния и управления SPP-порта (см. также раздел 1.1.2):

Регистр управления:		
бит 0	0 - нормальная установка	1 - вызывает вывод байта данных
бит 1	0 - нормальная установка	1 - автоматический перевод строки после возврата каретки
бит 2	0 - инициализировать порт принтера	1 - нормальная установка
бит 3	0 - отмена выбора принтера	1 - нормальная установка
бит 4	0 - прерывание принтера запрещено	1 - разрешено
биты 5 - 7	Не используются	
Регистр состояния:		
биты 0 - 2	Не используются	
бит 3	0 - ошибка принтера	1 - нет ошибки
бит 4	0 - принтер off-line	1 - принтер on-line
бит 5	0 - бумага вставлена	1 - нет бумаги
бит 6	0 - принтер подтверждает прием символа	1 - нормальная установка
бит 7	0 - принтер занят	1 - принтер свободен

Программирование параллельного порта на регистровом уровне используется в основном при написании драйверов для различных устройств с параллельными интерфейсами, протоколы которых отличаются от протокола интерфейса Centronics, или в прикладных программах, ориентированных на работу через параллельный порт на максимальной скорости.

Разновидность параллельного порта - двунаправленный порт также обычно используется в режиме совместимости - именно этот режим устанавливается изначально при выполнении программы POST. Однако программист может использовать расширенный режим работы порта для подключения нестандартной аппаратуры. В этом случае на компьютерах PS/2 выбор расширенного режима работы параллельного порта производится при конфигурации аппаратуры компьютера путем записи нулевого значения в бит 7 порта 0102h. В современных PC-совместимых компьютерах расширенные режимы работы контроллера параллельного порта могут быть выбраны при выполнении программы начальной конфигурации Setup BIOS, а при установке режима ECP становится возможным программное переключение его режимов работы.

1.9.1 Программирование SPP-порта

С программной точки зрения стандартный параллельный порт представлен тремя программно доступными регистрами. Посредством этих регистров осуществляется взаимодействие, как с контроллером порта, так и с подключенными к нему устройствами. При этом архитектура регистров позволяет на программном уровне реализовывать протоколы большинства параллельных интерфейсов.

Для упрощения взаимодействия с принтерами через стандартный параллельный порт может быть использована системная поддержка порта на уровне BIOS, представленная драйвером принтера INT 17h (см. раздел 1.4.3). Следует отметить, что функции прерывания INT 17h реализуют только протокол интерфейса Centronics.

1.9.1.1. Программирование SPP-порта на уровне регистров.

В качестве примера программирования стандартного параллельного порта на регистровом уровне рассмотрим реализацию процесса передачи байта данных через параллельный порт.

Для передачи одного байта данных необходимо выполнить следующие операции:

1. поместить передаваемый байт в регистр данных;
2. проверить готовность принтера к приему данных, прочитав регистр состояния и выделив старший разряд. Если значение старшего разряда равно 1, можно приступать к передаче байта; если значение равно 0 — снова опросить регистр состояния. Количество повторений цикла опроса должно быть ограничено по счетчику повторений или по времени, чтобы избежать зависания программы (если интервал ожидания готовности исчерпан, выдается сообщение об ошибке);
3. загрузить в регистр управления значение 0Dh (установка в 1 младшего разряда регистра управления формирует сигнал стробирования данных);
4. ожидать поступления сигнала подтверждения приема данных (когда данные приняты, бит 6 регистра состояния сбрасывается в 0). Цикл ожидания должен быть ограничен по времени;
5. загрузить в регистр управления значение 0Ch (в результате чего сигнал стробирования снимается).

Например, для параллельного порта LPT1, имеющего базовый адрес 378h, программный код передачи байта будет выглядеть следующим образом:

```
; Загрузить символ в регистр данных
        mov     DX,378h ;DX адресует регистр данных
        mov     AL,AH
        out     DX,AL
; Проверить состояние принтера
        inc     DX      ;DX адресует регистр состояния
        xor     CX,CX   ;выполнить 65536 циклов опроса
@@Busy: in      AL,DX
        test    AL,80h  ;проверить бит готовности
        jnz     @@Str1
        loop    @@Busy
        ; Интервал ожидания исчерпан, ошибка передачи
        jmp     @@Error
; Подать сигнал стробирования
@@Str1: inc     DX      ;DX адресует регистр управления
        mov     AL,0Dh
        out     DX,AL
; Ожидать поступление сигнала подтверждения
        dec     DX      ;DX адресует регистр состояния
        mov     CX,1000
@@Wait: in      AL,DX
        test    AL,40h  ;проверить бит подтверждения
        jnz     @@Str0
        loop    @@Wait
; Снять сигнал стробирования
@@Str0: inc     DX      ;DX адресует регистр управления
        mov     AL,0Ch
        out     DX,AL
```

В качестве другого примера рассмотрим реализацию в BIOS функции 00h (печать символа) прерывания INT 17h. Эта функция реализует протокол интерфейса Centronics. Следует отметить, что, начиная с BIOS для IBM PC/AT, выпущенного в марте 1986 года, считывание значения из регистра состояния параллельного порта производится дважды, что связано с тем, что изменение состояния сигнала BUSY имеет растянутый фронт.

Реализацию этой функции на уровне команд можно определить выполнением в пошаговом режиме, в среде отладчика реального режима, фрагмента программы, содержащей обращение к этой функции. Следует отметить, что все действия необходимо выполнять в операционной системе (ОС) DOS, а не из окна сеанса MS DOS ОС типа Windows 95/98. Это связано с тем, что графическая оболочка Windows перехватывает прерывание INT 17h и реализует его функции своим драйвером, работа которого не отслеживается отладчиком реального режима.

Ниже приводятся результаты отслеживания выполнения фрагмента программы:

```
2BE3:0100  B90004          MOV     CX,0400
2BE3:0103  BA0000          MOV     DX,0000
2BE3:0106  B400            MOV     AH,00
2BE3:0108  B021            MOV     AL,21
2BE3:010A  CD17            INT     17
2BE3:010C  E2F5            LOOP    0103
2BE3:010E  CC              INT3
```

в пошаговом режиме в среде отладчика AFD, при загрузке Windows 95 без графической оболочки, используя режим загрузки Command Prompt Only (из меню клавиши F8).

После выполнения команды INT 17 управление передается по адресу, указанному в векторе прерывания (четыре байта начиная с адреса 0:005C). Даже при загрузке ОС без графической оболочки прерывание перехватывается изменением содержимого вектора прерывания. В нашем случае происходит передача управления по адресу 06C5:0A28 (значение регистра CS (06C5) может быть иным в зависимости от конфигурации системы).

```
06C5:0A28  EAD2EF00F0      JMP     F000:EFD2
```

По этому адресу в режиме командной строки устанавливается команда передачи управления в BIOS по адресу F000:EFD2:

```
F000:EFD2  E9E003          JMP     F3B5
```

BIOS далее передает управление обработчику прерывания INT 17h (драйверу принтера), который и реализует выполнение функции 00h (печать символа):

```
F000:F3B5  52            PUSH   DX
F000:F3B6  51            PUSH   CX
F000:F3B7  53            PUSH   BX
F000:F3B8  57            PUSH   DI
F000:F3B9  56            PUSH   SI
F000:F3BA  55            PUSH   BP
F000:F3BB  FB            STI
F000:F3BC  83FA04        CMP    DX,0004
F000:F3BF  0F            DB     0F
F000:F3C0  83            DB     83
F000:F3C1  8800          MOV    [BX+SI],AL
F000:F3C3  80FC03        CMP    AH,03
F000:F3C6  7206          JC     F3CE
F000:F3C8  80EC02        SUB    AH,02
F000:F3CB  EB7E          JMP    F44B
F000:F3CD  90            NOP
F000:F3CE  8BFA          MOV    DI,DX
F000:F3D0  06            PUSH   ES
F000:F3D1  BB4000        MOV    BX,0040
F000:F3D4  8EC3          MOV    ES,BX
F000:F3D6  268A8D7800    MOV    CL,ES:[DI+0078]
F000:F3DB  D1E7          SHL    DI,1
F000:F3DD  268B950800    MOV    DX,ES:[DI+0008]
F000:F3E2  07            POP    ES
F000:F3E3  83FA00        CMP    DX,0000
F000:F3E6  0F            DB     0F
F000:F3E7  846100        TEST   [BX+DI+00],AH
F000:F3EA  8BF2          MOV    SI,DX
F000:F3EC  46            INC    SI
F000:F3ED  8BEA          MOV    BP,DX
F000:F3EF  83C502        ADD    BP,0002
F000:F3F2  50            PUSH   AX
F000:F3F3  32FF          XOR    BH,BH
F000:F3F5  8ADC          MOV    BL,AH
F000:F3F7  D1E3          SHL    BX,1
F000:F3F9  2EFFA7AFF3    JMP    CS:[BX+F3AF]
F000:F3FE  EE            OUT    DX,AL
F000:F3FF  E6E1          OUT    [E1],AL
F000:F401  8BD6          MOV    DX,SI
F000:F403  EC            IN     AL,DX
F000:F404  E6E1          OUT    [E1],AL
F000:F406  2480          AND    AL,80
F000:F408  7525          JNZ    F42F
F000:F40A  B490          MOV    AH,90
F000:F40C  B0FE          MOV    AL,FE
F000:F40E  CD15          INT    15
F000:F410  51            PUSH   CX
F000:F411  2E8B0E5FE7    MOV    CX,CS:[E75F]
F000:F416  2E8A3E61E7    MOV    BH,CS:[E761]
F000:F41B  B080          MOV    AL,80
F000:F41D  B480          MOV    AH,80
F000:F41F  E842F3        CALL   E764
F000:F422  59            POP    CX
F000:F423  0AE4          OR     AH,AH
F000:F425  7408          JZ     F42F
F000:F427  FEC9          DEC    CL
F000:F429  75E5          JNZ    F410
F000:F42B  0C01          OR     AL,01
F000:F42D  EB13          JMP    F442
F000:F42F  8BD5          MOV    DX,BP
F000:F431  B00D          MOV    AL,0D
F000:F433  EE            OUT    DX,AL
F000:F434  E6E1          OUT    [E1],AL
F000:F436  B00C          MOV    AL,0C
F000:F438  EE            OUT    DX,AL
F000:F439  E6E1          OUT    [E1],AL
F000:F43B  8BD6          MOV    DX,SI
F000:F43D  EC            IN     AL,DX
F000:F43E  E6E1          OUT    [E1],AL
F000:F440  24FE          AND    AL,FE
F000:F442  24F9          AND    AL,F9
F000:F444  3448          XOR    AL,48
F000:F446  8AE0          MOV    AH,AL
F000:F448  5B            POP    BX
F000:F449  8AC3          MOV    AL,BL
F000:F44B  5D            POP    BP
F000:F44C  5E            POP    SI
F000:F44D  5F            POP    DI
F000:F44E  5B            POP    BX
F000:F44F  59            POP    CX
F000:F450  5A            POP    DX
F000:F451  CF            IRET
```

Для нормального завершения этой функции необходимо, чтобы подключенный к порту принтер находился в состоянии готовности. В этом случае выполняются команды по следующим адресам:

с F000:F3B5 по F000:F3C6;
с F000:F3CE по F000:F408;
с F000:F42F по F000:F451.

Далее управление передается команде LOOP 0103 и выполнение функции драйвером заканчивается.

1.9.1.2. Программирование SPP-порта на уровне BIOS (INT 17h).

Функции прерывания BIOS INT 17h (см. раздел 1.4.3) используются в основном для взаимодействия с принтерами (печать символов, переключение режимов печати и т.п.). Однако они могут быть использованы и при работе с другими устройствами, поддерживающими интерфейс Centronics.

Независимо от того, в каком режиме осуществляется печать и какой командный язык управления печатью при этом используется, процесс вывода информации через параллельный порт заключается в побайтной передаче команд и данных с компьютера на принтер. Перечисленные в разделе 1.4.3 функции используют режим передачи стандартного параллельного порта (режим SPP), который является самым медленным, но зато поддерживается всеми моделями принтеров.

Ниже приведены процедуры, предназначенные для работы с принтером, подключенным к первому параллельному порту (LPT 1):

- процедура OutCharToLPT1 осуществляет вывод символа в порт LPT1 при помощи функций 00h прерывания BIOS INT 17h;
- процедура OutComniandToLPT1 использует подпрограмму OutCharToLPT1 для подачи на принтер командной последовательности символов (Esc-последовательности).

Реакция на ошибки, возникающие при обмене данными с принтером, в указанных функциях реализована примитивно: при любой ошибке выдается сообщение о том, что принтер не готов к печати, а затем происходит аварийное завершение работы программы (выход в DOS).

```
DATASEG
PErrTxt DB 12,27,"Принтер не готов к печати",0
ENDS

CODESEG
;******************************
;* ВЫВЕСТИ СИМВОЛ НА ПРИНТЕР *
;* Параметры:                 *
;* AL - код символа.          *
;******************************
PROC OutCharToLPT1 near
        pusha
; Вывести символ на печать
        mov     AH,0
        mov     DX,0
        int     17h
        test    AH,00101001b
        jnz     @@PrintingError
        popa
        ret
; ВЫДАТЬ СООБЩЕНИЕ ОБ ОШИБКЕ И ВЫЙТИ ИЗ ПРОГРАММЫ
@@PrintingError:
        ; Вывести сообщение об ошибке
        MFatalError PErrTxt
ENDP OutCharToLPT1
```

```
;****************************************
;*       ПОСЛАТЬ КОМАНДУ НА ПРИНТЕР     *
;* Параметры:                           *
;* DS:SI - указатель на строку команды. *
;* Первый байт строки содержит количество *
;* байтов команды, посылаемых на принтер. *
;****************************************
PROC OutCommandToLPT1 near
        pusha
        cld
; Загрузить счетчик байтов команды в CX
        lodsb
        xor     CX,CX
        mov     CL,AL
@@OutNextByte:
        lodsb
        call    OutCharToLPT1
        loop    @@OutNextByte
        popa
        ret
ENDP OutCommandToLPT1
ENDS
```

Эти процедуры будут использоваться в примерах, демонстрирующих печать в растровом режиме на различных типах принтеров (как подключаемый файл "list7_01.inc") (в пособие, посвященном программированию принтеров). В свою очередь в этой программе для выдачи сообщения об ошибке используется макрокоманда MfatalError, которая определена ниже в листинге 1.4 и становится доступной из подключаемого файла "list1_04.inc". В свою очередь макрокоманды, определенные в листинге 4, используют процедуры ввода-вывода листинга 2, доступные из подключаемого файла "list1_02.inc".

СОВЕТ: В случае если принтер на вашем компьютере подключен к порту LPT2, нужно изменить номер порта в процедуре OutCharToLPT1, то есть перед вызовом прерывания поместить в регистр DX значение 1 вместо 0 или поменять базовые адреса портов в рабочей области BIOS (0040:0008 и 0040:000A).

1.9.2.Программирование ECP порта

Спецификация ECP была разработана фирмами Microsoft и Hewlett-Packard. Она предусматривает введение в контроллер параллельного порта дополнительного блока регистров, изменение назначения стандартных регистров и использование специальных протоколов, увеличивающих скорость передачи данных более чем на порядок (с 1-50 Кбайт/с в режиме SPP до 2-5 Мбайт/с в режиме ECP).

Работа с параллельным портом в режиме ECP не поддерживается BIOS, поэтому его программирование реализуется на регистровом уровне. Для этого необходимо достаточно детальное знакомство с регистровой архитектурой ECP порта и протоколами его взаимодействия с периферийными устройствами.

Общие вопросы работы параллельного ECP порта уже были рассмотрены в разделах 1.3.4. и 1.3.5. В данном разделе более подробно рассмотрены вопросы, связанные с работой ECP порта и программным управлением режимами работы его контроллера.

Примечание: В этом разделе мнемоника обозначений разрядов регистров контроллера и сигналов на разъеме ECP порта несколько отличается от аналогичных обозначений, используемых в разделах 1.3.4. и 1.3.5 (используется альтернативные обозначения). Это связано с тем, что в данном разделе используется информация из [5] и эта мнемоника используется в пояснениях к ниже приведенным программам. В данном разделе по возможности указывается в скобках и альтернативная мнемоника.

1.9.2.1. Управление работой контроллера ECP

Для управления работой контроллера могут использоваться сигналы прерываний.

Контроллер ECP генерирует прерывания в следующих случаях:

- serviceIntr (ECR.2) = 0, dmaEn (ECR.3) = 1, завершена прямая передача данных (счетчик DMA достиг конечного значения);
- serviceIntr = 0, dmaEn = 0, direction (DCR.5) = 0 и имеется свободное место в очереди данных;
- serviceIntr = 0, dmaEn = 0, direction = 1 и в очереди имеется по крайней мере один байт данных;
- nErrIntEn (ECR.4) = 0 и сигнал nFault (DSR.3) переключился с высокого уровня на низкий;
- при переключении бита nErrIntEn из 1 в 0, если сигнал nFault имеет низкий уровень;
- ackIntEn (DCR.4) = 1 и сигнал nAsk (DSR.6) переходит из 0 в 1.

Переключение режимов работы контроллера ECP осуществляется программным обеспечением через дополнительный регистр управления Ecr. При установке направления передачи данных используется также бит direction регистра управления Dcr.

Из режимов 000 и 001 контроллер можно сразу переключить в любой другой режим. Если контроллер находится в одном из режимов 010-111, то его можно переключить только в режим 000 или режим 001; для переключения в другие режимы необходимо вначале переключиться в режим 000 или 001. Переключение бита direction в регистре управления может производиться только в режиме 001. Перед переключением режима необходимо убедиться, что в очереди FIFO нет данных (очередь пуста).

Перед началом использования любого режима передачи данных, отличного от стандартного режима SPP, программное обеспечение должно выполнить процедуру переговоров (negotiation) с периферийным устройством для согласования режима работы. Согласование осуществляется в соответствии со стандартом IEEE 1284.1.

ВНИМАНИЕ. Переговоры с периферийным устройством могут выполняться только в режимах 000 и 001 контроллера ECP.

После завершения процедуры согласования режима необходимо инициализировать регистр управления Dcr, загрузив в него код 0Ch при установке режима SPP:

strobe = 0, autofd = 0, nInit = 1, Se1ectIn = 1, ackIntEn = 0, direction = 0;

или код 04h при установке режимов IEEE 1284.

Далее производится установка режима ECP путем загрузки кода 74h в дополнительный регистр управления Ecr.

1.9.2.2. Процедура переговоров

Для того чтобы между контроллером параллельного порта и периферийным устройством была возможна передача данных, режимы работы контроллера и устройства должны быть согласованы друг с другом.

Стандарт IEEE 1284.1 предусматривает определенный порядок переключения периферийных устройств (в том числе — принтеров) из стандартного режима SPP в другие режимы передачи данных, а также порядок возврата в стандартный режим передачи.

Процедуры переговоров для разных режимов различаются между собой. Ниже мы будем рассматривать только процедуру переключения устройства в режим ECP и процедуру возврата в режим SPP из режима ECP.

Стандарт IEEE 1284.1 рассматривает различные протоколы передачи данных с точки зрения сигналов на линиях связи, но для программиста такой подход неудобен, так как некоторые сигналы представлены в регистрах контроллера порта в инверсной форме. Во избежание путаницы с обозначением сигналов на линиях кабеля и в регистрах контроллера процедуру переговоров будем рассматривать с точки зрения программиста, работающего с регистрами порта.

Для того чтобы переключить периферийное устройство в режим ECP, необходимо выполнить перечисленные ниже операции (см. раздел 1.3.5).

1. Установить режим передачи SPP, загрузив в дополнительный регистр управления код 14h (прерывания блокированы, использование DMA запрещено, прерывание по сигналу nFault запрещено, режим работы — SPP).
2. Загрузить в регистр данных код режима ECP — значение 10h (см. табл. 1.10 в разделе 1.3.5).
3. Загрузить в регистр управления код 0Ch (strobe = 0, autofd = 0, nInit = 1, SelectIn = 1).
4. Вставить задержку на один тик системного таймера (0,05 с).
5. Проверить готовность принтера к работе (биты Select и nBusy в регистре состояния должны быть установлены в 1).
6. Начать фазу переговоров, загрузив в регистр управления код 06h (strobe = 0, autofd = 1, nInit = 1, SelectIn = 0).
7. Вставить задержку на один тик системного таймера.
8. Проверить наличие в регистре состояния следующей комбинации сигналов: nFault = 1, Select = 1, PEггог = 1, nAsk = 0 (любая другая комбинация значений указывает на то, что принтер не соответствует стандарту IEEE 1284.1).
9. Загрузить в регистр управления код 07h (strobe = 1, autofd = 1, nInit = 1, SelectIn = 0).
10. Вставить задержку на одну микросекунду.
11. Загрузить в регистр управления код 04h (strobe = 0, autofd = 0, nimt = 1, SelectIn = 0).
12. Вставить задержку на один тик системного таймера.
13. Проверить наличие в регистре состояния следующей комбинации сигналов: Se1ect = 1, PEггог = 0, nAsk = 1 (любая другая комбинация значений указывает на то, что принтер не поддерживает режим ECP).
14. Начать фазу установки, загрузив в регистр управления код 06h (strobe = 0, autofd = 1, nInit = 1, SelectIn = 0).
15. Вставить задержку на один тик системного таймера.
16. Проверить значение бита PEггог в регистре состояния (если он не установлен в 1, произошел сбой).
17. Установить для контроллера порта режим ECP, загрузив в дополнительный регистр Ecr код 74h.
18. Установить нулевой адрес канала, загрузив в регистр EcpAFifo (регистр очереди адресов ECP)

константу 00h.

Если все перечисленные операции выполнены успешно, принтер готов к приему данных в режиме ECP.

ПРИМЕЧАНИЕ.
После выполнения фазы переговоров для устройства по умолчанию устанавливается нулевой адрес канала ECP. По этому адресу выполняется передача данных на принтер, а другие адреса обычно не используются.

После того как передача данных завершена, нужно перевести периферийное устройство из режима ECP в стандартный режим (процесс возврата в стандартный режим именуется в документации завершающей фазой).

Для выхода из режима ECP требуется выполнить указанные ниже операции.

1. Установить для контроллера порта режим SPP, загрузив в дополнительный регистр управления код 14h.
2. Загрузить в регистр управления код 0Ch (strobe = 0, autofd = 0, nInit = 1, SelectIn = 1).
3. Вставить задержку на один тик системного таймера.
4. Проверить наличие в регистре состояния следующей комбинации сигналов: nFault = 1, Select = 0, nAsk = 0, nBusy = 0 (любая другая комбинация означает сбой в работе принтера).
5. Загрузить в регистр управления код 0Eh (strobe = 0, autofd = 1, nInit = l, Se1ectIn = 1).
6. Вставить задержку на один тик системного таймера.
7. Проверить значение бита nAsk в регистре состояния (если он не установлен в 1, произошел сбой).
8. Загрузить в регистр управления код 0Ch (strobe = 0, autofd = 0, nInit = l,Se1ectIn = 1).

1.9.2.3. Передача данных в режиме ECP

Существует три способа передачи данных в режиме ECP:

- программно-управляемая передача данных;
- передача данных по прерываниям;
- передача данных в режиме DMA.

Режим программно-управляемой передачи самый простой и самый неэффективный, так как требует постоянного контроля состояния очереди данных. Все процессорное время в этом случае тратится на управление передачей, причем большая часть расходуется на опрос регистра Ecr. Возможно несколько вариантов реализации программной передачи.

1. Процессор ожидает полного освобождения очереди и загружает в нее один байт данных.
2. Процессор ожидает появления свободного места в очереди и загружает в нее байты данных, пока очередь не будет заполнена.
3. Процессор ожидает полного освобождения очереди и загружает в нее сразу 16 байт данных.

Режим передачи по прерываниям позволяет освободить процессор от необходимости постоянно контролировать состояние очереди данных. Чтобы начать передачу данных по прерываниям, нужно настроить соответствующий порту вектор прерывания на обработчик прерывания и сбросить в 0 бит блокировки служебных прерываний serviceIntr в регистре Ecr. В режиме передачи прерывание генерируется в том случае, когда в очереди появляется свободное место; в режиме приема прерывание вырабатывается, когда в очереди имеется хотя бы один байт. Обработчик прерывания должен знать, в каком режиме работы (передачи или приема) находится порт в данный момент, и в соответствии с этим либо загружать байт в очередь, либо считывать байт из очереди.

Использование DMA — самый эффективный, но самый сложный (с точки зрения настройки) способ передачи данных.

Подготовка к передаче в режиме DMA начинается с установки направления передачи. Далее следует настроить третий канал контроллера DMA, загрузив в него адрес буфера данных в оперативной памяти и количество передаваемых байтов. Затем нужно установить вектор прерывания от параллельного порта на обработчик прерывания по завершению передачи в режиме DMA. После этого нужно разрешить служебные прерывания и использование DMA, для чего следует сбросить в 0 бит блокировки служебных прерываний serviceIntr и установить в 1 бит управления режимом DMA dmaEn в регистре Ecr. Процесс передачи данных будет происходить без участия центрального процессора; когда передача данных будет завершена, следует запретить служебные прерывания и режим DMA.

1.9.2.4. Переключение направления передачи данных

Переключение направления передачи данных производится путем изменения значения бита direction в регистре управления. Переключение направления передачи, однако, разрешено только в режиме совместимости, поэтому контроллер должен быть временно переключен в режим 001. В этом режиме контроллер должен выполнить процедуру согласования направления передачи данных.

Процесс передачи данных от компьютера к периферийному устройству в спецификации ECP называется фазой прямой передачи данных (Forward Phase), а процесс передачи от периферийного

устройства к компьютеру — фазой обратной передачи (Reverse Phase). После установки режима ECP контроллер параллельного порта и периферийное устройство настроены на прямую передачу данных.

Для переключения направления передачи с прямого на инверсное нужно выполнить перечисленные ниже операции (см. раздел 1.3.4.1).

1. Загрузить в регистр управления код 06h (strobe = 0, autofd = 1, nInit = l, Se1ectIn = 0).
2. Вставить задержку на одну микросекунду.
3. Загрузить в регистр управления код 02h (strobe = 0, autofd = 1, nInit = 0, Se1ectIn = 0).
4. Вставить задержку на один тик системного таймера.
5. Проверить значение бита PError - сигнал PaperEnd на входе 12 разъема SPP-порта (см. табл. 1.7 раздела 1.3.4.1) в регистре состояния (если он не сброшен в 0, произошел сбой).

Для переключения направления передачи с инверсного на прямое нужно выполнить указанные далее операции.

1. Загрузить в регистр управления код 06h (strobe = 0, autofd = 1, nInit = l, Se1ectIn = 0).
2. Вставить задержку на один тик системного таймера.
3. Проверить значение бита PError в регистре состояния (если он не установлен в 1, произошел сбой).

ПРИМЕЧАНИЕ

Использование инверсного режима связано с определенным риском и требует неукоснительного соблюдения правил переключения между прямым и инверсным режимом, указанных в стандарте IEEE 1284.1 и спецификации ECP. При нарушении порядка переключения параллельный порт компьютера и порт принтера могут одновременно оказаться в режиме передачи, что может привести к повреждению выходных усилителей портов.

1.9.2.5. Пример программирования ECP-порта

Пример программы, которая выполняет переключение в режим ECP, передает на принтер строку символов, состоящую из заглавных латинских букв от A до Z, а затем выполняет возврат в режим SPP, приведен в листинге 1.1.

Листинг 1.1. Программа для проверки способности принтера работать в режиме ECP

```
IDEAL
P386
LOCALS
MODEL MEDIUM

; Подключить файл мнемонических обозначений
; кодов управляющих клавиш и цветовых кодов
include "list1_03.inc"
; Подключить файл макросов
include "list1_04.inc"

DATASEG
; Текстовые сообщения
Txt1 DB LIGHTCYAN,0,19
     DB "ПЕЧАТЬ ТЕКСТА С ИСПОЛЬЗОВАНИЕМ РЕЖИМА ECP",0
AnyK DB YELLOW,24,29,"Нажмите любую клавишу",0
WPrn DB YELLOW,12,26,"Ждите завершения печати ...",0
Err1 DB 12,25,"Порт находится не в режиме ECP",0
Err2 DB 12,27,"Принтер не готов к работе",0
Err3 DB 12,21,"Режим ECP не поддерживается принтером",0
ENDS

SEGMENT sseg para stack 'STACK'
        DB 400h DUP(?)
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC Test_ECP_Mode
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Проверить режим работы порта
        mov     DX,378h+402h
        in      AL,DX
        cmp     AL,0FFh
        je      @@Err1
; Вывести текстовые сообщения на экран
        mov     SI,offset Txt1
        call    ShowColorString
        mov     SI,offset WPrn
        call    ShowColorString

; ПОДГОТОВИТЕЛЬНАЯ ФАЗА
; Установить для порта режим SPP
        mov     DX,378h+402h
        mov     AL,00010100b
        out     DX,AL
; Загрузить код режима ECP в регистр данных
        mov     DX,378h
        mov     AL,00010000b
        out     DX,AL
; Установить strobe в 0, AutoFeed в 0, SelectIn в 1
        mov     DX,378h+2
        mov     AL,00001100b
        out     DX,AL
        call    Wait05s ;задержка на 0,05 с
; Принтер готов к работе?
        mov     DX,378h+1
        in      AL,DX
        and     AL,10010000b
        cmp     AL,10010000b
        jne     @@Err2

; ФАЗА ПЕРЕГОВОРОВ
; Установить strobe в 0, AutoFeed в 1, SelectIn в 0
        mov     DX,378h+2
        mov     AL,00000110b
        out     DX,AL
        call    Wait05s ;задержка на 0,05 с
; Принтер соответствует стандарту IEEE 1284?
        mov     DX,378h+1
```

```
        in      AL,DX
        and     AL,01111000b
        cmp     AL,00111000b
        jne     @@Err3
; Установить strobe в 1, AutoFeed в 1, SelectIn в 0
        mov     DX,378h+2
        mov     AL,00000111b
        out     DX,AL
        call    Waitlus ;задержка на 1 мкс
; Установить strobe в 0, AutoFeed в 0, SelectIn в 0
        mov     DX,378h+2
        mov     AL,00000100b
        out     DX,AL
        call    Wait05s ;задержка на 0,05 с
; Принтер поддерживает режим ECP?
        mov     DX,378h+1
        in      AL,DX
        and     AL,01110000b
        cmp     AL,01010000b
        jne     @@Err3

; ФАЗА УСТАНОВКИ
; Установить strobe в 0, AutoFeed в 1, SelectIn в 0
        mov     DX,378h+2
        mov     AL,00000110b
        out     DX,AL
        call    Wait05s ;задержка на 0,05 с
; Бит PError установлен?
        mov     DX,378h+1
        in      AL,DX
        test    AL,00100000b
        jz      @@Err3
; Установить для порта режим ECP

        mov     DX,378h+402h
        mov     AL,01110100b
        out     DX,AL
; Установка нулевого адреса канала
        mov     DX,378h
        mov     AL,0
        out     DX,AL

; ПЕРЕДАЧА ДАННЫХ НА ПРИНТЕР
; Печать строки символов (от A до Z)
        mov     AL,'A'
@@OutNextChar:
        call    ECP_Out
        inc     AL
        cmp     AL,'Z'
        jbe     @@OutNextChar
        mov     AL,0Dh ;возврат каретки
        call    ECP_Out
        mov     AL,0Ah ;перевод строки
        call    ECP_Out
        mov     AL,0Ch ;перевод формата
        call    ECP_Out

; Вывести текстовые сообщения на экран
        call    ClearScreen
        mov     SI,offset Txt1
        call    ShowColorString
        mov     SI,offset AnyK
        call    ShowColorString
; Ожидать нажатия клавиши
        call    GetChar

; ЗАВЕРШАЮЩАЯ ФАЗА
; Установить для порта режим SPP
        mov     DX,378h+402h
        mov     AL,00010100b
        out     DX,AL
; Установить SelectIn в 1, AutoFeed в 0, strobe в 0
        mov     DX,378h+2
        mov     AL,00001100b
        out     DX,AL
        call    Wait05s ;задержка на 0,05 с
; В регистре состояния присутствует комбинация
; nFault=1, Select=0, nAck=0, nBusy=0?
        mov     DX,378h+1
        in      AL,DX
        and     AL,11011000b
        cmp     AL,00001000b
        jne     @@Err3
; Установить strobe в 0, AutoFeed в 1, SelectIn в 1

        mov     DX,378h+2
        mov     AL,00001110b
        out     DX,AL
        call    Wait05s ;задержка на 0,05 с
; В регистре состояния бит nAck=1?
        mov     DX,378h+1
        in      AL,DX
        test    AL,01000000b
        jz      @@Err3
; Установить SelectIn в 1, AutoFeed в 0, strobe в 0
        mov     DX,378h+2
        mov     AL,00001100b
        out     DX,AL

; Переустановить текстовый режим
        mov     ax,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h

; Сообщения об ошибках
@@Err1: MFatalError Err1 ;порт не в режиме ECP
@@Err2: MFatalError Err2 ;принтер не готов к работе
@@Err3: MFatalError Err3 ;режим ECP не поддерживается
ENDP Test_ECP_Mode

;************************************************
;* ПРОЦЕДУРА ДЛЯ ВЫВОДА БАЙТА ДАННЫХ В РЕЖИМЕ ECP *
;* Передаваемые параметры:                      *
;* AL - выводимый байт данных.                  *
;************************************************
PROC ECP_Out near
        push    AX
        push    DX
        mov     AH,AL
; Ожидать очистки очереди данных ECP
        mov     DX,378h+402h
@@Wait_FIFO_Empty:
        in      AL,DX
        and     AL,00000001b
        jz      @@Wait_FIFO_Empty
; Загрузить байт данных в очередь
        sub     DX,2
        mov     AL,AH
        out     DX,AL
        pop     DX
        pop     AX
        ret
ENDP ECP_Out
;*****************************************
;* ЗАДЕРЖКА НА ОДИН ТИК СИСТЕМНОГО ТАЙМЕРА *
;*****************************************
PROC Wait05s near
        push    ES
        push    EAX
        mov     AX,0
        mov     ES,AX
        mov     EAX,[ES:046Ch]
        inc     EAX
@@Wait: cmp     EAX,[ES:046Ch]
        jae     @@Wait
        pop     EAX
        pop     ES
        ret
ENDP Wait05s
```

```
;*********************************
:* ЗАДЕРЖКА НА ОДНУ МИКРОСЕКУНДУ *
;*********************************
PROC Wait1us near

        push    CX
        mov     CX,1000
@@Wait: nop
        loop    @@Wait
        pop     CX
        ret
ENDP Wait1us
ENDS

; Подключить процедуры вывода данных на экран
include "list1_02.inc"

END
```

ПРИМЕЧАНИЕ

Пример из листинга 1.1 пригоден только для принтеров, поддерживающих текстовый режим печати. Перед запуском программы контроллер порта должен быть настроен на режим ECP. Если системная плата поддерживает EPP BIOS, можно выполнить переключение контроллера при помощи программы, приведенной в листинге 1.5. Если EPP BIOS не поддерживается, нужно переключить контроллер в режим ECP в процессе начальной загрузки компьютера, с помощью BIOS SETUP. Проверьте также базовый адрес набора регистров порта — пример рассчитан на использование стандартного адреса (378h).

В программе для проверки способности принтера работать в режиме ECP (листинг 1.1) используется несколько вспомогательных процедур, необходимых для отображения информации на экране монитора и ввода с клавиатуры. Подобные процедуры общего назначения объединены в отдельную группу, оформленную в виде подключаемого файла (include-файла), приведенного в листинге 1.2. Данный файл используется и в других примерах программ, которые будут представлены в последующих разделах. Он может быть использован и при реализации тестовых программ для других подсистем PC.

Назначение подпрограмм, включенных в листинге 2 следующее:

- процедура ShowASCIIChar осуществляет вывод символа в ASCII-коде в заданную позицию экрана;
- процедура ShowHexByte отображает в заданную позицию экрана содержимое регистра AL (байт данных) в шестнадцатеричном коде;
- процедура ShowHexWord отображает в заданную позицию экрана содержимое регистра AX (слово данных) в шестнадцатеричном коде;
- процедура ShowHexDWord отображает в заданную позицию экрана содержимое регистра EAX (двойное слово) в шестнадцатеричном коде;
- процедура ShowBinByte отображает в заданную позицию экрана содержимое регистра AL (байт данных) в двоичном коде;
- процедура ShowBinDWord отображает в заданную позицию экрана содержимое регистра EAX (двойное слово) в двоичном коде;
- процедура ShowString выводит текстовую строку в заданную область экрана, причем используется цвет символов и фона, заданный по умолчанию;
- процедура ShowText использует процедуру ShowString для вывода текста (группы строк, цвет которых определяется по умолчанию) на экран;
- процедура ShowColorString выводит текстовую строку заданного цвета в заданную область экрана;
- процедура ShowColorText использует процедуру ShowColorString для вывода разноцветного текста (группы строк, цвет каждой из которых задается индивидуально) на экран;
- процедура ShowASCIIField служит для вывода на экран текстового поля фиксированного размера;
- процедура SetCursorPosition предназначена для управления положением курсора, но в данном примере используется только для того, чтобы убрать курсор за пределы экрана (сделать курсор невидимым);
- процедура GetChar ожидает ввода любого символа с клавиатуры, а затем считывает ASCII-код и скан-код этого символа;
- процедура WaitChar проверяет наличие символа в буфере клавиатуры, и если символ есть — считывает его ASCII-код и скан-код;
- процедура C1earScreen осуществляет очистку экрана;
- процедура Beep предназначена для выдачи оператору звукового сигнала (обычно в случае какой-либо ошибки);
- процедура FatalError осуществляет выдачу сообщения о фатальной ошибке, после чего производится экстренное (аварийное) прекращение работы программы.

Процедуры вывода информации на экран используют метод прямой записи в видеопамять с учетом особенностей текстового режима:

- каждый символ кодируется в видеопамяти двумя байтами: первый байт содержит код символа, а второй описывает цвет символа и фон знакоместа;
- видеопамять отображена на адресное пространство процессора до адреса 0B8000h (для доступа к ней

обычно применяются сегментный регистр ES, в который записывается число B800h, и любой индексный регистр, в который заносится смещение символа от начала видеопамяти);

- длина одной строки составляет 80 символов (160 байт);
- на экране помещается 25 текстовых строк.

Листинг 1.2. Процедуры ввода-вывода общего назначения для работы в цветном текстовом режиме (подключаемый файл list1_02.inc).

```
DATASEG
; Цвет и фон выводимого текста (по умолчанию установим
; вывод белого текста по черному фону)
TextColorAndBackground DB 0Fh
; Начальная позиция для вывода текстовой строки на экран
ScreenString DW ?
ScreenColumn DW ?
ENDS

CODESEG
; Адрес основного сегмента данных
MainDataSeg DW ?
;*************************************************
;*        ВЫВОД БАЙТА НА ЭКРАН В КОДЕ ASCII      *
;* Подпрограмма выводит содержимое регистра AL в  *
;* коде ASCII в указанную позицию экрана.         *
;* Координаты позиции передаются через глобальные *
;* переменные ScreenString и ScreenColumn. После  *
;* выполнения операции вывода происходит автомати- *
;* ческое приращение значений этих переменных.    *
;*************************************************
PROC ShowASCIIChar near
        pusha
        push    DS
        push    ES
        mov     DI,[CS:MainDataSeg]
        mov     DS,DI
        cld
; Настроить пару ES:DI для прямого вывода в видеопамять
        push    AX
        ; Загрузить адрес сегмента видеоданных в ES
        mov     AX,0B800h
        mov     ES,AX
        ; Умножить номер строки на длину строки в байтах
        mov     AX,[ScreenString]
        mov     DX,160
        mul     DX
        ; Прибавить номер колонки (дважды)
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX
        pop     AX
        mov     AH,[TextColorAndBackground]
        stosw

; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 2 символа влево
        inc     [ScreenColumn]
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@End
        ; Если достигнута правая граница экрана -
        ; перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
@@End:  pop     ES
        pop     DS
        popa
        ret
ENDP ShowASCIIChar

;*************************************************
;*    ВЫВОД БАЙТА НА ЭКРАН В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ    *
;* Подпрограмма выводит содержимое регистра AL        *
;* в шестнадцатеричном коде в заданную позицию экрана. *
;* Координаты позиции передаются через глобальные     *
;* переменные ScreenString и ScreenColumn. После      *
;* выполнения операции вывода происходит автомати-    *
;* ческое приращение значений этих переменных.        *
;*****************************************************
PROC ShowHexByte near
        pusha
        push    DS
        push    ES
; Настроить DS на глобальный сегмент данных
        mov     DI,[CS:MainDataSeg]
        mov     DS,DI
        cld

; Настроить пару ES:DI для прямого вывода в видеопамять
        push    AX
        ; Загрузить адрес сегмента видеоданных в ES
        mov     AX,0B800h
        mov     ES,AX
        ; Умножить номер строки на длину строки в байтах
        mov     AX,[ScreenString]
        mov     DX,160
        mul     DX
        ; Прибавить к полученному произведению номер
        ; колонки (дважды)
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX
        pop     AX

; Использовать цвет символов, заданный по умолчанию
        mov     AH,[TextColorAndBackground]
; Вывести старший разряд числа
        push    AX
        ; Выделить старший разряд
        shr     AL,4
        ; Преобразовать старший разряд в код ASCII
        add     AL,'0'
        cmp     AL,'9'
        jbe     @@M0
        add     AL,'A'-'9'- 1
        ; Вывести разряд числа на экран
@@M0:   stosw
        pop     AX
; Вывести младший разряд числа
        ; Выделить младший разряд числа
        and     AL,0FH
        ; Преобразовать младший разряд в код ASCII
        add     AL,'0'
        cmp     AL,'9'
        jbe     @@M1
        add     AL,'A'-'9'- 1
        ; Вывести разряд числа на экран
@@M1:   stosw

; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 2 символа влево
        add     [ScreenColumn],2
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@End
        ; Если достигнута правая граница экрана -
        ; перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
@@End:  pop     ES
        pop     DS
        popa
        ret
ENDP ShowHexByte
```

```
;*****************************************************
;*         ВЫВОД 16-РАЗРЯДНОГО СЛОВА НА ЭКРАН        *
;*              В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ             *
;* Параметры:                                        *
;* AX - число, которое будет выведено на экран.      *
;* Номер строки передается через глобальную          *
;* переменную ScreenString, номер столбца - через    *
;* переменную ScreenColumn, цвет текста определяется *
;* глобальной переменой TextColorAndBackground.      *
;*****************************************************
PROC ShowHexWord NEAR
        xchg    AL,AH
        call    ShowHexByte
        xchg    AL,AH
        call    ShowHexByte
        ret
ENDP ShowHexWord

;*****************************************************
;*         ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН        *
;*              В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ             *
;* Параметры:                                        *
;* EAX - число, которое будет выведено на экран.     *
;* Номер строки передается через глобальную          *
;* переменную ScreenString, номер столбца - через    *
;* переменную ScreenColumn, цвет текста определяется *
;* глобальной переменой TextColorAndBackground.      *
;*****************************************************
PROC ShowHexDWord NEAR
        rol     EAX,8
        call    ShowHexByte
        rol     EAX,8
        call    ShowHexByte
        rol     EAX,8
        call    ShowHexByte
        rol     EAX,8
        call    ShowHexByte
        ret
ENDP ShowHexDWord

;*************************************************
;*      ВЫВОД БАЙТА НА ЭКРАН В ДВОИЧНОМ КОДЕ     *
;* Подпрограмма выводит содержимое регистра AL   *
;* в двоичном коде в указанную позицию экрана.   *
;* Координаты позиции передаются через глобальные *
;* переменные ScreenString и ScreenColumn. После *
;* выполнения операции вывода происходит автома- *
;* тическое приращение значений этих переменных. *
;*************************************************
PROC ShowBinByte near
        pusha
        push    DS
        push    ES
        ; Копируем отображаемый байт в BL
        mov     BL,AL
        mov     AX,[CS:MainDataSeg]
        mov     DS,AX
        cld
        ; Загрузить адрес "текстовой" видеопамяти в ES
        mov     AX,0B800h
        mov     ES,AX
        ; Умножить номер строки на длину строки в байтах
        mov     AX,[ScreenString]
        mov     DX,160
        mul     DX
        ; Прибавить дважды номер колонки
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        ; Переписать результат в индексный регистр
        mov     DI,AX

; Отобразить разряды числа (начиная со старшего)
        mov     AH,[TextColorAndBackground]

        mov     CX,8    ;счетчик разрядов
@@L0:   mov     AL,'0'
        mov     AL,'1'
        ; Вывести разряд числа на экран
@@L1:   stosw
        loop    @@L0

; Подготовка для вывода следующих байтов
        ; Перевести текущую позицию на 8 символов влево
        add     [ScreenColumn],8
        ; Проверить пересечение правой границы экрана
        cmp     [ScreenColumn],80
        jb      @@End
        ; Если достигнута правая граница экрана -
        ; перейти на следующую строку
        sub     [ScreenColumn],80
        inc     [ScreenString]
; Конец подпрограммы
@@End:  pop     ES
        pop     DS
        popa
        ret
ENDP ShowBinByte
;*****************************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА НА ЭКРАН В ДВОИЧНОМ КОДЕ *
;* Параметры:                                        *
;* AX - число, которое будет выведено на экран.      *
;* Номер строки передается через глобальную          *
;* переменную ScreenString, номер столбца - через    *
;* переменную ScreenColumn, цвет текста определяется *
;* глобальной переменой TextColorAndBackground.      *
;*****************************************************
PROC ShowBinWord NEAR
        rol     AX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     AX,8
        call    ShowBinByte
        ret
ENDP ShowBinWord

;*****************************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА НА ЭКРАН В ДВОИЧНОМ КОДЕ *
;* Параметры:                                        *
;* EAX - число, которое будет выведено на экран.     *
;* Номер строки передается через глобальную          *
;* переменную ScreenString, номер столбца - через    *
;* переменную ScreenColumn, цвет текста определяется *
;* глобальной переменой TextColorAndBackground.      *
;*****************************************************
PROC ShowBinDWord NEAR
        rol     EAX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    ShowBinByte
        inc     [ScreenColumn]
        rol     EAX,8
        call    ShowBinByte
        ret
ENDP ShowBinDWord

;*****************************************************
;*         ВЫВОД ТЕКСТОВОЙ СТРОКИ НА ЭКРАН           *
;* Все параметры передаются через одну структуру:    *
;* первый байт - номер начальной строки (0-24);      *
;* второй байт - номер начальной колонки (0-79);     *
;* далее идет строка, ограниченная нулем.            *
;* Адрес структуры передается через регистры DS:SI.  *
;*****************************************************
PROC ShowString near
        push    AX
        push    BX
        push    DI
        push    ES
; Настроить регистр ES на глобальный сегмент данных
        mov     AX,[CS:MainDataSeg]
        mov     ES,AX
; Запомнить цвет текста в BL
        mov     BL,[ES:TextColorAndBackground]
; Настроить регистр ES на видеопамять
        mov     AX,0B800h
        mov     ES,AX
```

```
        cld
; Вычислить адрес для строки в видеопамяти
        ; Загрузить номер строки экрана в AL и
        ; умножить его на длину строки в байтах
        lodsb
        ; Проверка: номер строки не должен превышать
        ; предела нижней границы экрана
        cmp     AL,24
        ja      @@Error
        mov     AH,160
        mul     AH
        ; Переписать результат в индексный регистр DI
        mov     DI,AX
        ; Загрузить номер столбца и дважды
        ; прибавить его к DI
        lodsb
        cmp     AL,79   ;номер колонки не должен
        ja      @@Error ;превышать ширины экрана
        mov     BH,AL   ;запомнить номер колонки
        xor     AH,AH   ;обнулить AH
        add     DI,AX
        add     DI,AX
        ; Загрузить атрибут цвета в AH
        mov     AH,BL
@@L1:   ; Загрузить очередной символ строки в AL
        lodsb
        ; Проверка на 0 (на конец строки)
        and     AL,AL
        jz      @@L2
        ; Проверить номер колонки символа
        cmp     BH,79
        ja      @@Error ;нарушена правая граница экрана
        ; Вывести символ на экран
        stosw
        inc     BH      ;увеличить номер колонки
        jmp     @@L1
@@L2:   pop     ES
        pop     DI
        pop     BX
        pop     AX
        ret
@@Error: ;Немедленный выход в DOS по ошибке
        mov     AH,4Ch
        int     21h
ENDP ShowString

;*****************************************************
;*         ВЫВОД ТЕКСТА (ГРУППЫ СТРОК) НА ЭКРАН      *
;* ShowText использует процедуру ShowString для вывода *
;* на экран группы строк.                            *
;* Параметры:                                        *
;* CX - количество строк;                            *
;* DS:SI - адрес первой строки в группе.             *
;* Строки должны иметь заданный для ShowString формат *
;* и располагаться в памяти последовательно.         *
;* При выводе текста используются принятые по        *
;* умолчанию цвет и фон.                             *
        ret
ENDP ShowText

;*****************************************************
;*  ВЫВОД ТЕКСТОВОЙ СТРОКИ ЗАДАННОГО ЦВЕТА НА ЭКРАН  *
;* Все параметры передаются через одну структуру:    *
;* первый байт - атрибут цвета и фона для строки;    *
;* второй байт - номер начальной строки (0-24);      *
;* третий байт - номер начальной колонки (0-79);     *
;* далее идет строка, ограниченная нулем.            *
;* Адрес структуры передается через регистры DS:SI.  *
;*****************************************************
PROC ShowColorString near
        push    AX
        ; Запомнить цвет, используемый по умолчанию
        mov     AL,[TextColorAndBackground]
        push    AX
        ; Установить цвет строки
        cld
        lodsb
        mov     [TextColorAndBackground],AL
        ; Использовать функцию ShowString
        call    ShowString
        ; Восстановить цвет, используемый по умолчанию
        pop     AX
        mov     [TextColorAndBackground],AL
        pop     AX
        ret
ENDP ShowColorString

;*****************************************************
;*    ВЫВОД ЦВЕТНОГО ТЕКСТА (ГРУППЫ СТРОК) НА ЭКРАН  *
;* ShowColorText использует процедуру ShowColorString *
;* для вывода на экран группы разноцветных строк.    *
;* Параметры:                                        *
;* CX - количество строк;                            *
;* DS:SI - адрес первой строки в группе.             *
;* Строки должны иметь заданный для ShowColorString  *
;* формат и располагаться в памяти последовательно.  *
;* При выводе текста используются принятые по        *
;* умолчанию цвет и фон.                             *
;*****************************************************
PROC ShowColorText near
; Цикл вывода строк
@@NextColorString:
        call    ShowColorString
        loop    @@NextColorString
; Процедура не сохраняет значения в CX и SI
        ret
ENDP ShowColorText

;*********************************
;*  УСТАНОВИТЬ ПОЗИЦИЮ КУРСОРА   *
;* Входные параметры:            *
;* ScreenString - номер строки   *
;* ScreenColumn - номер столбца  *
;*********************************
PROC SetCursorPosition NEAR
        pusha
; Вычисление линейного адреса курсора
        mov     AX,[ScreenString]
        mov     BX,80
        mul     BX
        add     AX,[ScreenColumn]
        mov     BL,AL ;запомнить младший байт
; Прямой вывод позиции курсора
; в регистры видеоконтроллера
        mov     DX,3D4h
        ; Вывести старший байт адреса курсора
        mov     AL,0Eh
        out     DX,AX
        ; Вывести младший байт адреса курсора
        inc     AL
        mov     AH,BL
        out     DX,AX
        popa
        ret
ENDP SetCursorPosition

;*****************************************
;*       ПРИНЯТЬ СИМВОЛ ОТ КЛАВИАТУРЫ    *
;* Процедура осуществляет ввод символа с *
;* помощью функции 00h прерывания Int16h. *
;* Для "текстовых" управляющих клавиш вместо *
;* скан-кодов используются ASCII-коды.   *
;* Входных параметров нет.               *
;* Функция возвращает:                   *
;* AL - код символа;                     *
```

```
;* AH - управляющий код, если в AL ноль.     *
;*********************************************
PROC GetChar NEAR
; Очистить буфер клавиатуры
@@ClearBuffer:
        mov     AH,1
        int     16h
        jz      @@WaitChar
        mov     AH,0
        int     16h
        jmp short @@ClearBuffer
; Ожидать нажатия клавиши и принять код символа
@@WaitChar:
        mov     AH,0
        int     16h
; Обработать принятый код
        and     AL,AL
        jnz     @@Get1
        ret     ;(в AL - ноль, в AH - управляющий код)
@@Get1: cmp     AL,32
        jnb     @@Get2
        ; Переписать в AH управляющий код
        xchg    AL,AH
        mov     AL,0
        ret     ;(в AL - ноль, в AH - управляющий код)
@@Get2: mov     AH,0
        ret     ;(в AL - код буквы, в AH - ноль)
ENDP GetChar

;*********************************************
;*        ПРИНЯТЬ СИМВОЛ ОТ КЛАВИАТУРЫ,      *
;*            ЕСЛИ ОН ЕСТЬ В БУФЕРЕ          *
;* Процедура проверяет наличие символа в буфере *
;* клавиатуры и считывает его, если он есть.  *
;* Входных параметров нет.                    *
;* Функция возвращает:                        *
;* AL - код символа;                          *
;* AH - управляющий код, если в AL ноль.      *
;* Если в AL и AH нули - нажатий не было.     *
;*********************************************
PROC WaitChar NEAR
; Проверить наличие символа в буфере клавиатуры
        mov     AH,1
        int     16h
        jz      @@NoInput
; Принять символ от клавиатуры
        mov     AX,0
        int     16h
        and     AL,AL
        jnz     @@GET1
        ret     ;в AL - ноль, в AH - управляющий код
@@GET1: cmp     AL,32
        jnb     @@GET2
        mov     AH,AL ;переписать в AH управляющий код
        mov     AL,0
        ret     ;в AL - ноль, в AH - управляющий код
@@GET2: mov     AH,0
        ret     ;в AL - код буквы, в AH - ноль
@@NoInput:
        xor     AX,AX
        ret     ;в AL и AH - нули
ENDP WaitChar

;************************************
;* ОЧИСТКА ЭКРАНА В ТЕКСТОВОМ РЕЖИМЕ *
;*  (процедура параметров не имеет)  *
;************************************
PROC ClearScreen NEAR
        pusha
        push    ES
; Настроить ES:DI на "текстовую" область видеопамяти
        mov     AX,0B800h
        mov     ES,AX
        cld
        mov     DI,0
; Вывести 2000 "пустых" символов (ASCII-код 0) с
; атрибутом "белый цвет, черный фон"
        mov     CX,2000
        mov     AX,0F00H
        rep     stosw
        pop     ES
        popa
        ret
ENDP ClearScreen

;*****************************************************
;* ПОДАЧА ЗВУКОВОГО СИГНАЛА ЧЕРЕЗ ВСТРОЕННЫЙ ДИНАМИК *
;*          (процедура параметров не имеет)          *
;*****************************************************
PROC Beep NEAR
        push    AX
        push    DX
; Послать на терминал код "звонок" (07h)
        mov     AH,2
        mov     DL,7
        int     21h
        pop     DX
        pop     AX
        ret
ENDP Beep

;*****************************************************
;*      ВЫВОД НА ЭКРАН ТЕКСТОВОГО ПОЛЯ ДАННЫХ        *
;* Передаваемые параметры:                           *
;* DS:SI - указатель на структуру данных;            *
;* BX - смещение поля от начала структуры;           *
;* CX - длина поля в байтах.                         *
;* Цвет задается переменной TextColorAndBackground.  *
;* Координаты позиции передаются через глобальные    *
;* переменные ScreenString и ScreenColumn.           *
;*****************************************************
PROC ShowASCIIField near
        pusha
        push    ES
        mov     AX,0B800h    ;Настроить ES для прямого
        mov     ES,AX        ;вывода на экран
; Установить указатель на начало поля
        add     SI,BX
; Вычислить начальную позицию в видеопамяти
        mov     AX,[ScreenString]
        mov     DI,160
        mul     DI
        add     AX,[ScreenColumn]
        add     AX,[ScreenColumn]
        mov     DI,AX
; Использовать цвет, заданный по умолчанию
        mov     AH,[TextColorAndBackground]
; Вывести поле на экран
@@NextChar:
        lodsb
        stosw
        loop    @@NextChar
        pop     ES
        popa
        ret
ENDP ShowASCIIField

;*****************************************************
;*       ВЫДАЧА СООБЩЕНИЯ О ФАТАЛЬНОЙ ОШИБКЕ        *
;*         И ЭКСТРЕННЫЙ ВЫХОД ИЗ ПРОГРАММЫ           *
;* Параметры:                                        *
;* DS:SI - указатель на строку сообщения об ошибке,  *
;* представленную в формате ShowColorString.         *
;*****************************************************
PROC FatalError near
```

```
; Переустановить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Настроить DS на глобальный сегмент данных
        mov     DI,[CS:MainDataSeg]
        mov     DS,DI
; Вывести сообщение об ошибке красным цветом
        mov     [TextColorAndBackground],12
        call    ShowString
; Переместить курсор в нижнюю часть экрана
        mov     [ScreenString],24
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Аварийный выход из программы
        mov     AH,4Ch
        int     21h
ENDP FatalError
ENDS
```

Приведенная в листинге 1.2 процедура ввода символа GetChar выполняет определенные преобразования над данными, выдаваемыми функцией 00h по прерыванию Int 16h (прерывание Int 16h обеспечивает работу с клавиатурой с использованием функций BIOS). Дело в том, что некоторые из управляющих символов, перечисленных в табл. 1.15, традиционно имеют как ASCII-коды, так и скан-коды, причем значения этих кодов не совпадают. С целью упрощения последующего анализа кодов введенных символов процедура GetChar переносит ASCII-коды символов со значениями от 0 до 20h из регистра AL в регистр AH, заменяя соответствующие скан-коды (AL при этом обнуляется).

Таблица 1.15. Управляющие символы ASCII-кода

Код символа	Мнемоническое обозначение	Назначение символа
00h	NUL	Пустой символ
01h	SOH	Начало заголовка (начало блока данных)
02h	STX	Начало текста
03h	ETX	Конец текста
04h	EOT	Конец передачи
05h	ENQ	Запрос подтверждения
06h	ACK	Подтверждение
07h	BEL	Звонок (звуковой сигнал)
08h	BS	Забой (возврат на одну позицию влево)
09h	HT	Горизонтальная табуляция
0Ah	LF	Перевод строки
0Bh	VT	Вертикальная табуляция
0Ch	FF	Перевод формата (переход к новой странице)
0Dh	CR	Возврат каретки
0Eh	SO	Переход на нижний регистр
0Fh	SI	Переход на верхний регистр
10h	DLE	Завершение сеанса связи
11h	DC1	Управление устройством № 1
12h	DC2	Управление устройством № 2
13h	DC3	Управление устройством № 3
14h	DC4	Управление устройством № 4
15h	NAK	Ошибка передачи
16h	SYN	Холостой ход передатчика
17h	ETB	Конец передачи блока
18h	CAN	Отмена
19h	EM	Конец носителя данных
1Ah	SUB	Подстановка (замена символа)
1Bh	ESC	Переход (посылка сложной команды)
1Ch	FS	Разделитель файлов
1Dh	GS	Разделитель групп
1Eh	RS	Разделитель записей
1Fh	US	Разделитель элементов
7Fh	DEL	Удаление символа

В листинге 1.3. даны мнемонические обозначения для кодов наиболее часто применяемых управляющих клавиш, которые процедура GetChar сохраняет в регистре AH (символ является управляющим, если в регистре AL был возвращен код 0). Следует отметить, что мнемоника, используемая в листинге 1.3 несколько отличается от общепринятой (см. Табл. 1.15). Кроме того, в листинг 1.3. включены мнемонические обозначения цветовых оттенков, которые будут применяться в программах, работающих в 16-цветных и 256-цветных режимах.

Листинг 1.3. Мнемонические обозначения кодов управляющих клавиш и цветовых оттенков (подключаемый файл list1_03.inc).

```
; КОДЫ УПРАВЛЯЮЩИХ КЛАВИШ
; Для "текстовых" управляющих клавиш вместо скан-кодов
; используются ASCII-коды:
B_RUBOUT equ   8 ;забой
B_TAB    equ   9 ;табуляция
B_LF     equ  10 ;перевод строки
B_ENTER  equ  13 ;возврат каретки
B_ESC    equ  27 ;"Esc"
; Скан-коды функциональных клавиш:
F1       equ  59
F2       equ  60
F3       equ  61
F4       equ  62
F5       equ  63
F6       equ  64
F7       equ  65
F8       equ  66
F9       equ  67
F10      equ  68
; Скан-коды клавиш дополнительной клавиатуры:
B_HOME   equ  71 ;перейти в начало
B_UP     equ  72 ;стрелка вверх
B_PGUP   equ  73 ;на страницу вверх
B_BS     equ  75 ;стрелка влево
B_FWD    equ  77 ;стрелка вправо
B_END    equ  79 ;перейти в конец
B_DN     equ  80 ;стрелка вниз
B_PGDN   equ  81 ;на страницу вниз
B_INS    equ  82 ;переключить режим (вставка/замещение)
B_DEL    equ  83 ;удалить символ над курсором

; МНЕМОНИЧЕСКИЕ ОБОЗНАЧЕНИЯ ЦВЕТОВ
; "Темные" цвета (можно использовать для фона и текста)
BLACK        equ  0 ;черный
BLUE         equ  1 ;темно-синий
GREEN        equ  2 ;темно-зеленый
CYAN         equ  3 ;бирюзовый (циан)
RED          equ  4 ;темно-красный
MAGENTA      equ  5 ;темно-фиолетовый
BROWN        equ  6 ;коричневый
LIGHTGREY    equ  7 ;серый
; "Светлые" цвета (только для текста)
DARKGREY     equ  8 ;темно-серый
LIGHTBLUE    equ  9 ;синий
LIGHTGREEN   equ 10 ;зеленый
LIGHTCYAN    equ 11 ;голубой
LIGHTRED     equ 12 ;красный
LIGHTMAGENTA equ 13 ;фиолетовый
YELLOW       equ 14 ;желтый
WHITE        equ 15 ;белый
```

Листинг 1.4 содержит набор макрокоманд, делающих более наглядными и компактными участки программного кода, в которых осуществляются операции ввода-вывода.

Листинг 1.4. Макрокоманды для вывода данных на экран (подключаемый файл list1_04.inc)

```
; МАКРОКОМАНДЫ ТЕКСТОВОГО РЕЖИМА
;*************************************
;*    ВЫВОД БАЙТА В ДВОИЧНОМ КОДЕ    *
;* Параметры:                        *
;* SString - номер строки экрана;    *
;* SColumn - номер колонки экрана;   *
;* BData - отображаемый байт данных. *
;*************************************
MACRO MShowBinByte SString,SColumn,BData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AL,BData
        call    ShowBinByte
ENDM

;*********************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА В ДВОИЧНОМ КОДЕ *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* WData - отображаемое слово данных.        *
;*********************************************
MACRO MShowBinWord SString,SColumn,WData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AX,WData
        call    ShowBinWord
ENDM
;*********************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА В ДВОИЧНОМ КОДЕ *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* DData - отображаемое слово данных.        *
;*********************************************
MACRO MShowBinDWord SString,SColumn,DData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     EAX,DData
        call    ShowBinDWord
ENDM

;*************************************
;* ВЫВОД БАЙТА В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ *
;* Параметры:                        *
;* SString - номер строки экрана;    *
;* SColumn - номер колонки экрана;   *
;* BData - отображаемый байт данных. *
;*************************************
MACRO MShowHexByte SString,SColumn,BData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AL,BData
        call    ShowHexByte
ENDM

;*****************************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ *
;* Параметры:                                        *
;* SString - номер строки экрана;                    *
;* SColumn - номер колонки экрана;                   *
;* WData - отображаемое слово данных.                *
;*****************************************************
MACRO MShowHexWord SString,SColumn,WData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AX,WData
        call    ShowHexWord
ENDM

;*****************************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА В ШЕСТНАДЦАТЕРИЧНОМ КОДЕ *
;* Параметры:                                        *
;* SString - номер строки экрана;                    *
;*****************************************************
MACRO MShowHexDWord SString,SColumn,DData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     EAX,DData
        call    ShowHexDWord
ENDM
```

```
;************************************
;*   ВЫВОД БАЙТА В ДЕСЯТИЧНОМ КОДЕ   *
;* Параметры:                       *
;* SString - номер строки экрана;   *
;* SColumn - номер колонки экрана;  *
;* BData - отображаемый байт данных. *
;************************************
MACRO MShowDecByte SString,SColumn,BData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AL,BData
        call    ShowDecByte
ENDM
;*********************************************
;* ВЫВОД 16-РАЗРЯДНОГО СЛОВА В ДЕСЯТИЧНОМ КОДЕ *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* WData - отображаемое слово данных.        *
;*********************************************
MACRO MShowDecWord SString,SColumn,WData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     AX,WData
        call    ShowDecWord
ENDM
;*********************************************
;* ВЫВОД 32-РАЗРЯДНОГО СЛОВА В ДЕСЯТИЧНОМ КОДЕ *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* DData - отображаемое слово данных.        *
;*********************************************
MACRO MShowDecDWord SString,SColumn,DData
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     EAX,DData
        call    ShowDecDWord
ENDM
;*********************************************
;*        ВЫВОД ТЕКСТОВОГО ПОЛЯ ДАННЫХ       *
;* Параметры:                                *
;* SString - номер строки экрана;            *
;* SColumn - номер колонки экрана;           *
;* DOffs - смещение поля от начала структуры; *
;* DSize - длина поля в байтах;              *
;* DS:SI - указатель на структуру данных.    *
;*********************************************
MACRO MShowASCIIField SString,SColumn,DOffs,DSize
        mov     [ScreenString],SString
        mov     [ScreenColumn],SColumn
        mov     BX,DOffs
        mov     CX,DSize
        call    ShowASCIIField
ENDM
;******************************
;*   ВЫВОД ТЕКСТОВОЙ СТРОКИ   *
;* Параметры:                 *
;* TOffset - смещение строки.  *
;******************************
MACRO MShowString TOffset
        mov     SI,offset TOffset
        call    ShowString
ENDM
;**********************************
;* ВЫВОД ЦВЕТНОЙ ТЕКСТОВОЙ СТРОКИ *
;* Параметры:                     *
;* TOffset - смещение строки.     *
;**********************************
MACRO MShowColorString TOffset
        mov     SI,offset TOffset
        call    ShowColorString
ENDM
;*****************************************
;*        ВЫВОД ОДНОЦВЕТНОГО ТЕКСТА      *
;* Параметры:                            *
;* TStrings - количество строк в тексте;  *
;* TOffset - смещение первой строки текста. *
;*****************************************
MACRO MShowText TStrings,TOffset
        mov     CX,TStrings
        mov     SI,offset TOffset
        call    ShowText
ENDM
;*****************************************
;*        ВЫВОД РАЗНОЦВЕТНОГО ТЕКСТА     *
;* Параметры:                            *
;* TStrings - количество строк в тексте;  *
;* TOffset - смещение первой строки текста. *
;*****************************************
MACRO MShowColorText TStrings,TOffset
        mov     CX,TStrings
        mov     SI,offset TOffset
        call    ShowColorText
ENDM
;***************************************
;*       ВЫВОД СООБЩЕНИЯ ОБ ОШИБКЕ     *
;* Параметры:                          *
;* TOffset - смещение строки сообщения. *
;***************************************
MACRO MFatalError TOffset
        mov     SI,offset TOffset
        call    FatalError
ENDM
; МАКРОКОМАНДЫ ГРАФИЧЕСКОГО РЕЖИМА
;******************************
;*   ВЫВОД ТЕКСТОВОЙ СТРОКИ   *
;* Параметры:                 *
;* TOffset - смещение строки. *
;******************************
MACRO MGShowString TOffset
        mov     SI,offset TOffset
        call    GShowString
ENDM
;*****************************************
;*        ВЫВОД ОДНОЦВЕТНОГО ТЕКСТА      *
;* Параметры:                            *
;* TStrings - количество строк в тексте;  *
;* TOffset - смещение первой строки текста. *
;*****************************************
MACRO MGShowText TStrings,TOffset
        mov     CX,TStrings
        mov     SI,offset TOffset
        call    GShowText
ENDM
```

1.9.3. Программирование EPP-порта

Порот EPP (Enhanced Parallel Port — улучшенный параллельный порт) был разработан компаниями Intel, Xircom и Zenith Data Systems задолго до принятия стандарта IEEE 1284. Этот порт был предназначен для повышения производительности обмена по параллельному интерфейсу со сканерами и внешними дисководами. Системная поддержка EPP-порта в виде EPP BIOS появилась к сожалению значительно позже и до сих пор не поддерживается многими производителями системных плат. Общие вопросы работы параллельного EPP-порта уже были рассмотрены в разделе 1.3.3. В данном разделе более подробно рассмотрены вопросы, связанные с работой EPP-порта и его программированием.

1.9.3.1. Протокол и циклы работы EPP-порта.

Как упоминалось в разделе 1.3.3, протокол EPP обеспечивает четыре типа циклов обмена: *запись данных; чтение данных; запись адреса; чтение адреса.* Адресные циклы используются для передачи адресной, канальной и управляющей информации. Циклы обмена данными отличаются от адресных циклов применяемыми стробирующими сигналами. Как и в режиме ECP внешние сигналы EPP-порта для каждого цикла обмена формируются аппаратно по одной операции записи или чтения в регистр порта. Однако в отличие от ECP порта, в котором процессор ведет обмен с буфером порта, в EPP режиме при обращении к регистру данных или адреса EPP-порта формируется внешний цикл обмена, *вложенный* в цикл обмена системной шины процессора (иногда эти циклы называют *связанными).* Например, цикл записи данных состоит из следующих фаз.

1. Программа выполняет цикл вывода (IOWR#) в порт Base+4 (EPP Data Port).
2. Контроллер EPP-порта устанавливает сигнал Write# (низкий уровень), идентифицируя режим передачи по линиям интерфейса и данные помещаются на выходную шину LPT-порта.
3. При низком уровне Wait# контроллером устанавливается строб данных.
4. Порт ждет подтверждения от ПУ (перевода линии Wait# в высокий уровень).
5. По положительному сигналу Wait# контроллер снимает строб данных и внешний EPP-цикл завершается.
6. Завершается процессорный цикл вывода.
7. ПУ устанавливает низкий уровень Wait#, указывая на возможность начала следующего цикла.

ПРИМЕЧАНИЕ: В описании цикла используется обозначение сигналов в соответствии с табл. 1.5 раздела 1.3.3.

Такой протокол блокированного квитирования *(interlocked handshakes)* позволяет автоматически настраиваться на скорость обмена, доступную и хосту, и ПУ. ПУ может регулировать длительность всех фаз обмена с помощью всего лишь одного сигнала Wait#, что позволяет достигать высоких скоростей обмена (0,5-2 Мбайт/с) и ПУ может, в принципе, работать со скоростью устройства, подключаемого через слот ISA.

Естественно, ПУ не должно «подвешивать» процессор на шинном цикле обмена. Это гарантирует механизм тайм-аутов PC, который принудительно завершает любой цикл обмена, длящийся более 15 мкс. В ряде реализаций EPP за тайм-аутом интерфейса следит сам адаптер — если ПУ не отвечает в течение определенного времени (5 мкс), цикл прекращается и в дополнительном (не стандартизованном) регистре состояния адаптера фиксируется ошибка.

Устройства с интерфейсом EPP, разработанные до принятия IEEE 1284, отличаются началом цикла: строб DataStb# или AddrStb# устанавливается независимо от состояния WAIT#. Это означает, что ПУ не может задержать начало следующего цикла (хотя может растянуть его на требуемое время). Такая спецификация называется EPP 1.7 (предложена Xircom). Именно она применялась в контроллере 82360. Периферия, совместимая с IEEE 1284 EPP, будет нормально работать с контроллером EPP 1.7, но ПУ в стандарте EPP 1.7 может отказаться работать с контроллером EPP 1284.

С *программной* точки зрения контроллер EPP-порта выглядит просто (см. табл. 6 раздела 1.3.3). К трем регистрам стандартного порта, имеющим смещение 0, 1 и 2 относительно базового адреса порта, добавлены два регистра (EPP Address Port и EPP Data Port), чтение и запись в которые вызывает генерацию связанных внешних циклов.

Назначение регистров стандартного порта сохранено для совместимости EPP-порта с ПУ и ПО, рассчитанными на применение программно-управляемого обмена. Поскольку сигналы квитирования адаптером вырабатываются аппаратно, при записи в регистр управления CR биты 0, 1 и 3, соответствующие сигналам Strobe#, AutoFeed# и SelectIn# должны иметь нулевые значения. Программное вмешательство могло бы нарушить последовательность квитирования. Некоторые адаптеры имеют специальные средства защиты *(EPP Protect),* при включении которых программная модификация этих бит блокируется. Использование регистра данных EPP позволяет осуществлять передачу блока данных с помощью одной инструкции REP INSB или REP OUTSB. Некоторые адаптеры допускают *16/32-битное обращение к* регистру данных EPP. При этом адаптер просто дешифрует адрес со смещением в диапазоне 4-7 как адрес регистра данных EPP, но процессору сообщает о разрядности 8 бит. Тогда 16- или 32-битное обращение по адресу регистра данных EPP приведет к автоматической генерации двух или четырех шинных циклов по нарастающим адресам, начиная со смещения 4. Эти циклы будут выполняться быстрее, чем то же количество одиночных циклов. Более «продвинутые» адаптеры для адреса регистра данных EPP сообщают разрядность 32 бит и для них до 4 байт может быть передано за один цикл обращения процессора. Таким образом обеспечивается производительность до 2 Мбайт/с, достаточная для адаптеров локальных сетей, внешних дисков, стримеров и CD-ROM. Адресные циклы EPP всегда выполняются только в однобайтном режиме.

Важной чертой EPP является то, что обращение процессора к ПУ осуществляется в реальном времени — нет буферизации. Драйвер способен отслеживать состояние и подавать команды в точно известные моменты времени. Циклы чтения и записи могут чередоваться в произвольном порядке или идти блоками. Такой тип обмена удобен для *регистро-ориентированных ПУ* или ПУ, работающих в *реальном времени,* например устройств сбора информации и управления. Этот режим пригоден и для устройств хранения данных, сетевых адаптеров, принтеров, сканеров и т. п.

К сожалению, режим EPP поддерживается не всеми портами — он отсутствует, к примеру, в ряде блокнотных ПК. Так что при разработке собственных устройств ради большей совместимости с компьютерами приходится ориентироваться на режим ECP.

1.9.3.2. Программирование EPP-порта на уровне регистров

В качестве примера программирования EPP-порта приведем фрагмент программы, реализующей выдачу на линии интерфеса LPT1 константы AAh через регистры EPP Data Port и EPP Address Port. Предполагается, что предварительно средствами программы Setup BIOS контроллер параллельного порта был установлен в режим ECP+EPP, что позволит программно переключать режимы работы контроллера порта через регистр ECR - главный управляющий регистр порта ECP. Программа сориентирована на выполнение ее в среде полноэкранного отладчика реального режима (типа AFD). В связи с этим в операторах переходов метки необходимо заменять конкретными адресами команд, отмеченных в программе этими метками.

Следует также отметить, что, поскольку сигнальный протокол EPP порта реализуется на аппаратном уровне, то для реализации передачи данных и адресов через EPP-порт к его выходу необходимо подключить устройство (или макет устройства) поддерживающее этот сигнальный протокол.

Пример фрагмента программы

```
        mov   dx,077A     ;Установка режима Bi-directional через ECR - глав-
        mov   al,34       ;ный управляющий регистр ECP
        out   dx,al
        mov   dx,037A     ;Установка цикла записи, переводом линии
        mov   al,01       ;Write# (контакт 1) в низкий уровень.
        out   dx,al
        mov   dx,077A     ;Установка EPP режима записью в регистр ECR
        mov   al,94       ;константы 94
        out   dx,al
A:      mov   dx,037C     ;Через регистр данных EPP выдается на линии
        mov   al,AA       ;AD [0:7] (контакты 2-9) константа AAh
        out   dx,al
        mov   cx,000F     ;Программная задержка
B:      loop  B
        mov   dx,037B     ;Через регистр адреса EPP выдается на линии
        out   dx,al       ;AD [0:7] (контакты 2-9) константа AAh
        mov   cx,000F     ;Программная задержка
C:      loop  C
        jmp   A           ;Зацикливание программы (выход из бесконечного
                          ;цикла по комбинации клавиш Ctrl+Esc)
```

1.9.3.3. Программирование EPP-порта на уровне EPP BIOS.

К сожалению, появился EPP BIOS со значительным опозданием и до сих пор не поддерживается многими изготовителями системных плат. Кроме того, этот набор функций, как явствует из его названия, ориентирован в основном на использование режима EPP, который практически не поддерживается изготовителями принтеров. При работе с принтерами реальную пользу могут принести только головная функция - функция 02h (проверка наличия EPP BIOS, см. раздел 1.4.4). В случае если EPP BIOS поддерживается системой, функция возвращает вектор (“точку входа”) для вызова всех остальных функций EPP BIOS. Функция установки режима представляет особый интерес, так как позволяет задать нужный режим работы контроллера прямо из прикладной программы (до появления EPP BIOS операцию выбора режима можно было осуществить только в процессе начальной загрузки компьютера, через BIOS Setup).

Все остальные функции EPP BIOS при работе с принтером могут пригодиться только в том случае, если принтер подключен к компьютеру через автоматический мультиплексор. В России, однако, такие мультиплексоры используются крайне редко вследствие относительного дефицита периферийного оборудования (на один системный блок редко приходится более одного принтера, а со сканером и внешним дисководом принтер обычно объединяют в цепочку, без мультиплексора).

В листинге 1.5 приведен пример программы, которая использует функцию установки режима работы EPP BIOS для переключения контроллера параллельного порта в режим ECP.

Листинг 1.5. Программа для проверки наличия EPP BIOS и переключения порта LPT1 в режим ECP

```
IDEAL
P386
LOCALS
MODEL MEDIUM
                                  ; Подключить файл мнемонических обозначений
                                  ; кодов управляющих клавиш и цветовых кодов
                                  include "list1_03.inc"
                                  ; Подключить файл макросов
                                  include "list1_04.inc"
```

```
DATASEG
; Вектор точки входа EPP BIOS
label EPP_Vector DWORD
EPP_Offset      DW ?
EPP_Segment     DW ?
; Конфигурация порта EPP
InterruptLevel  DB ?
BIOS_Revision   DB ?
IO_Capabilities DB ?
IO_BaseAddress  DW ?
label EPP_Manufacturer DWORD
Manuf_Offset    DW ?
Manuf_Segment   DW ?
; Режим работы параллельного порта
OperationMode   DB ?
; Текстовые сообщения
AnyK DB YELLOW,24,29,"Нажмите любую клавишу",0
Txt1 DB LIGHTCYAN,0,28,"ТЕСТИРОВАНИЕ EPP BIOS",0
Txt2 DB 2,23,"Вектор EPP BIOS:     :",0

     DB 3,8,"Номер используемого прерывания:",0
     DB 4,17,"Номер версии EPP BIOS:",0
     DB 5,14,"Возможности ввода-вывода:",0
     DB 6,3,"Базовый адрес блока регистров порта:",0
     DB 8,0,"Разработчик BIOS:",0
ECPY DB LIGHTCYAN,12,26,"Порт переключен в режим ECP",0
ECPN DB LIGHTRED,12,22,"Режим ECP не поддерживается портом",0
Err1 DB 12,22,"EPP BIOS не поддерживается системой",0
ENDS

SEGMENT sseg para stack 'STACK'
        DB 400h DUP(?)
ENDS

CODESEG
;*****************************
;* Основной модуль программы *
;*****************************
PROC EPP_BIOS_Test
        mov     AX,DGROUP
        mov     DS,AX
        mov     [CS:MainDataSeg],AX
; Установить текстовый режим и очистить экран
        mov     AX,3
        int     10h
; Скрыть курсор - убрать за нижнюю границу экрана
        mov     [ScreenString],25
        mov     [ScreenColumn],0
        call    SetCursorPosition
; Вывести текстовые сообщения на экран
        mov     SI,offset Txt1
        call    ShowColorString
        mov     SI,offset AnyK
        call    ShowColorString
; Инициализация EPP
        mov     AH,2
        mov     DX,0
        mov     AL,0
        mov     CH,'E'
        mov     BX,"PP"
        int     17h
; Проверка наличия EPP BIOS
        or      AH,AH
        jnz     @@No_EPP
        cmp     AL,'E'
        jne     @@No_EPP
        cmp     CX,"PP"
        jne     @@No_EPP
; Сохранение вектора точки входа EPP BIOS
        mov     [EPP_Offset],BX
        mov     [EPP_Segment],DX
; Вывести наименования параметров
        mov     [TextColorAndBackground],LIGHTGREEN
        MShowText 6,Txt2
        mov     [TextColorAndBackground],WHITE
; Вывести вектор точки входа EPP BIOS на экран
        MShowHexWord 2,40,[EPP_Segment]
        MShowHexWord 2,45,[EPP_Offset]
; Определить конфигурацию порта EPP
        push    ES
        mov     AH,0
        mov     DL,0
        call    [Epp_Vector]
; Сохранить параметры настройки порта
        mov     [InterruptLevel],AL
        mov     [BIOS_Revision],BH
        mov     [IO_Capabilities],BL
        mov     [IO_BaseAddress],CX
        mov     [Manuf_Offset],DI
        mov     AX,ES
        mov     [Manuf_Segment],AX
        pop     ES
; Вывести значения параметров
        MShowHexByte 3,40,[InterruptLevel]
        MShowHexByte 4,40,[BIOS_Revision]
        MShowBinByte 5,40,[IO_Capabilities]
        MShowHexWord 6,40,[IO_BaseAddress]
        ; Вывести имя разработчика BIOS
        pusha
        push    DS
        push    ES
        mov     AX,0B800h
        mov     ES,AX
        ; Задать позицию строки в видеопамяти
        mov     DI,160*8 + 18*2
        ; Использовать цвет, заданный по умолчанию
        mov     AH,[TextColorAndBackground]
        ; Установить указатель на начало ASCIIZ-строки
        mov     SI,[Manuf_Offset]
        mov     BX,[Manuf_Segment]
        mov     DS,BX
        mov     CX,62 ;ограничитель длины строки
@@NextChar:
        lodsb
        and     AL,AL ;конец строки?
        jz      @@Zero
        stosw
        loop    @@NextChar
@@Zero: pop     ES

        pop     DS
        popa

; Переключить порт в режим ECP
        mov     AH,1
        mov     DL,0
        mov     AL,1000b
        call    [Epp_Vector]
        ; Операция выполнена?
        cmp     AH,0
        jne     @@ECP_Not_Supported
        MShowColorString ECPY ;режим ECP установлен
        jmp     @@End
@@ECP_Not_Supported:
        MShowColorString ECPN ;режим ECP не установлен

@@End:  call    GetChar
; Переустановить текстовый режим
        mov     ax,3
        int     10h
; Выход в DOS
        mov     AH,4Ch
        int     21h

; Сообщения об ошибках
@@No_EPP:
        MFatalError Err1 ; отсутствует EPP BIOS
ENDP EPP_BIOS_Test
ENDS
```

```
; Подключить процедуры вывода данных на экран
include "list1_02.inc"
```

```
END
```

Программа использует универсальные процедуры ввода-вывода из раздела 1.9.2.6 (подключаемые файлы "list1_03.inc", "list1_04.inc" и "list1_02.inc")

Контрольные вопросы

1. сколько и каких регистров содержит LPT порт в режиме совместимости?
2. В каких случаях используется программирование LPT порта на уровне регистров?
3. Как устанавливаются расширенные режимы LPT порта в PS/2 и в PC совместимых компьютерах?
4. Каковы особенности программирования SPP порта?
5. Охарактеризуйте программирование SPP порта на уровне регистров.
6. Охарактеризуйте программирование SPP порта на уровне BIOS.
7. Дайте общую характеристику ECP порта.
8. Охарактеризуйте поддержку ECP порта на уровне BIOS.
9. Перечислите регистры ECP порта, охарактеризуйте их адресацию и и режимы, в которых они применяются.
10. Охарактеризуйте регистры ECP порта со смещением адреса 400h и 401h.
11. Охарактеризуйте дополнительный регистр управления ECR.
12. В каких случаях контроллер ECP порта генерирует аппаратные прерывания?
13. Как осуществляется переключение режимов работы контроллера ECP порта и направление передачи через порт?
14. Когда необходимо выполнять процедуру переговоров?
15. В каком режиме работы порта выполняется процедура переговоров?
16. Какой константой инициализируется регистр Dcr после проведения процедуры переговоров?
17. Какие этапы процедуры переговоров необходимо выполнить, что бы переключить внешнее устройство на прием данных в режиме ECP?
18. Какие этапы процедуры переговоров необходимо выполнить, что бы переключить внешнее устройство из режима ECП в режим совместимости?
19. Какие три способа передачи данных можно реализовать в режиме ECP?
20. Охарактеризуйте варианты программной передачи в режиме ECP.
21. Охарактеризуйте режим передачи по прерыванию через ECP порт.
22. Охарактеризуйте обмен данными через ECP порт в режиме DMA.
23. Охарактеризуйте общие положения переключения направления передачи данных через ECP порт.
24. Какие операции необходимо выполнить при переключении передачи с прямой на инверсную?
25. Какие операции необходимо выполнить при переключении передачи с инверсной на прямую?
26. Охарактеризуйте программу, приведенную в листинге 1.1.
27. Кратко охарактеризуйте EPP порт.
28. Охарактеризуйте фазы цикла вывода через EPP порт.
29. С помощью какого сигнала внешнее устройство может регулировать скорость обмена через EPP порот?
30. Охарактеризуйте механизм тайм-аутов.
31. В чем заключается отличие устройств стандарта EPP 1.7 от EPP устройств, поддерживающих стандарт IEEE 1284?
32. Что понимается под EPP Protect?
33. Как организуется 16 и 32 битные циклы обмена через EPP порт?
34. Охарактеризуйте программирование EPP порта на уровне регистров.
35. Охарактеризуйте программирование EPP порта на уровне BIOS.

В данном разделе использованы материалы из [3, 5-9].

2. ПРАКТИКУМ "Программирование параллельного порта ПК"

Практикум включает в себя пять лабораторных работ, посвященных различным аспектам программного управления обменом информацией через параллельный порт ПК, работающий в основных режимах восьмиразрядного обмена: SPP, Byte Mode (Bi directional), EPP и ECP.

Для изучения принципов программирования портов EPP и ECP с аппаратной реализацией протокола интерфейса в работах используются макеты устройств, поддерживающих эти протоколы или порты других ПК с аппаратной или программной реализацией этих протоколов. Подсоединение портов к макетам и другим ПК осуществляется через кабельные соединения, обеспечивающие их корректное взаимодействие. При этом обеспечивается доступ ко всем линиям интерфейса для контроля их состояния с помощью измерительных устройств (например, осциллографа). Корректное соединение и доступ к линиям интерфейса осуществляется с помощью дополнительных разъемов, установленных на макетах устройств или кроссовых платах.

Фрагменты программ, реализующих алгоритмы и протоколы обмена, пишутся в мнемокодах Ассемблера, а отлаживаются и выполняются в среде полноэкранного отладчика реального режима типа AFD (SDT) (см. пункты 3.1.и 3.2 приложения) или отладчика DEBUG (см. пункт 3.3 приложения). Эти отладчики имеют режимы ассемблера и дизассемблера, а также набор команд, позволяющих набирать, отлаживать и выполнять эти фрагменты программ прямо в своей среде. Как правило, они работают в среде DOS, однако с некоторыми ограничениями они могут функционировать и в окне сеанса MS-DOS операционных систем типа Windows 95/98.

2.1. Макеты устройств, кроссовые платы и кабельные соединения.

Для исследования работы параллельного порта в стандартном режиме SPP (см. раздел 1.1) и в двунаправленном байтовом режиме Bait Mode (см. раздел 1.3.2) с использованием программирования порта на уровне регистров его контроллера можно не применять дополнительных макетов устройств. В этом случае можно ограничиться разъемом-заглушкой с защитными резисторами и с соответствующим набором подсоединенных к нему проводников. В этих режимах протокол работы порта полностью определяется той программой, которая будет разработана для реализации портом тех или иных действий по управлению состоянием управляющих линий, чтению сигналов с линий состояния и передаче или приему данных по линиям данных интерфейса.

При использование в программах системной поддержки стандартного порта на уровне BIOS (функции программного прерывания INT 17h - драйвер принтера) необходимо обеспечить аппаратное или программное управление состоянием линии Busy интерфейса. Это связано с тем, что BIOS во время выполнения функций прерывания INT 17h контролирует состояние линии Busy, и если она остается в высоком состоянии в течение времени, предусмотренного функцией тайм-аута порта, происходит аварийное завершение выполнения соответствующей функции INT 17h (см. разделы 1.4.1 - 1.4.4).

При исследовании параллельного порта в режимах EPP и ECP, в которых протокол обмена реализуется на аппаратном уровне, необходимо применять макеты, поддерживающие этот протокол на аппаратном уровне, или подключать через соответствующие кабели и кроссовые платы параллельные порты других компьютеров с аппаратной или программной поддержкой этих протоколов. При этом необходимо учитывать, что интерфейсы и принципы работы EPP и ECP портов значительно отличаются друг от друга (см. разделы 1.3.3 и 1.3.4).

2.1.1. Макеты устройств

На рис. 2.1 представлена схема транзисторного варианта макета устройства, реализующего функцию квитирования для параллельного порта, работающего в режиме передачи данных. Обозначение сигналов соответствует порту SPP (см. табл. 1.1 раздела 1.1). Для режимов EPP и ECP обозначение сигналов иное (см. табл.1.5 и 1.7 в разделах 1.3.3 и 1.3.4). Питание макета осуществляется по линии Init#. (Символ # в обозначении сигнала указывает на то, что сигнал в активном состоянии имеет низкий уровень напряжения). Поэтому в программе необходимо предусмотреть установку на выходе Init# потенциала высокого уровня и поддержания его во время проведения исследования работы порта. Квитирующий сигнал Busy вырабатывается по переднему фронту сигнала Strobe#. Задержка сигнала Busy осуществляется интегрирующей RC-цепочкой, подключенной к базе транзистора. Задержка имитирует инерционность устройства, связанную со временем анализа устройством состояния управляющих линий интерфейса и выполнением действий по считыванию данных с шины данных интерфейса.

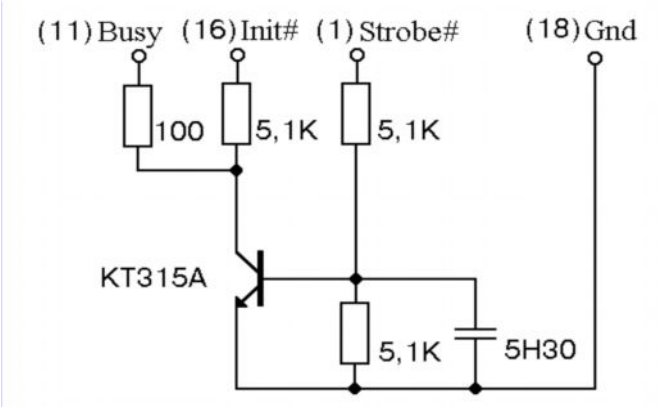


Рис. 2.1. Транзисторная схема макета, обрабатывающая один стробирующий сигнал.

На рис. 2.2 представлена схема транзисторного варианта макета устройства, реализующего функции квитирования для порта EPP, работающего в режиме передачи и приема данных и адреса. Тип информации определяется соответствующим стробирующим сигналом: DataStb# и AddrStb# соответственно (обозначение сигналов соответствует табл. 1.5 раздела 1.3.3). Поскольку на оба стробирующих сигнала вырабатывается один и тот же квитирующий сигнал Wite#, то этот макет может выполнять функции устройства, представленного на рис. 2.1. Питание макета производится по линии Reset# интерфейса. Поэтому в программе необходимо предусмотреть установку на выходе Reset# потенциала высокого уровня и поддержания его во время проведения исследования работы порта.

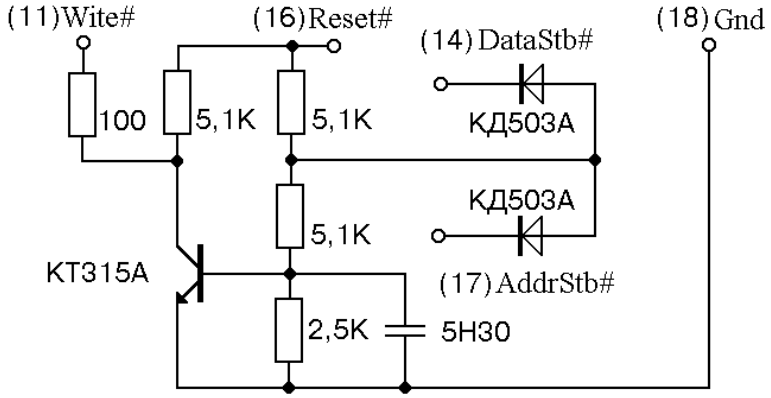


Рис. 2.2. Транзисторная схема, обрабатывающая два стробирующих сигнала

Для получения более крутых фронтов квитирующих сигналов можно использовать макет, выполненный на интегральной логической микросхеме типа триггера Шмидта (напри-

мер, К155ТЛ1). Схема такого макета представлена на рис. 2.3. Для этой схемы необходимо использовать дополнительную линию питания +5В, которая может поступать от источника питания ПК. Для развязки цепи питания по низким и высоким частотам рекомендуется также использование накопительной емкости С4 и высокочастотного конденсатора С3. (см. рис. 2.3). Формирование задержки квитирующего сигнала "Подтверждение" в этой схеме осуществляется отдельно для адресного строба и строба данных: соответственно цепочками R2C2 и R1C1.

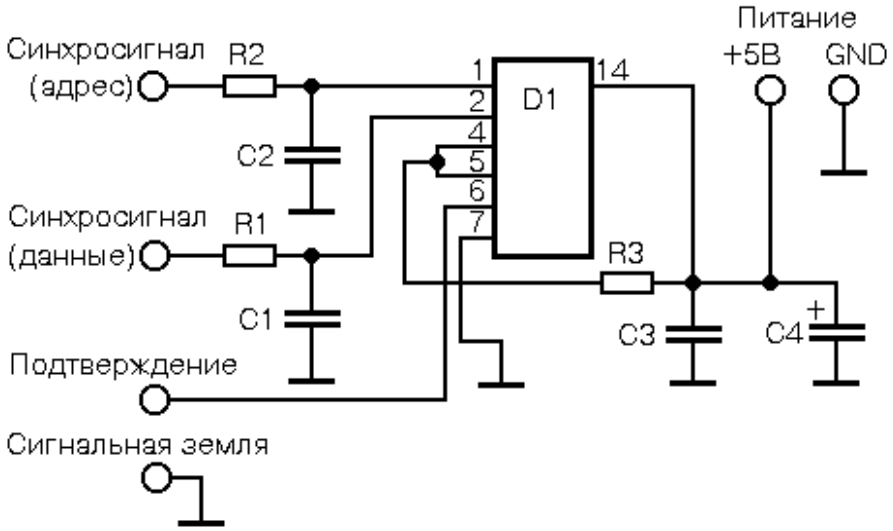


Рис. 2.3. Схема макета на триггере Шмидта, обрабатывающая два стробирующий сигнал: R1=270 Ом, R2=270 Ом, R3=5,1 кОм; C1=н10, C2=н30, C3=5н90, C4=100 мкФ, D1 – К155ТЛ1.

Более сложные макеты могут генерировать не только квитирующие сигналы, но и имитировать как прием, так и передачу в ПК данных и другой информации (команды адреса и т.п.).

На рис. 2.4 представлена схема макета, поддерживающего обмен с параллельным портом ПК, работающего в режиме EPP.

Данный макет позволяет имитировать как прием информации со стороны EPP-порта, так и передачу в порт содержимого счетчика D3 макета. Направление обменом между EPP-портом и макетом определяется состоянием сигнала Write# (обозначение сигналов см. в табл.1.5 раздела 1.3.3). При низком уровне сигнала Write# данные и адреса передаются из EPP-порта в макет, а при высоком - в обратном направлении.

При работе EPP-порта в режиме приема информации содержимое счетчика макета меняется после получения со стороны EPP-порта стробирующих сигналов адреса или данных (AddrStb# или DataStb# соответственно) с задержкой, формируемой цепочкой R2C1. Для разрешения работы счетчика необходимо сформировать высокий уровень на линии Reset# (контакт 16) записью "1" в разряд CR.2 регистра управления SPP CR. По сигналу DataStb# содержимое четырехразрядного счетчика D3 передается в EPP-порт по линиям D0 - D3 интерфейса через буферное устройство D4.1 с тремя состояниями на выходе. По сигналу AddrStb# содержимое счетчика D3 передается в EPP-порт по линиям D4 - D7 интерфейса через буферное устройство D5.1 с тремя состояниями на выходе.

Сигнал Intr# - прерывание от ПУ, формируется триггером Шмидта D1.1 без задержки, а квитирующий сигнал Wite# формируется макетом с задержкой, определяемой постоянной времени цепочки R2C1. Эта задержка позволяет макету выдать информацию на линии D0 - D7 по передним фронтам сигналов AddrStb# или DataStb#, поступающими через D4.2 на разрешающие входы E1 буферных устройств D4.1 и D5.1.

Содержимое счетчика D3 можно сбрасывать по сигналу Reset#, поступающему со стороны EPP-порта ПК.

При работе EPP-порта в режиме передачи информации буферные устройства D4.1 и D5.1 макета переводятся в высокоомное состояние (в третье, в высокоимпедансное, в Z-состояние) низким уровнем сигнала Write#. В этом режиме макет формирует только квитирующий сигнал Wite# и сигнал прерывания Intr# в ответ на стробсигналы AddrStb# и DataStb# EPP-порта.

Питающее напряжение +5В подается на макет по дополнительной линии от источника питания ПК. На плате макета необходимо предусмотреть развязку по питанию в области низких и высоких частот.

Подробности работы EPP-порта представлены в разделе 1.3.3.

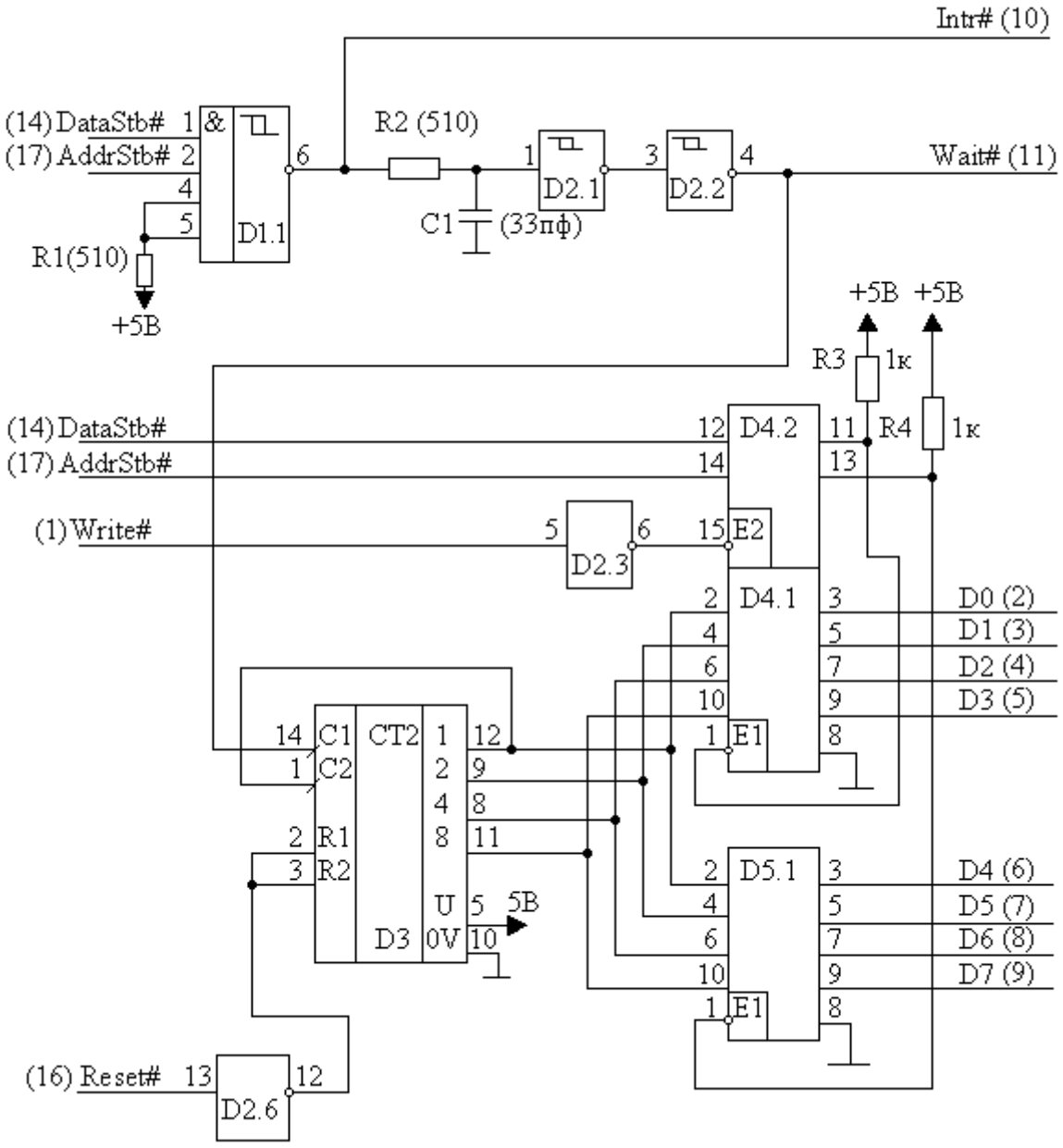


Рис. 2.4. Макет устройства, поддерживающего обмен с параллельным портом, работающим в режиме EPP. Используемые микросхемы: D1 – К155ТЛ1; D2 – К155ТЛ2; D3 – К155ИЕ5; D4, B5 – К155ЛП11; числа, которыми помечены интерфейсные сигналы, определяют номера контактов разъема EPP-порта.

На рис. 2.5 представлена схема макета, поддерживающего обмен с параллельным портом ПК, работающего в режиме ECP.

Данный макет позволяет имитировать как прием информации со стороны параллельного порта ПК, работающего в режиме ECP, так и передачу в компьютер содержимого

счетчика макета. Содержимое счетчика меняется при получении от порта сигнала подтверждения приема информации (сигнал HostAck#). Содержимое счетчика D3 выдается в линии интерфейса через буферные устройства D4.1 и D5.1, которые в режиме приема переводятся по выходу в высокоомное состояние. Переключение в высокоомное состояние производится сигналом ReversReq#, поступающим со стороны контроллера параллельного порта при его переключении в режим передачи информации.

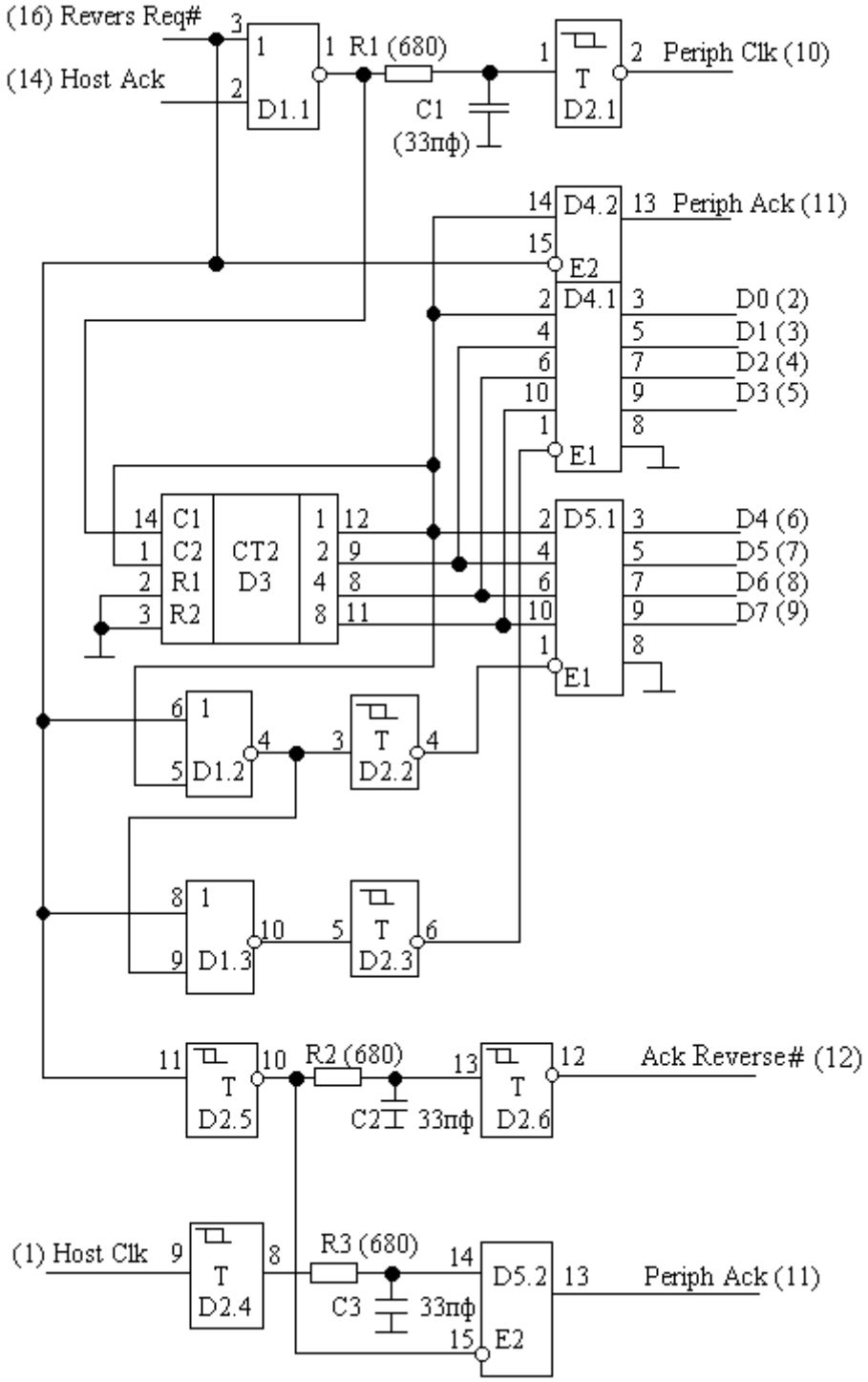


Рис. 2.5. Макет устройства, поддерживающего обмен с параллельным портом, работающим в режиме ECP. Используемые микросхемы: D1 – К155ЛЕ1; D2 – К155ТЛ2; D3 – К155ИЕ5; D4, B5 – К155ЛП11.

Макет передает в порт информацию, которая имеет признак, как данных, так и команд. Этот признак формирует соответствующий уровень сигнала PeriphAck на выходе 13 микросхемы D4.2. Высокий уровень сигнала соответствует передаче в порт данных

(четное значение счетчика – передается по линям четырех младших разрядов шины данных интерфейса через буферное устройство D4.1), а низкий – команд (нечетное значение счетчика – передается старшими четырьмя разрядами шины данных интерфейса через буферное устройство D5.1).

Передача данных и команд в ECP-порт сопровождается сигналом PeriphClk, формируемом на выходе 2 микросхемы D2.1. Этот сигнал активизируется при низком уровне сигнала ReversReq# в ответ на появление высокого уровня сигнала HostAck#. Задержка фронта сигнала PeriphClk относительно фронта сигнала HostAck# обеспечивается RC-цепочкой, подключенной к входу 1 микросхемы D2.1.

При работе ECP-порта в режиме передачи данных или команд макет только формирует квитирующий сигнал PeriphAck на выходе 13 микросхемы D5.2 в ответ на стробирующий сигнал HostClk порта. Задержка фронта сигнала PeriphAck по отношению к фронту сигнала HostClk осуществляется интегрирующей цепочкой R3C3. Сигнал PeriphAck формируется макетом как при приеме данных и команд со стороны ECP-порта, так и при передаче данных и команд в ECP-порт. При этом смысл этих сигналов различен. Поэтому эти сигналы формируются на разных микросхемах, обеспечивающих три состояния на выходах (D4.2 и D5.2). При высоком уровне сигнала ReversReq# (ECP-порт работает в режиме передачи) в третье состояние переводится выход 13 микросхемы D4.2, а при низком (ECP-порт работает в режиме приема) - выход 13 микросхемы D5.2.

Макет формирует также квитирующий сигнал AckReverse# в ответ на сигнал ReversReq# ECP-порта. Задержка AckReverse# осуществляется интегрирующей цепочкой R2C2.

Питающее напряжение +5В подается на макет по дополнительной линии от источника питания ПК. На плате макета необходимо предусмотреть развязку по питанию в области низких и высоких частот.

Подробности работы ECP-порта приведены в разделе 1.3.4.

2.1.2. Кабели и кроссовые платы

Для соединения компьютера с макетами используется кабель AMAM (см. раздел 1.3.6) с одинаковыми 25-контактными разъемами типа вилка (DB-25) на концах. Для защиты параллельных портов в выходные линии кабелей были подключены резисторы величиной 130 Ом (см. табл. 2.1).

Таблица 2.1. Схема доработки кабеля

Разъем 1		Разъем 2	
Контакт	Сопротивление, Ом	Контакт	Сопротивление, Ом
1	130	1	-
2	130	2	-
3	130	3	-
4	130	4	-
5	130	5	-
6	130	6	-
7	130	7	-
8	130	8	-
9	130	9	-
14	130	14	-
15	-	15	-
16	130	16	-
17	130	17	-

Для реализации перекрестного соединения линий интерфейса, используемого при соединении между собой параллельных портов двух компьютеров в режиме ECP и Byte Mode, используется кроссовая плата и два кобеля AMAM с защитными резисторами. На крос-

совой плате между двумя 25-контактными разъемами типа розетка производится разводка соединений контактов разъемов согласно таблице 2.2. Третий разъем типа розетка на кроссовой плате предназначен для осуществления контроля состояния линий интерфейса. Одна и та же кроссовая плата используется как в режиме Byte Mode, так и в режиме ECP. Различаются только названия сигналов и их назначение (см. табл. 1.7 раздела 1.3.4 и табл. 1.4 раздела 1.3.2).

Таблица 2.2. Кабель связи PC-PC в режиме ECP и Byte Mode (Bi-directional)

Разъем X1			Разъем X2		
Контакт	Имя в ECP	Имя в Byte Mode	Имя в ECP	Имя в Byte Mode	Контакт
1	HostClk	HostClk	PeriphClk	PerClk	10
14	HostAck	HostBusy	PeriphAck	PerBusy	11
17	1284Active	1284Active	Xflag	Xflag	13
16	ReverseRequest#	Init#	AckReverse#	AckDataReq	12
10	PeriphClk	PerClk	**HostClk**	HostClk	1
11	PeriphAck	PerBusy	HostAck	HostBusy	14
12	AckReverse#	AckDataReq	ReverseRequest#	Init#	16
13	Xflag	Xflag	1284Active	1284Active	17
2-9	Data[0:7]	Data[0:7]	Data[0:7]	Data[0:7]	2-9

В табл. 2.3 описано назначение выводов разъема LPT-порта в различных режимах и их соответствие битам регистров стандартного порта.

Таблица 2.2. Назначение выводов разъема LPT-порта и бит регистров в режимах SPP, ECP и EPP

Контакт	**I/O**	**Бит**[1]	**SPP**	**Bi-Di**	**ECP**	**EPP**
1	O	CR.0\	Strobe#	HostClk	HostClk	Write#
2	I/O	DR.0	Data 0	Data 0	Data 0	Data 0
3	I/O	DR.1	Data 1	Data 1	Data 1	Data 1
4	I/O	DR.2	Data 2	Data 2	Data 2	Data 2
5	I/O	DR.3	Data 3	Data 3	Data 3	Data 3
6	I/O	DR.4	Data 4	Data 4	Data 4	Data 4
7	I/O	DR.5	Data 5	Data 5	Data 5	Data 5
8	I/O	DR.6	Data 6	Data 6	Data 6	Data 6
9	I/O	DR.7	Data 7	Data 7	Data 7	Data 7
10	I	SR.6	Ack#	PerClk	PeriphClk	INTR#
11	I	SR.7\	Busy	PerBusy	PeriphAck	Wait#
12	I	SR.5	PaperEnd	AckDataReq[1]	AckReverse#	-[2]
13	I	SR.4	Select	Xflag[1]	Xflag	-[2]
14	O	CR.1\	Auto LF#	HostBusy	HostAck	DataStb#
15	I	SR.3	Error#	DataAvail#[1]	PeriphRequest#	-[2]
16	O	CR.2	Init	Init	ReverseRequest#	Reset#
17	O	CR.3\	Select In#	1284Active	1284Active	AddrStb#

[1] Символом « \ » отмечены инвертированные сигналы (1 в регистре соответствует низкому уровню линии).

[2] Определяется пользователем.

2.2. Общие указания по выполнению и оформлению лабораторных работ

Выполнение каждой лабораторной работы состоит из трех этапов:

1. подготовка к лабораторной работе;
2. проведение работ по непосредственному практическому изучению двунаправленного параллельного порта ПК, принципов его работы и программирования в изучаемом режиме;
3. оформление отчета и защита лабораторной работы.

Подготовка к лабораторной работе заключается в изучении описания лабораторной работы, описания соответствующего режима работы контроллера параллельного порта, отладчика AFD, ознакомление с примерами программ и временными диаграммами, поясняющими правила обмена информацией через параллельный порт в соответствующем режиме его работы. Подготовка к работе проводится студентом самостоятельно в рамках внеаудиторной работы.

Проведение работ по непосредственному изучению принципов работы двунаправленного параллельного порта разбивается на два этапа.

На первом этапе выполняются все пункты задания, сформулированные в разделе описания "Порядок выполнения лабораторной работы". Они включают в себя: набор, отладку и выполнение фрагментов программ, приведенных в описании, в среде отладчика AFD. Во время отладки и выполнения фрагментов программ контролируется состояние линий интерфейса и контроллера порта как по изменению содержимого окон отладчика AFD, так и с помощью осциллографа или других измерительных средств.

На втором этапе выполняются индивидуальные задания, сформулированные преподавателем после проверки результатов первого этапа работы.

Оформление отчета по лабораторной работе осуществляется во внеаудиторное время.

Отчет по лабораторной работе должен содержать следующий материал:

- постановку задачи выполняемой лабораторной работы;
- краткое описание работы параллельного порта в соответствующем режиме;
- тексты программ на ассемблере (мнемокоды команд процессора 8086) с подробными комментариями;
- осциллограммы, таблицы и т.п.;
- выводы по проделанной работе.

Отчет должен быть оформлен грамотно и аккуратно.

Сдача (защита) лабораторной работы производится во время выполнения следующей лабораторной работы или в дополнительное время по согласованию с преподавателем.

2.3. Лабораторная работа 1 (описание)
“Стандартный параллельный порт ПК (SPP-порт)“

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы “Стандартный параллельный порт ПК (SPP-порт)“ является ознакомление с принципами работы стандартного параллельного порта, принципами его программирования на уровне регистров контроллера порта и на уровне функций прерывания INT 17h BIOS, а также с приемами непосредственной его программной настройки на передачу данных и программной реализации протокола интерфейса Centronics.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ В РАБОТЕ

В лабораторной работе используется любой ПК со стандартным параллельным портом, начиная с IBM PC/AT, электронный осциллограф и может использоваться макет внешнего устройства (см. раздел 2.1).

3. КРАТКОЕ ОПИСАНИЕ SPP-ПОРТА

SPP (Standard Parallel Port) является однонаправленным портом, через который программно реализуется протокол обмена интерфейса Centronics (см. рис.1)

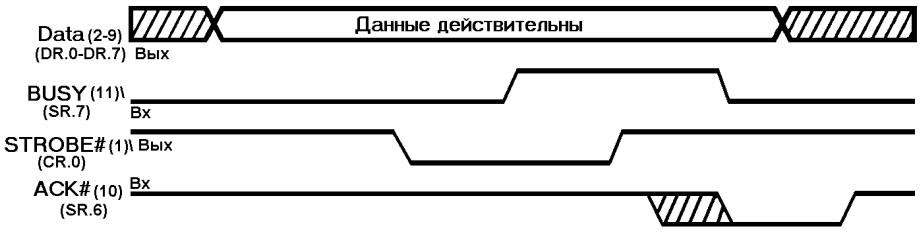


Рис. 1. Передача данных по протоколу Centronics

Перечислим шаги программной реализации процедуры вывода байта по интерфейсу Centronics с указанием требуемого количества шинных операций процессора.

1. Вывод байта в регистр данных (1 цикл IOWR#).
2. Ввод из регистра состояния и проверка готовности устройства (бит SR. 7 — сигнал Busy). Этот шаг зацикливается до получения готовности или до срабатывания программного тайм-аута (минимум 1 цикл IORD#).
3. По получению готовности выводом в регистр управления устанавливается строб данных (бит CR. 0), а следующим выводом строб снимается. Обычно, чтобы переключить только один бит (строб), регистр управления предварительно считывается, что к двум циклам IOWR# добавляет еще один цикл IORD#.

Видно, что для вывода одного байта требуется 4-5 операций ввода-вывода с регистрами порта (в лучшем случае, когда готовность обнаружена по первому чтению регистра состояния).

Название и назначение сигналов разъема порта также соответствует интерфейсу Centronics (см. табл. 1.1 и 1.2 раздела 1.1 пособия).

Для управления работой порта, определения состояния порта и подключенного к нему устройства, а также программной реализации протокола передачи данных контроллер SPP-порта содержит три 8-разрядных регистра, расположенных по соседним адресам в пространстве ввода-вывода, начиная с базового адреса порта BASE (3BCh, 378h или 278h).

Data Register **(DR) —** ***регистр данных,*** адрес: BASE. Данные, записанные в этот регистр, *выводятся* на выходные линии Data [7:0]. Данные, считанные из этого регистра, в зависимости от схемотехники адаптера соответствуют либо ранее записанным данным, либо сигналам, поступающим на вход порта со стороны внешнего устройства.

Status Register **(SR) —** ***регистр состояния*** (только чтение), адрес: BASE+1. Регистр

отображает *5-битный порт ввода* сигналов состояния принтера и флаг прерывания. Бит SR.7 инвертируется — низкому уровню сигнала соответствует единичное значение бита в регистре, и наоборот.

Ниже приведено назначение бит регистра состояния (в скобках даны номера контактов разъема порта).

SR. 7 — BUSY - инверсное отображение состояния линии Busy (11): при низком уровне на линии устанавливается единичное значение бита — разрешение на вывод очередного байта.

SR. 6 — ACK (Acknowledge) - отображение состояния линии Ack# (10).

SR. 5 — PE (Paper End) - отображение состояния линии Paper End (12). Единичное значение соответствует высокому уровню линии, устанавливается при отсутствии бумаги в принтере.

SR. 4 — Select - отображение состояния линии Select (13). Единичное значение соответствует высокому уровню линии — сигналу о включении принтера.

SR. 3 — Error - отображение состояния линии Error# (15). Нулевое значение соответствует низкому уровню линии — сигналу о любой ошибке принтера.

SR. 2—PIRQ - флаг прерывания по сигналу Ack#. Бит обнуляется, если сигнал Ack# вызвал аппаратное прерывание. Единичное значение устанавливается по аппаратному сбросу и после чтения регистра состояния.

SR [1, 0] - зарезервированы.

Control Register* (CR) — *регистр управления, адрес=BASE+2, допускает запись и чтение. Регистр связан с *4-битным портом вывода управляющих сигналов* (биты 0-3) для которых возможно и чтение; выходной буфер обычно имеет тип «открытый коллектор». Это позволяет корректно использовать линии данного регистра как входные при программировании их в высокий уровень. Биты 0, 1, 3 инвертируются.

Ниже приведено назначение бит регистра управления.

CR [7, 6] — зарезервированы.

CR. 5 — Dir - бит управления направлением передачи (только для портов PS/2, см. ниже, и в портах, поддерживающих стандарт IEEE 1284). Запись единицы переводит порт данных в режим ввода. При чтении регистра CR, состояние бита не определено.

CR. 4— IRQE - разрешение прерывания. Единичное значение разрешает прерывание по спаду сигнала на линии Ack# — сигнал подтверждения приема и запроса следующего байта.

CR. 3—SELIN - единичное значение бита соответствует низкому уровню на выходе Select In# (17) - сигналу, разрешающему работу принтера по интерфейсу *Centronics.*

CR. 2— INIT - нулевое значение бита соответствует низкому уровню на выходе Init# (16) - сигнал аппаратного сброса принтера.

CR. 1 — AUTO LF - единичное значение бита соответствует низкому уровню на выходе Auto LF# (14) - сигналу на автоматический перевод строки (LF — Line Feed) по приему байта возврата каретки (CR). Иногда сигнал и бит называют AutoFD или AutoFDXT.

CR. 0 — STRB - единичное значение бита соответствует низкому уровню на выходе Strobe# (1) — сигналу стробирования выходных данных.

Запрос аппаратного прерывания (обычно IRQ7 или IRQ5) вырабатывается по отрицательному перепаду сигнала на контакте 10 разъема порта (Ack#) при значении CR4 =1. Во избежание ложных прерываний контакт 10 соединён резистором с шиной +5 В. Прерывание вырабатывается, когда принтер подтверждает прием предыдущего байта. BIOS это прерывание не использует и не обслуживает.

Более подробная информация по SPP-порту представлена в разделе 1.1 пособия.

4. СИСТЕМНАЯ ПОДДЕРЖКА SPP-ПОРТА

Системная поддержка BIOS SPP-порта включает идентификацию (поиск) установленных портов на стадии POST (Power On Self Test - тест начального включения), сервисы печати SPP-порта (Int 17h - драйвер принтера и Int 05h - печать копии экрана).

Идентификация LPT-портов на стадии POST.

В процессе начального тестирования POST BIOS проверяет наличие параллельных портов по адресам 3BCh, 378h и 278h и помещает базовые адреса обнаруженных портов в ячейки области данных BIOS (BIOS Data Area) 0:0408h, 0:040Ah, 0:040Ch. Эти ячейки хранят адреса портов LPT1 - LPT3, нулевое значение адреса является признаком отсутствия порта с данным номером. BIOS поддерживает также и LPT4, однако на стадии POST он не ищется. Поэтому, если пользователь желает работать с этим портом он должен сам занести его адрес в ячейку памяти 0:040Eh. BIOS предусматривает также функции тайм-аута портов, длительность которых может программироваться пользователем. Для этого используются адреса 0:0478h, 0:0479h, 0:047Ah и 0:047Bh. Значения тайм-аута устанавливаются POST на 20 и могут быть изменены в диапазоне от 1 до 255. Каждая единица соответствует примерно одной секунде. В BIOS Data Area резервируется также ячейка 0:0500h в которую помещается информация о состоянии функции "Печать копии экрана" (прерывание Int 05h).

Драйвер принтера INT 17h

Для работы через параллельный SPP-порт с принтером предназначена группа функций BIOS, вызываемых по программному прерыванию Int 17h. После выполнения любой из функций данной группы в регистре AH будет возвращен код состояния принтера. Биты кода состояния имеют следующее значение:

- бит 0 — признак тайм-аута (0 — нормальное состояние, 1 — ошибка тайм-аута, то есть принтер не отвечает);
- биты 1 и 2 — не используются, установлены в 0;
- бит 3 — признак ошибки ввода-вывода (0 — ошибка, 1 — нет ошибки);
- бит 4 — признак выбора принтера (0 — принтер в автономном режиме, 1 — принтер в режиме подключения);
- бит 5 — контроль наличия бумаги (0 — бумага имеется, 1 — нет бумаги);
- бит 6 — подтверждение приема (0 — подтверждение приема символа, 1 — обычное состояние - запрос передачи следующего байта);
- бит 7 — признак занятости принтера (0 — принтер занят, 1 — принтер свободен).

Прерывание Int 17h, функция 00h: вывести символ на принтер

Функция предназначена для выполнения побайтного вывода информации. Это основная функция данной группы — она обеспечивает посылку команд и данных на принтер. При этом выполняются протокольные правила интерфейса Centronics. При нарушении этих правил срабатывает функция таймаута. Например, если сигнал Busy удерживается в высоком состоянии в течение времени, превышающем значение таймаута, то функция завершается аварийно и данные в порт не выдаются. Время выполнения функции примерно 15 мкс.

Перед вызовом прерывания требуется записать в регистры процессора следующую информацию:

в AH — значение 00h;

в AL — код выводимого символа;

в DX - номер порта принтера (0 - LPT1, 1 - LPT2, 2 - LPT3, 3 - LPT4).

Прерывание Int 17h, функция 01h: инициализировать порт

Функция предназначена для выполнения инициализации (сброса) интерфейса принтера. Вызывают данную функцию после обнаружения серьезных сбоев в работе принтера. Функция устанавливает линию Init в состояние высокого уровня на время около 20 мкс.

Время выполнения функции примерно 70 мс.

Перед вызовом прерывания требуется записать в регистры следующую информацию:

в AH — значение 01h;

в DX — номер порта принтера (0 - LPT1, 1 - LPT2, 2 - LPT3, 3 - LPT4).

Прерывание Int 17h, функция 02h: получить состояние принтера

Функция возвращает текущее состояние принтера. Обычно ее вызывают перед началом печати очередной строки или нового листа, чтобы проверить готовность принтера к печати: включен ли принтер и заправлена ли в него бумага. Время выполнения функции примерно 5 мкс.

Перед вызовом прерывания требуется записать в регистры следующую информацию:

в AH — значение 02h;

в DX — номер порта принтера (0 - LPT1, 1 - LPT2, 2 - LPT3, 3 - LPT4).

Более подробную информацию по системной поддержке SPP-порта можно найти в разделе 1.4. пособия.

Программирование SPP-порта на уровне регистров и прерывания INT 17h BIOS рассмотрено в разделе 1.9.1. пособия.

5. УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНОЙ РАБОТЕ

Подготовка к лабораторной работе заключается в изучении описания лабораторной работы “Стандартный параллельный порт ПК (SPP-порт)”, описания контроллера параллельного порта (раздел 1.1), набора функций прерывания INT 17h BIOS (раздел 1.4.3), отладчика AFD (раздулы 3.1 и 3.2 приложения 3 пособия), принципов программирования SPP-порта (раздел 1.9.1) и ознакомлении с примерами программ, приведенных в данном описании.

Перед началом работ необходимо убедиться в том, что средствами BIOS Setup установлен SPP режим работы контроллера параллельного порта.

6. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

6.1. Средствами AFD определить базовый адрес параллельного порта ПК.

6.2. Выяснить, исходя из протокола Centronics, какие линии интерфейса необходимо удерживать в верхнем, а какие в нижнем уровне при использовании функции 00h прерывания INT 17h для выдачи данных на шину данных порта (контакты D7 - D0).

6.3. Написать и отладить в среде отладчика AFD фрагмент программы, генерирующей на выводе D0 разъема порта импульсы с положительным полупериодом 10 мкс и отрицательным полупериодом 2,5 мкс, используя в качестве прототипа программу 1 из раздела 7 "Примеры программ". Импульсы наблюдать на экране осциллографа.

6.4. Написать и отладить в среде отладчика AFD фрагмент программы генерирующей такие же импульсы, как и в пункте 6.3, но с использованием функций прерывания INT 17h. Пример использования функций прерывания INT 17h приведен в программе 2.

6.5. Написать и отладить в среде AFD фрагмент программы, реализующей на выводах D0, D1, D2 и D3 импульсы с длительностью положительного полу периода отличающейся в два раза. Длительность полу периода на выводе D0 должна составлять 40 мкс. Управление состоянием выводов D0 - D3 осуществлять функцией 00h прерывания INT 17h. За основу взять программу 2 раздела 7 описания.

6.6. Реализовать предыдущее задание, программируя SPP-порт на уровне регистров. За основу взять программу 1 раздела 7 описания.

6.7. Написать и отладить в среде AFD фрагмент программы, реализующей на выводах D0 - D3 сигналов с частотами, отличающимися в два раза друг от друга. Максимальная частота 100 кГц на выводе D0. За основу взять программу 3 раздела 7 описания.

6.8. Реализовать предыдущее задание с использование функции 00h прерывания INT 17h.

6.9. Написать и отладить программу, генерирующую на выводе D0 сигнал максимально возможной частоты при программировании порта на уровне регистров и с использованием прерывания INT 17h.

6.10. Определить длительность одного цикла программной задержки LOOP.

Примечание: Осциллограммы, наблюдаемые на экране осциллографа, зарисовываются с указанием их параметров и представляются в отчетах по работе.

7. ПРИМЕРЫ ПРОГРАММ

Программа 1. Генерирование прямоугольных импульсов на выводе D0 разъема порта (контакт 2)

```
      MUV     AX,0040      ;Установка сегментного адреса области
      MOV     ES,AX        ;данных BIOS (BIOS Data Area)
      MOV     DX,ES:[0008] ;Занесение базового адреса LPT1 в регистр DX
A:    MOV     AL,01        ;Вывод в регистр данных параллельного порта
      OUT     DX,AL        ;константы 01 (высокий уровень на выводе D0)
      MOV     CX,FF        ;Программная задержка, определяемая
A2:   LOOP    A2           ;константой в регистр CX и быстродействием ПК
      MUV     AL,00        ;Вывод в регистр данных параллельного порта
      OUT     DX,AL        ;константы 00 (низкий уровень на выводе D0)
      MOV     CX,BFF       ;Программная задержка, определяемая
A3:   LOOP    A3           ;константой в регистр CX и быстродействием ПК
      JMP     A            ;Переход на повторение циклов вывода
```

Программа содержит бесконечный цикл. Остановка программы осуществляется комбинацией клавиш Ctrl + Esc.

Программа 2. Генерирование импульсов с различной длительностью положительного полу периода на выводах D0, D1, D2 и D3 (контакты 2, 3, 4, и 5 разъема порта)

```
A:    MOV     AH,00    ;Функция "печать символа"
      MUV     AL,01    ;Символ вывода 01 - высокий уровень на D0
      INT 17  AL,00    ;Вызов драйвера принтера
      MUV     CX,0001  ;Программная задержка, определяемая
A1:   LOOP    A1       ;константой в регистр CX и быстродействием ПК
      MUV     AH,00    ;Функция "печать символа"
      MOV     AL,02    ;Символ вывода 02 - высокий уровень на D1
      INT     17       ;Вызов драйвера принтера
      MOV     CX,00FF  ;Программная задержка, определяемая
A2:   LOOP    A2       ;константой в регистр CX и быстродействием ПК
      MUV     AH,00    ;Функция "печать символа"
      MUV     AL,04    ;Символ вывода 04 - высокий уровень на D2
      INT 17           ;Вызов драйвера принтера
      MUV     CX,02F0  ;Программная задержка, определяемая
A3:   LOOP    A3       ;константой в регистр CX и быстродействием ПК
      MUV     AH,00    ;Функция "печать символа"
      MUV     AL,08    ;Символ вывода 08 - высокий уровень на D3
      INT 17           ;Вызов драйвера принтера
      MUV     CX,0690  ;Программная задержка, определяемая
A4:   LOOP    A4       ;константой в регистр CX и быстродействием ПК
      JMP     A        ;Переход на повторение циклов вывода
```

Программа содержит бесконечный цикл. Остановка программы осуществляется комбинацией клавиш Ctrl + Esc. Программа может остановиться внутри процедуры обработки прерывания INT 17h. В этом случае для возвращения к прерванной программе необходимо

средствами AFD восстановить содержимое сегментного регистра команд CS и регистра указателя команд IP. Содержимое регистра CS восстанавливается по информации, сохранившейся в одном из регистров DS, SS, ES, HS или FS (при загрузке отладчика эти регистры, включая и CS, содержат одинаковые коды), а в IP заносится значение, использовавшееся при запуске программы командой G отладчика (по умолчанию 100h).

Программа 3. Генерирование импульсов на выводах D0 - D7, частоты которых отличаются друг от друга в два раза.

```
        MUV     AX,0040         ;Установка сегментного адреса области
        MOV     ES,AX           ;данных BIOS
        MOV     DX,ES:[0008]    ;Занесение базового адреса LPT1 в регистр DX
        MOV     AL,01           ;Вывод в регистр данных параллельного порта
A:      OUT     DX,AL           ;константы 01 (высокий уровень на выводе D0)
        MOV     CX,FF           ;Программная задержка, определяемая константой,
A2:     LOOP    A2              ;занесенной в регистр CX и быстродействием ПК
        INC     AL              ;Увеличение на 1 содержимого регистра AL
        JMP     A               ;Переход на повторение циклов вывода
```

Программа содержит бесконечный цикл. Остановка программы осуществляется комбинацией клавиш Ctrl + Esc.

Относительная адресация регистров SPP-порта представлена в разделе 3 этого описания. В программах в качестве базового адреса порта используется адрес порта LPT1, указанный по адресу 0040:0008 памяти рабочей области BIOS.

8. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

При подготовке к лабораторным работам и к их защите рекомендуется пользоваться информацией, представленной в основном разделе учебно-методического пособия и в дополнительной литературе [1, 3, 6, 10].

9. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Охарактеризуйте временную диаграмму протокола интерфейса Centronics (рис.1) и особенности ее программной реализации.
2. Охарактеризуйте регистры контроллера SPP порта.
3. Объясните назначение сигнальных линий интерфейса Centronics.
4. Почему длина обычного кабеля интерфейса Centronics ограничена двумя метрами?
5. Опишите схемотехнику SPP порта.
6. В чем заключается системная поддержка SPP порта на уровне BIOS?
7. Охарактеризуйте функции драйвера принтера INT 17h.
8. Чем определяется длительность программной задержки, реализуемой командой LOOP?
9. Как организовать программную задержку длительностью в несколько секунд?
10. Как с помощью осциллографа можно оценить длительность выполнения одного цикла команды LOOP?

2.4. Лабораторная работа 2 (описание) “Работа параллельного порта в EPP режиме“

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы “Работа параллельного порта в EPP режиме“ является ознакомление с принципами работы двунаправленного параллельного порта в режиме EPP, принципами его программирования на уровне регистров контроллера порта и на уровне функций EPP BIOS, а также с приемами непосредственной его программной настройки на прием и выдачу данных и адресной информации.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ В РАБОТЕ

В лабораторной работе используется ПК Pentium с параллельным портом, поддерживающим стандарт IEEE 1284, электронный осциллограф и макет внешнего устройства, поддерживающего протокол EPP порта (см. раздел 2.1).

3. КРАТКОЕ ОПИСАНИЕ EPP-ПОРТА

EPP (Enhanced Parallel Port) - скоростной (до 2 Мбайт/с) двунаправленный интерфейс, являющийся развитием Centronics и Bi-directional режимов LPT порта.

Его главная особенность - прием или передача байта со стороны ПК осуществляется за одно обращение к порту (более того при 16/32 разрядном обращении принимается или передается сразу 2/4 байта соответственно). Цикл квитирования приема-передачи реализуется аппаратно. В отличие от программно-управляемых режимов (режим SPP и Bi-directional), внешние сигналы EPP порта (как информационные, так и сигналы квитирования) для каждого цикла обмена формируются аппаратно по одной операции записи или чтения в регистр данных или адреса порта. Другой отличительной особенностью EPP является возможность передачи и приема не только данных, но и адресной информации. На рис. 1. приведена диаграмма *цикла записи* данных, иллюстрирующая внешний цикл обмена, *вложенный* в цикл записи системной шины ПК (иногда эти циклы называют *связанными)*. В скобках после идентификатора сигнала указан номер контакта разъема, через который проходит этот сигнал.

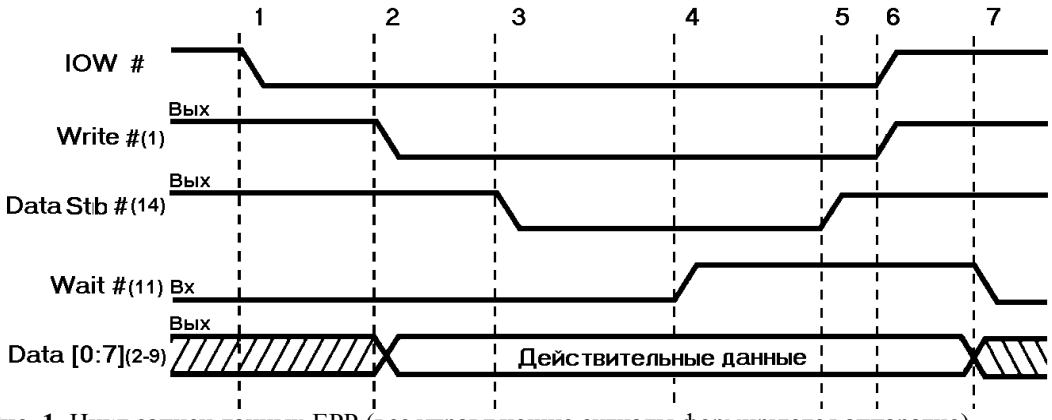


Рис. 1. Цикл записи данных EPP (все управляющие сигналы формируются аппаратно)

Адресный цикл записи отличается от цикла данных только используемым стробом внешнего интерфейса. Пример адресного *цикла чтения* приведен на рис. 2., цикл чтения данных отличается только применением другого стробирующего сигнала.

Существует несколько стандартов EPP интерфейса. Поддержку стандартов EPP1.7 и EPP1.9 можно встретить в BIOS ряда компьютеров (отличие EPP1.7 и EPP1.9 в способе квитирования). Стандарт EPP1.9 был положен в основу стандарта IEEE 1284. Стандарт IEEE 1284 описывает все режимы LPT-порта: Compatibility, Nibble, Byte, EPP, ECP. В стандарте сигналы LPT-порта получили другие функции и другие названия. (Соответствие функций и названий сигналов LPT – порта в режиме SPP и EPP см. в табл. 1.2 раздела 1.1.3 и табл.1.5. раздела 1.3.3 соответственно).

Более подробное описание EPP-порта приведено в разделе 1.3.3.

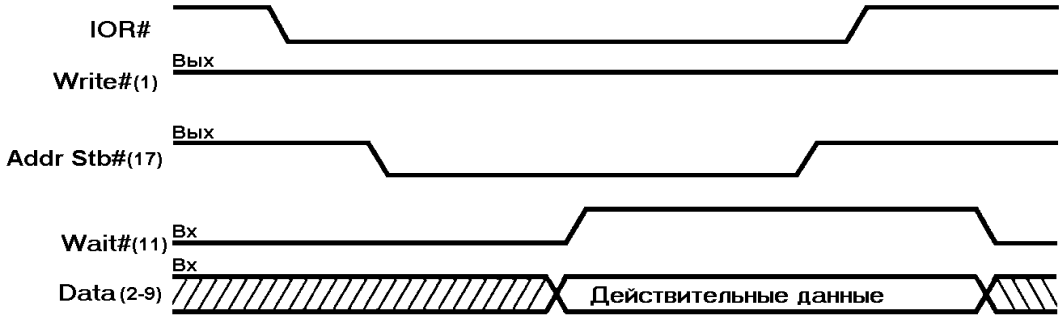


Рис. 2. Адресный цикл чтения EPP (все управляющие сигналы формируются аппаратно)

4. ФУНКЦИИ EPP BIOS

Дополнительный набор функций BIOS, предназначен для обслуживания новых режимов работы и получил наименование EPP BIOS. При работе с функциями EPP BIOS используется прерывание INT 17h, т. е то же прерывание, что и при работе со стандартным параллельным портом (драйвер принтера). К сожалению, появился EPP BIOS со значительным опозданием и до сих пор не поддерживается многими изготовителями системных плат. Кроме того, этот набор, как явствует из его названия, ориентирован на использование режима EPP, который практически не поддерживается изготовителями принтеров. При работе с принтерами реальную пользу могут принести только головная функция - функция 02h (проверка наличия EPP BIOS). В случае если EPP BIOS поддерживается системой, функция возвращает вектор (“точку входа”) для вызова всех остальных функций EPP BIOS. Функция установки режима представляет особый интерес, так как позволяет задать нужный режим работы контроллера прямо из прикладной программы (до появления EPP BIOS операцию выбора режима можно было осуществить только в процессе начальной загрузки компьютера, через BIOS Setup).

Описание функций EPP BIOS приведено в разделе 1.4.4.

Персональные компьютеры, на которых отлаживались фрагменты программ, не поддерживали дополнительных функций EPP BIOS. По этой причине изучение функций EPP BIOS носит в большей степени теоретический характер.

Программирование EPP-порта на уровне регистров и EPP BIOS рассмотрено в разделе 1.9.3.

5. УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНОЙ РАБОТЕ

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 2.2 “Общие указания по выполнению и оформлению лабораторных работ”.

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы “Работа параллельного порта в EPP режиме”, описания контроллера параллельного EPP порта (раздел 1.3.3), дополнительного набора функций EPP BIOS (раздел 1.4.4), отладчика AFD (раздулы 3.1 и 3.2 приложения 3 пособия), принципов программирования EPP-порта (раздел 1.9.3). Необходимо также ознакомиться с описанием макета периферийного устройства, используемого в режиме EPP (см. раздел 2.1) и ознакомиться с примерами программ, приведенными в данном описании (раздел 7).

Перед началом работ необходимо убедиться в том, что средствами BIOS Setup установлен ECP+EPP или ECP режимы работы контроллера параллельного порта. Эти режимы поддерживают программное переключение режима работы контроллера порта в любой из режимов, предусмотренных стандартом IEEE 1284.

6. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Ознакомиться с приведенными ниже программами и описанием макета периферийного устройства (см. раздел 2.1).

2. Подключить макет через соответствующий кабель к разъему параллельного порта ПК.
3. В окне дизассемблера отладчика AFD набрать программу, приведенную в Примере (раздел 7). Проверить и запустить ее командой "G" отладчика AFD. После запуска программы контролировать осциллографом сигналы Write#, DataStb#, AddrStb#, Wait# и сигналы на шине данных. Добиться приема/передачи данных и адреса между компьютером и макетом.
4. Снять осциллограммы управляющих сигналов и сигналов на линиях шины данных. Осциллограммы снимать в режиме внешнего запуска. В качестве запускающего сигнала необходимо применять соответствующие стробирующие сигналы (DataStb# или AddrStb#). Сравнить полученные осциллограммы с временными диаграммами, приведенными в описании.
5. Проверить поддержку дополнительных функций EPP BIOS параллельным портом компьютера (см. раздел 1.4.4.1).
6. Получить дополнительное задание от преподавателя

7. ПРИМЕР ПРОГРАММ

Программа 1 .Выдача на шину адреса/данных EPP-порта данных и адресной информации. Работает в совокупности с макетом, генерирующим квитирующие сигналы EPP режима.

```
        MOV     DX,077A     ;Установка Bi-directional режима через ECR -
        MOV     AL,34       ;главный управляющий регистр ECP
        OUT     DX,AL
        MOV     AL,94       ;Установка EPP режима записью в регистр ECR
        OUT     DX,AL       ;константы 94
        MOV     DX,037A     ;Разрешение работы счетчика D3 макета
        MOV     AL,04       ; высоким уровнем сигнала Reset# (CR.2 =1)
        OUT     DX,AL       ;и высоких уровней на контактах 1, 14, 17
A:      MOV     DX,037C     ;Через регистр данных EPP выдаем на линии
        MOV     AL,AA       ;AD [0:7] (контакты 2-9) константу AAh
        OUT     DX,AL
        MOV     CX,000F     ;Программная задержка
B:      LOOP    B
        MOV     AL,55
        MOV     DX,037B     ;Через регистр адреса EPP выдаем на линии
        OUT     DX,AL       ;AD [0:7] (контакты 2-9) константу 55h
        MOV     CX,000F     ;Программная задержка
C:      LOOP    C
        JMP     A           ;Зацикливание программы (выход из бесконеч-
                            ного цикла по комбинации клавиш Ctrl+Esc)
```

В программе в качестве базового адреса порта используется адрес 378h (адрес LPT2). Описание регистров SPP-порта и Bi-directional приведено в разделе 1.1.2 и в табл. 1.4 раздела 1.3.2, EPP порта - в разделе 1.3.3 и ECP порта - в разделе 1.3.4.

Выше приведенную программу можно модифицировать, сделав ее независимой от конкретного значения базового адреса порта LPT1. Это достигается использованием значения базового адреса из рабочей области данных BIOS по адресу 0040:0008h (см. раздел 1.4.1) и относительные адреса регистров. Относительные адреса регистров (смещения) EPP-порта представлена в табл. 1.6 раздела 1.3.3, а ECP-порта в табл. 1.9 раздела 1.3.4.

Пример такой программы приведен ниже.

Программа 2. Использование относительных адресов регистров порта и базового адреса из рабочей области данных BIOS.

```
        MUV     AX,0040     ;Установка сегментного адреса области
        MOV     ES,AX       ;данных BIOS
        MOV     DX,ES:[0008] ;Занесение базового адреса LPT1 в регистр DX
        ADD     DX,0402     ;Установка в DX адреса регистра ECR
        MOV     AL,34       ;Установка Bi-directional режима через ECR -
        OUT     DX,AL       ;главный управляющий регистр ECP
        SUB     DX,0400     ;Установка адреса регистра CR SPP-порта.
        MOV     AL,04       ;Разрешение счетчика D3 макета высоким
        OUT     DX,AL       ;уровнем сигнала Reset# (CR.2 =1)
        ADD     DX,0400     ;Установка адреса регистра ECR.
        MOV     AL,94       ;Установка EPP режима записью в регистр ECR
        OUT     DX,AL       ;константы 94
        SUB     DX,0402     ;Установка адреса базового регистра порта
A:      ADD     DX,0004     ;Установка адреса регистра данных EPP
        MOV     AL,AA       ;Через регистр данных EPP выдается на линии
        OUT     DX,AL       ;AD [0:7] (контакты 2-9) константа AAh
        MOV     CX,000F     ;Программная задержка, определяемая
B:      LOOP    B           ;константой в регистр CX и быстродействием ПК
        SUB     DX,0001     ;Установка адреса регистра адреса EPP
        MOV     AL,55       ;Через регистр адреса EPP выдается на линии
        OUT     DX,AL       ;AD [0:7] (контакты 2-9) константа 55h
        MOV     CX,000F     ;Программная задержка, определяемая
C:      LOOP    C           ;константой в регистр CX и быстродействием ПК
        SUB     DX,0003     ;Установка в DX адреса базового регистра порта
        JMP     A           ;Зацикливание программы (выход из бесконечного
                            цикла по комбинации клавиш Ctrl+Esc)
```

8. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

При подготовке к лабораторным работам и к их защите рекомендуется пользоваться информацией, представленной в основном разделе учебно-методического пособия и в дополнительной литературе [1, 3, 6, 10].

9. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как расшифровывается аббревиатура EPP?
2. Каковы особенности EPP порта?
3. Какими способами можно установить EPP режим работы контроллера параллельного порта, поддерживающего стандарт IEEE 1284?
4. Что означает связность циклов чтения/записи EPP порта?
5. Чем отличается стандарт EPP 1.7 и EPP 1.9?
6. Какие дополнительные регистры содержит контроллер EPP порта (по сравнению с контроллером SPP порта)?
7. Охарактеризуйте системную поддержку EPP порта на уровне BIOS.
8. Как вызываются функции EPP BIOS?
9. Каковы особенности использования относительных адресов регистров контроллера EPP порта?

2.5. Лабораторная работа 3 (описание) “Работа параллельного порта в ECP режиме“

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы “Работа параллельного порта в ECP режиме“ является ознакомление с принципами работы двунаправленного параллельного порта в режиме ECP, принципами его программирования, а также с приемами непосредственной его программной настройки на другие режимы работы и на выдачу и прием команд и данных непосредственно в режиме ECP.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА , ИСПОЛЬЗУЕМЫЕ В РАБОТЕ

В лабораторной работе используется ПК Pentium с параллельным портом, поддерживающим стандарт IEEE 1284, электронный осциллограф и макет внешнего устройства, поддерживающего протокол ECP порта (см. раздел 2.1).

3. КРАТКОЕ ОПИСАНИЕ ECP-ПОРТА

Протокол ECP (Extended Capability Port — порт с расширенными возможностями) был предложен фирмами Hewlett Packard и Microsoft как прогрессивный режим связи с периферией типа принтеров и сканеров. Данный протокол обеспечивает высокопроизводительный двунаправленный обмен данными компьютера с периферийными устройствами. Контроллер ECP порта, также как и контроллер EPP порта генерирует внешние протокольные сигналы квитирования аппаратно, но его работа существенно отличается от режима EPP.

На рис. 1,а приведена диаграмма двух циклов прямой передачи: цикла данных и командного цикла.

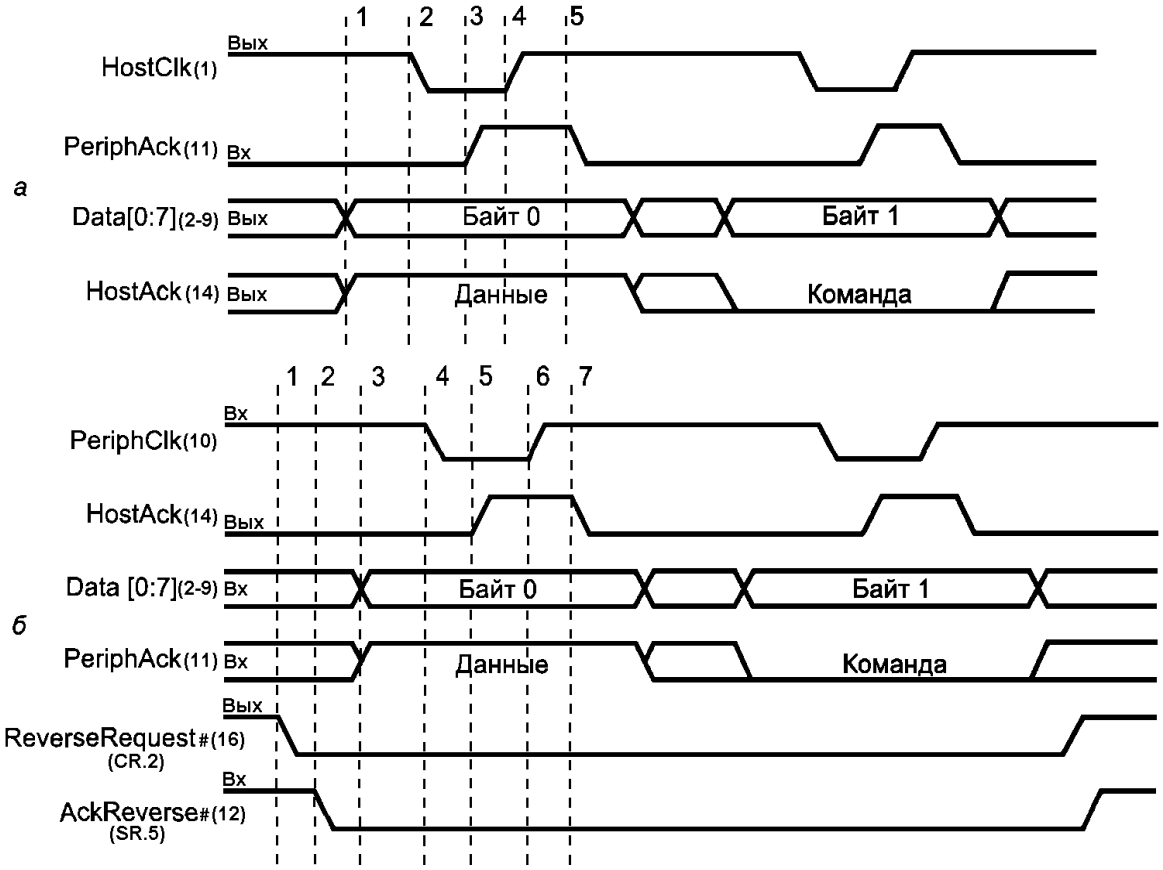


Рис. 1. Передача в режиме ECP: а — прямая, б — обратная

Обратите внимание на то, что, в отличие от диаграмм обмена EPP (см. пункт 1.3.3), на данных рисунках не приведены сигналы циклов системной шины процессора. Это не случайно, поскольку в данном режиме обмен программы с периферийным устройством разбивается на два относительно независимых процесса, которые связаны через FIFO-буфер.

Обмен с FIFO-буфером порта может осуществляться как с использованием DMA, так и программным вводом-выводом. Обмен ПУ с буфером аппаратно выполняет контроллер ECP порта. Драйвер в режиме ECP не имеет информации о точном состоянии процесса обмена, но в случае обмена с принтером обычно важна только информация о том, завершен он или нет.

На рис.1, б приведена пара циклов обратной передачи.

Поскольку передачи в ECP разделены FIFO-буферами, которые могут присутствовать на обеих сторонах интерфейса, важно понимать, на каком этапе данные можно будет считать переданными. Из рис.1 видно и другое отличие ECP от EPP. В режиме ECP *смена направления* должна быть обязательно согласована.

Контроллер ECP порта позволяет программно устанавливать различные режимы работы и для реализации своих функций содержит дополнительные, по сравнению с режимом EPP, регистры.

Режимы, поддерживаемые ECP – портом представлены в табл. 1.8 раздела 1.3.4.2 и табл. 1.16 раздела 1.9.2.1, а его регистры - в табл. 1.9 раздела 1.3.4.2 и 1.15 раздела 1.9.2.1.

Из таблицы 1.8 раздела 1.3.4.2 видно, что если параллельный порт установлен с помощью процедуры SETUP в режим ECP или ECP+EPP, то существует программная возможность переключения порта в другие, указанные в таблице 1.8 режимы. Это переключение осуществляется с помощью главного управляющего регистра ECR, формат которого представлен в табл. 1.

Таблица 1. Структура регистра ECR

ECR7	ECR6	ECR5	ECR4	ECR3	ECR2	ECR1	ECR0
ECP MODE			ERRINTREN#	DMAEN	SERVICEINTR	FIFOFS	FIFOES

- ECR.0 (FIFOES) — признак освобождения очереди (1 — очередь пуста, 0 — очередь содержит по крайней мере один байт данных);
- ECR.1 (FIFOFS) — признак отсутствия свободного места в очереди (0 — в очереди есть место по крайней мере для одного байта данных; 1 — очередь заполнена);
- ECR.2 (SERVICEINTR) — блокировка служебных прерываний (0 — прерывания разрешены, 1 — запрещено использование DMA и обслуживание прерываний);
- ECR.3 (DMAEN) — управление режимом DMA (0 — использование DMA запрещено, 1 — использование DMA разрешено при условии SERVICEINTR = 0);
- ECR.4 (ERRINTREN#) — блокировка обслуживания прерывания по сигналу Error# контакт 15 разъема порта (0 — разрешено прерывание при переходе сигнала Error# с высокого уровня на низкий, 1 — прерывание от Error# запрещено);
- ECR.5 - ECR.7 — код режима работы контроллера ECP (табл. 1.8.или 1.16).

В связи с этим все лабораторные работы цикла проводятся с настроенным на режим ECP или ECP+EPP контроллером порта. Их программная настройка на рабочий режим должна быть предусмотрена в соответствующей управляющей программе (см. примеры программ в описаниях лабораторных работ).

Более подробная информация по всему режиму ECPприведена в разделе 1.3.4 , а по его программированию - в разделе 1.9.2. пособия.

4. УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНОЙ РАБОТЕ

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 2.2 “Общие указания по выполнению и оформлению лабораторных работ”.

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы “Ребота парелельного порта в ECP режиме”, описания контроллера параллельного ECP порта (раздел 1.3.4), отладчика AFD (раздулы 3.1 и 3.2 приложения 3 пособия), принципов программирования ECP-порта (раздел 1.9.2). Необходимо также ознакомиться

с описанием макета периферийного устройства, используемого в режиме ECP (см. раздел 2.1) и ознакомиться с примерами программ, приведенными в данном описании (раздел 6).

Перед началом работ необходимо убедиться в том, что средствами BIOS Setup установлен ECP или ECP+EPP режимы работы контроллера параллельного порта. Эти режимы поддерживают программное переключение режима работы контроллера порта в любой из режимов, предусмотренных стандартом IEEE 1284.

5. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

5.1.Ознакомиться с приведенными ниже программами и описанием макета периферийного устройства (см. раздел 2.1).

5.2.Подключить макет через соответствующий кабель к разъему параллельного порта ПК.

5.3.Отладить программу вывода данных в режиме ECP (в данной лабораторной работе рассматривается работа параллельного порта без поддержки DMA(ПДП)).

5.4.Отладить программу вывода адреса в режиме ECP (в данной лабораторной работе рассматривается работа параллельного порта без поддержки DMA(ПДП)).

5.5.Снять осцилограммы передачи данных и передачи адреса. Снятие осцилограмм проводить с внешней синхронизацией относительно сигнала HostClk(1). Сравнить полученные осциллогаммы с приведенными временными зависимостями на рис 1

5.6.Написать и отладить фрагмент программы, реализующей алгоритмы переключения направления передачи данных, приведенные в разделе 1.9.2.5. К порту должен быть подключен макет устройства, схема которого представлена на рис. 2.5.

5.7.Получить индивидуальное задание от преподавателя и выполнить его.

6. ПРИМЕРЫ ПРОГРАММ

Программа 1. Передача данных в режиме ECP

```
        MOV     DX,077A     ;Установка адреса регистра ECR
        MOV     AL,34       ;Установка Bi-directional режима через ECR -
        OUT     DX,AL       ;главный управляющий регистр ECP
        MOV     DX,037A     ;Задание режима прямой передачи, установкой линии
        MOV     AL,04       ;ReverseRequest#(контакт 16) в высокий уровень через
        OUT     DX,AL       ;разряд CR.2 регистра управления SPP (Init#)
        MOV     DX,077A     ;Занесение в DX адреса регистра ECR порта
        MOV     AL,74       ;Установка режима ECP занесением в регистр ECR
        OUT     DX,AL       ;порта константы 74
A       MOV     DX,0778     ;Занесение в DX адреса регистра ECPDFIFO порта
        MOV     AL,AA       ;Вывод константы AAh через регистр ECPDFIFO
        OUT     DX,AL       ;
        MOV     CX,FF       ;программная задержка, определяемая содержимым
A1:     LOOP    A1          ;CX и быстродействием ПК
        JMP     A           ;зацикливание программы. Выход из цикла: Ctrl+Esc
```

Программа 2. Передача адреса в режиме ECP

```
        MUV     AX,0040     ;Установка сегментного адреса области
        MOV     ES,AX       ;данных BIOS (BIOS Data Area)
        MOV     DX,ES:[0008] ;Занесение базового адреса LPT1 в регистр DX
        ADD     DX,0402     ;Установка в DX адреса регистра ECR (402-смещение)
        MOV     AL,34       ;Установка режима Bi-Di занесением в регистр ECR
        OUT     DX,AL       ;порта константы 34
        SUB     DX,03FE     ;Установка адреса регистра CR
        MOV     AL,04       ;Установка линии Init# (ReverseRequest#) в состояние
        OUT     DX,AL       ;высокого уровня через разряд CR.2
        ADD     DX,03FE     ;Установка в DX адреса регистра ECR
```

```
        MOV     AL,74       ;Установка режима ECP занесением в регистр ECR
        OUT     DX,AL       ;порта константы 74
        SUB     DX,402      ;Установка в DX адреса регистр ECPAFIFO
A:      MOV     AL,55       ;Вывод через регистр ECPAFIFO
        OUT     DX,AL       ;константы 55h
        MOV     CX,FF       ;Программная задержка, определяемая содержимым
A1:     LOOP    A1          ;CX и быстродействием ПК
        JMP     A           ;зацикливание программы. Выход из цикла: Ctrl+Esc
```

В программе 1 используются абсолютные адреса регистров порта LPT2. В программе 2 базовый адрес порта определяется из памяти рабочей области BIOS, относящейся к базовым адресам параллельных портов (см. раздел 1.4.1). Адреса остальных регистров определяются смещением относительно базового адреса (см. табл. 1.9 раздела 1.3.4.2 и табл. 1.15 раздела 1.9.2.1).

7. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

При подготовке к лабораторным работам и к их защите рекомендуется пользоваться информацией, представленной в основном разделе учебно-методического пособия и в дополнительной литературе [1, 3, 6, 10].

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как расшифровывается аббревиатура EPP?
2. Для чего используется ECP порт?
3. В чем сходство и различие ECP и EPP портов?
4. Как изменить направление передачи информации через ECP порт?
5. Как реализуется сжатие информации ECP портом?
6. Охарактеризуйте буферизацию в ECP порту.
7. Какие дополнительные регистры по сравнению с EPP портом содержит контроллер ECP порта?
8. Охарактеризуйте функции разрядов регистра ECR - главного (дополнительного) управляющего регистра ECP.
9. В каких режимах может работать контроллер ECP порта?
10. В чем особенность адресации регистров ECP порта?
11. Охарактеризуйте алгоритм (протокол) прямой и обратной передачи через ECP порт.
12. Охарактеризуйте все регистры ECP порта; в каких режимах они применяются?
13. Как производится согласование режимов IEEE 1284?
14. Охарактеризуйте физический и электрический интерфейсы IEEE 1284.
15. Каковы этапы развития стандарта IEEE 1284?

2.6. Лабораторная работа 4 (описание)

“Передача данных через ECP порт с использованием режима DMA(ПДП)“

1 ЦЕЛЬ РАБОТЫ

Целью лабораторной работы “Передача данных через ECP порт с использованием режима DMA(ПДП)“ является ознакомление со структурой подсистемы DMA (ПДП), принципами ее работы и использование режима DMA при обмене через параллельный порт ECP.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА , ИСПОЛЬЗУЕМЫЕ В РАБОТЕ

В лабораторной работе используется ПК Pentium с параллельным портом, поддерживающим стандарт IEEE 1284, электронный осциллограф и макет внешнего устройства, поддерживающего протокол ECP порта (см. раздел 2.1). В качестве макета может использоваться ПК с параллельным портом, поддерживающим стандарт IEEE 1284.

3. КРАТКОЕ ОПИСАНИЕ ECP ПОРТА

Протокол ECP (Extended Capability Port — порт с расширенными возможностями) был предложен фирмами Hewlett Packard и Microsoft как прогрессивный режим связи с периферией типа принтеров и сканеров. Данный протокол обеспечивает высокопроизводительный двунаправленный обмен данными компьютера с периферийными устройствами. Контроллер ECP порта, также как и контроллер EPP порта генерирует внешние протокольные сигналы квитирования аппаратно, но его работа существенно отличается от режима EPP.

На рис. 1,а приведена диаграмма двух циклов прямой передачи: цикла данных и командного цикла.

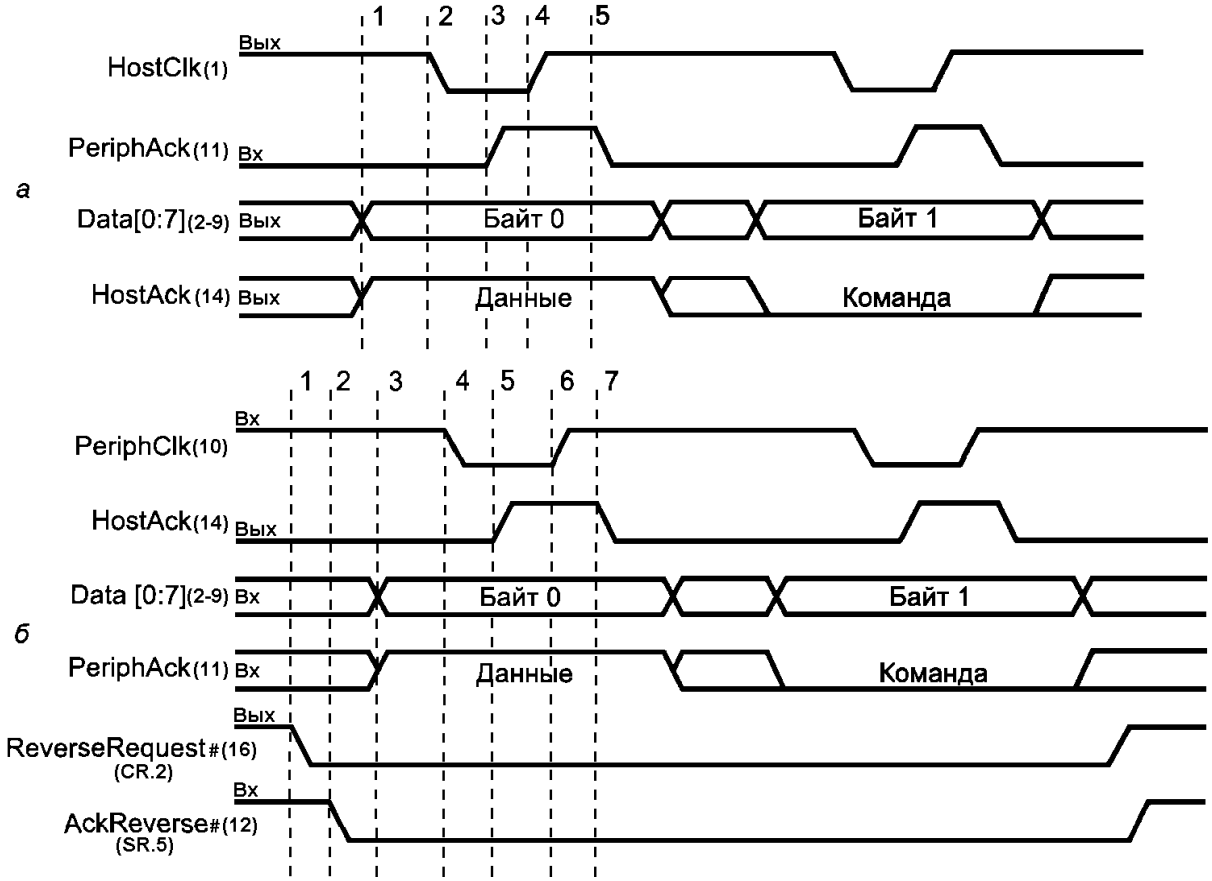


Рис. 1. Передача в режиме ECP: а — прямая, б — обратная

Обратите внимание на то, что, в отличие от диаграмм обмена EPP (см. пункт 1.3.3), на данных рисунках не приведены сигналы циклов системной шины процессора. Это не случайно, поскольку в данном режиме обмен программы с периферийным устройством разби-

вается на два относительно независимых процесса, которые связаны через FIFO-буфер. Обмен с FIFO-буфером порта может осуществляться как с использованием DMA, так и программным вводом-выводом. Обмен ПУ с буфером аппаратно выполняет контроллер ECP порта. Драйвер в режиме ECP не имеет информации о точном состоянии процесса обмена, но в случае обмена с принтером обычно важна только информация о том, завершен он или нет.

На рис.1, б приведена пара циклов обратной передачи.

Поскольку передачи в ECP разделены FIFO-буферами, которые могут присутствовать на обеих сторонах интерфейса, важно понимать, на каком этапе данные можно будет считать переданными. Из рис.1 видно и другое отличие ECP от EPP. В режиме ECP *смена направления* должна быть обязательно согласована.

Контроллер ECP порта позволяет программно устанавливать различные режимы работы и для реализации своих функций содержит дополнительные, по сравнению с режимом EPP, регистры.

Режимы, поддерживаемые ECP – портом представлены в табл. 1.8 раздела 1.3.4.2 и табл. 1.16 раздела 1.9.2.1, а его регистры - в табл. 1.9 раздела 1.3.4.2 и 1.15 раздела 1.9.2.1.

Из таблицы 1.8 раздела 1.3.4.2 видно, что если параллельный порт установлен с помощью процедуры SETUP в режим ECP или ECP+EPP, то существует программная возможность переключения порта в другие, указанные в таблице 1.8 режимы. Это переключение осуществляется с помощью главного управляющего регистра ECR, формат которого представлен в табл. 1.

Таблица 1. Структура регистра ECR

ECR7	ECR6	ECR5	ECR4	ECR3	ECR2	ECR1	ECR0
ECP MODE			ERRINTREN#	DMAEN	SERVICEINTR	FIFOFS	FIFOES

- ECR.0 (FIFOES) — признак освобождения очереди (1 — очередь пуста, 0 — очередь содержит по крайней мере один байт данных);
- ECR.1 (FIFOFS) — признак отсутствия свободного места в очереди (0 — в очереди есть место по крайней мере для одного байта данных; 1 — очередь заполнена);
- ECR.2 (SERVICEINTR) — блокировка служебных прерываний (0 — прерывания разрешены, 1 — запрещено использование DMA и обслуживание прерываний);
- ECR.3 (DMAEN) — управление режимом DMA (0 — использование DMA запрещено, 1 — использование DMA разрешено при условии SERVICEINTR = 0);
- ECR.4 (ERRINTREN#) — блокировка обслуживания прерывания по сигналу Error# контакт 15 разъема порта (0 — разрешено прерывание при переходе сигнала Error# с высокого уровня на низкий, 1 — прерывание от Error# запрещено);
- ECR.5 - ECR.7 — код режима работы контроллера ECP (табл. 1.8.или 1.16).

Более подробная информация по всему режиму ECP приведена в разделе 1.3.4, а по программированию ECP порта - в разделе 1.9.2 пособия.

4. КРАТКОЕ ОПИСАНИЕ ОСОБЕННОСТЕЙ РАБОТЫ ECP ПОРТА В РЕЖИМЕ DMA

Прямой доступ к памяти - ПДП (DMA - Direct Memory Access) - это метод непосредственного обращения к памяти, минуя процессор. Процессор отвечает только за программирование (инициализацию) каналов DMA: настройку на определенный тип передачи, задание начального адреса и размера массива обмениваемых данных. Обычно DMA используется для обмена блоками данных между системной памятью и относительно быстродействующими устройствами ввода-вывода.

Подготовка к передаче данных через ECP порт в режиме DMA (ПДП) начинается с установки направления передачи порта. Далее настраивается третий канал контроллера DMA:

- по адресу 000Bh загружается байт инструкции режима работы канала (см. приложение к данной лабораторной работе (раздел 9.2.7.4));
- по адресу 0006h загружаются два младших байта адреса буфера данных, выделяемого в оперативной памяти (см. раздел 9.1.4 данного описания);
- по адресу 0082h регистра страниц загружается третий байт этого адреса (см. раздел 9.1.3 данного описания);
- по адресу 0007h загружается количество передаваемых байтов (см. раздел 9.1.4 данного описания);
- по адресу 000Ah загружается команда 03h сброса разряда маски третьего канала, (см. раздел 9.1.4 и 9.2.7.7 данного описания).

Затем нужно установить вектор прерывания от параллельного порта на обработчик прерывания, обрабатывающий завершение передачи в режиме DMA (если предусмотрено использование прерываний). После этого нужно разрешить служебные прерывания и использование DMA портом ECP, для чего следует сбросить в 0 бит блокировки служебных прерываний SERVICEINTR и установить в 1 бит управления режимом DMA DMAEN в регистре ECR порта ECP (см. таблицу 1 данного описания). Это инициирует начало процесса обмена в режиме DMA. Процесс передачи данных будет происходить без участия центрального процессора; когда передача данных будет завершена, следует запретить прерывания и режим DMA.

5. УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНОЙ РАБОТЕ

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 2.2 “Общие указания по выполнению и оформлению лабораторных работ”.

Подготовка к данной лабораторной работе заключается в изучении описания лабораторной работы “Передача данных через ECP порт с использованием режима DMA (ПДП)”, описания контроллера параллельного ECP порта (раздел 1.3.4), принципов его программирования (раздел 1.9.2) и правилами работы с отладчиком AFD (раздулы 3.1 и 3.2 приложения 3 пособия). Необходимо также ознакомиться с описанием макета периферийного устройства, используемого в режиме ECP (см. раздел 2.1), с принципами работы и программирования подсистемы DMA (см. приложение к данной лабораторной работе - раздел 9) и ознакомиться с примерами программ, приведенными в данном описании (раздел 6).

Перед началом работ необходимо убедиться в том, что средствами BIOS Setup установлен ECP или ECP+EPP режимы работы контроллера параллельного порта с поддержкой DMA (канал 3).

6. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

6.1.Ознакомиться с приведенной ниже программой и описанием макета периферийного устройства, поддерживающего протокол ECP (см. раздел 2.1).

6.2.Подключить к разъему параллельного порта ПК через соответствующий кабель макет устройства, поддерживающего протокол ECP или через соответствующую кроссовую плату подключить ECP порт другого компьютера.

6.3.Отладить программу вывода данных в режиме ECP с поддержкой DMA(ПДП).

6.4.Снять осцилограммы передачи данных. Снятие осцилограмм проводить с внешней синхронизацией относительно сигнала HostClk(1). Сравнить полученные осциллогаммы с временными диаграммами, приведенными в данном описании на рис 1.

6.5.Написать и отладить фрагмент программы, реализующей алгоритм приема данных в режиме DMA с учетом правил переключения направления передачи через ECP порт (см. разделы 1.3.4.1 и 1.9.2.5 пособия). К порту должен быть подключен макет устройства, схема которого представлена на рис. 2.5.

6.6.Получить индивидуальное задание от преподавателя и выполнить его.

Примечание: Если вместо макета внешнего устройства используется параллельный порт другого компьютера, то его надо подключать через соответствующую кроссовую плату (см. резделл 2.1.2), а на компьютере-макете необходимо запустить соответствующую программу, например программу 2 резделла 7 данного описания. Эта программа эмулирует устройство-приемник, поддерживающего протокол ECP.

7. ПРИМЕРЫ ПРОГРАММ

Приведенные ниже программы настроены на работу с портом LPT2 (базовый адрес 378h, см. раздел 1.4.1), поддерживающим стандарт IEEE 1284. Перед началом работы с программами необходимо средствами BIOS Setup установить ECP или ECP+EPP режим работы параллельного порта с поддержкой прямого доступа (DMA).

Для нормального функционирования программы 1 необходимо к разъему порта подключить макет устройства, поддерживающего протокол ECP порта или ECP порт другого компьютера, на котором должна работать программа, реализующая прием и передачу данных через параллельный порт в режиме ECP.

Программа 1 (передатчик) реализует следующие действия:

1.Переключает порт ECP в режим Bi-directional (запись константы 34h в регистр ECR).

2. Записью константы 04h в регистр управления стандартного порта CR (адрес 37Ah) устанавливает на выводах 16 и 17 разъема порта высокие уровни сигналов ReverseRequest# и 1284Active, а также настраивает порт на передачу (см. раздел 1.1.2 и табл. 1.7 раздела 1.3.4.1)

3. Производит инициализацию третьего канала прямого доступа к памяти DMA1:

3.1. В регистр режима ведомого контроллера DMA1 с адресом 00Bh (см. табл. 4 раздела 10.1.4 описания) записывает конфигурационную константу 4Bh, настраивающую канал 3 DMA1 на режим одиночного вывода с инкрементацией счетчика адреса (см. формат регистра режима в приложении описания - пункт 10.2.7.4).

Примечание: В режиме декремента счетчика адреска (константа 6Bh вместо 4Bh) подсистема DMA функционирует ненормально. При константе 6Bh сеанс DOS Windows прерывается по ошибке, а при работе в режиме командной строки передается только один байт массива. При блочной передаче (константа 8Bh) результат аналогичен одиночной передаче. Не работает режим авто инициализации (константа 5Bh). При константе 5Bh контроллер работает как при константах 4B.

3.2. Преобразует 16-разрядное значение сегментного регистра DS в 20-разрядный код, указывающий на базовый адрес блока памяти, из которого в режиме DMA выводятся данные через ECP порт. Значение 20-разрядного адреса формируется в регистрах AX и DL умножением содержимого DS на 10h (16 в десятичном виде).

3.3. Записывает два младших байта 20-разрядного базового адреса из AX в 16-разрядный счетчик адреса канала 3 контроллера DMA1. Запись производится по адресу 006h (см. табл. 4 в разделе 10.1.4. описания) сначала младший байт из AL, а затем второй - из AH.

3.4. Записывает третий байт базового адреса из DL в регистр страниц канала 3 подсистемы прямого доступа к памяти по адресу 82h (см. табл. 3 в разделе 10.1.3.описания).

3.5. Записывает количество циклов DMA (в нашем случае 0Fh) в регистр счетчика числа циклов DMA канала 3 по адресу 07h (см. табл. 4 раздела 10.1.4.описания). Записывается сначала младший байт (0Fh), а затем - старший - (00h).

3.6. Сбрасывает разряд маски канала 3 DMA1, записывая константу 03h (см. раздел 10.2.7.7 описания) по адресу 0Ah (см. табл. 4 раздела 10.1.4.описания).

4. Устанавливает режим ECP порта с разрешением DMA записью в регистр ECR по адресу 77Ah (см. табл. 1.9 раздела 1.3.4.2) константы 78h (см. табл. 1.8 и описание регистра ECR в разделе 1.3.4.2). Установка этого режима инициирует начало передачи данных в режиме DMA.

5. Тестирование регистра состояния контроллера DMA1 (адрес 08h) на окончание циклов DMA по каналу 3.

6. Переключение ECP порта в режим запрещения DMA (константа 74h заносится в регистр ECR по адресу 77Ah)

7. Организует программную задержку на время, определяемое содержимым регистра CX и временем выполнения команды LOOP. Эта задержка позволяет отделить 16-байтовые передачи в режиме прямого доступа, выполняемые за каждый цикл выполнения программы 1 для удобства контроля процесса передачи осциллографом.

8. Зацикливает выполнение программы (переход на пункт 1) для предоставления возможности контроля состояния линий интерфейса с помощью обычного электронного осциллографа. Выход из цикла осуществляется комбинацией клавиш Ctrl+Esc. При однократном выполнении программы вместо перехода на начало программы необходимо применить команду INT3 - возврат в AFD.

Текст программы 1

```
A:      MOV     DX, 077A    ;Установка режима Bi-directional порта ECP с
        MOV     AL, 34      ;запретом служебных прерываний и DMA
        OUT     DX, AL
        MOV     DX, 037A    ;Установка порта в режим передачи
        MOV     AL, 04      ;1284Aktive - H-уровень и Init# (ReverseRequest# в
        OUT     DX, AL      режиме ECP) - H-уровень
        MOV     DX, 000B    ;Адрес регистра режима DMA1 заносится в DX
        MOV     AL, 4B      ;Установка режима работы канала 3 DMA: -
        OUT     DX, AL      ;одиночный вывод через канал 3

                            ;Установка счетчика адреса и регистра страниц на адрес
                            блока памяти из которого будут выдаваться в порт данные в режиме DMA:
        MOV     AX, DS      ;Содержимое DS пересылается в AX
        MOV     BX, 10      ;Множитель 10h (16) записывается в BX
        MOV     DX, 0000    ;Обнуляем регистр DX
        MUL     BX          ;Умножение содержимого AX на 16 (сдвиг влево на 4
                            разряда с переносом в DX четырех старших разрядов
        OUT     [06],AL     Младший байт адреса в счетчик адреса канала 3 DMA1
        MOV     AL, AH      ;Запись второго байта адреса в счетчик адреса
        OUT     [06], AL    ;канала 3 DMA1
        MOV     AL, DL      ;Третий байт адреса из DL пересылается а AL
        OUT     [82], AL    ;и записывается в регистр страниц канала 3 DMA

        MOV     DX, 0007    ;Запись количества циклов (слов) DMA(ПДП)
        MOV     AL, 0F      ;в счетчик числа циклов канала 3 DMA1 (16 циклов)
        OUT     DX, AL      ;младший байт
        MOV     AL, 00
        OUT     DX, AL      ;старший байт
        MOV     DX, 000A    ;Сброс маски канала 3 DMA(ПДП) записью константы
        MOV     AL, 03      ;03 по адресу 00Ah контроллера DMA1
        OUT     DX, AL
        MOV     DX, 077A    ;Установка режима ECP с разрешением
        MOV     AL, 78      ;обмена по каналу DMA (ПДП) (инициируется обмен
        OUT     DX, AL      ;через порт в режиме DMA)
        MOV     DX, 0008    ;Тест на заполнение счетчика канала 3 DMA(ПДП)
C:      IN      AL, DX      ;чтением регистра состояния каналов DMA1
        TEST    AL, 08      ;и проверкой разряда конца циклов канала 3 DMA1
        JZ      C
        MOV     DX, 077A    ;Режим ECP с запрещением
```

```
        MOV     AL, 74      ;обмена по каналу DMA(ПДП)
        OUT     DX, AL
        MOV     CX, FFFF    ;Формирование величины программной задержки
B:      LOOP    B           ;Программная задержка

        JMP     A           ;Зацикливание программы; выход из цикла - Ctrl+Esc
                            ;При однократном выполнении программы вместо
                            ;команды JMP A необходимо применить команду INT 3
                            ;- возврат в AFD
```

Программа 2.(приемник) реализует следующие функции:

1. Настраивает адресацию сегмента данных для размещения в нем информации, принимаемой из параллельного порта.

2. Настраивает параллельный порт на режим приема данных занесением в регистр CR (адрес 37Ah) константы 24h (см. разделы 1.1.2, 1.3.4.1. и 2.1.2)

3. Устанавливает режим ECP без поддержки DMA занесением константы 74h (см. табл. 1.8 и описание регистра ECR в разделе 1.3.4.2) в регистр ECR по адресу 77Ah.

4. Определяет факт заполнения входного буфера анализируя содержимое разряда ECR.1, если ECR.1=1, то буфер заполнен (см. раздел 1.9.2.1 и 1.3.4.2). Если буфер не заполнен, то организуется цикл ожидания заполнения буфера.

5. Организует чтение данных из буфера и занесение их в блок памяти по адресу ES:[DI]. Затем, после чтения очередного байта, содержимое регистра DI инкрементируется на единицу.

6. После прочтения 16 байт управление передается на начало программы. Выход из бесконечного цикла осуществляется по комбинации клавиш Ctrl+Esc. При однократном выполнении программы вместо перехода на начало программы необходимо применить команду INT 3 - возврат в AFD.

Текст программы 2

```
A0:     MOV     DI, 0000    ;Обнуляем регистр DI
        MOV     AX, DS      ;Запоминаем значение
        MOV     ES, AX      ;регистра DS в ES
        MOV     DX, 037A    ;Настройка на
        MOV     AL, 24      ;прием данных
        OUT     DX, AL
        MOV     DX, 077A    ;Установка режима ECP
        MOV     AL, 74      ;без поддержки DMA(ПДП)
        OUT     DX, AL
A:      MOV     DX, 077A
        IN      AL, DX      ;Тест на заполнение буфера (проверка первого
        TEST    AL, 02      ;разряда регистра ECR)
        JZ      A
        MOV     DX, 0778
B:      IN      AL, DX      ;Читаем данные из буфера порта
        MOV     ES:[DI], AL ;и заносим по адресу ES:[DI]
        INC     DI
        TEST    DI, 0010    ;Тест DI (выполнение 16 циклов чтения)
        JZ      B
        JMP     A0          ;Зацикливание программы. Выход из цикла: Ctrl+Esc
                            ;При однократном выполнении программы вместо
                            ;команды JMP A необходимо применить команду INT 3
                            ;- возврат в AFD
```

8. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

При подготовке к лабораторным работам и к их защите рекомендуется пользоваться информацией, представленной в описаниях лабораторных работ, в основном разделе учебно-методического пособия и в дополнительной литературе [1, 3, 6, 10].

9. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Можно ли в качестве макетного устройства для ECP порта, работающего в режиме DMA использовать ПК с SPP портом?
2. Можно ли обмениваться командной информацией через ECP порт в режиме DMA?
3. Охарактеризуйте этапы процедуры инициализации канала DMA.
4. По каким событиям может быть прекращен обмен данными при работе канала DMA в режиме передачи по требованию?
5. Какие режимы обмера может поддерживать канал DMA контроллера i8237A?
6. Сколько и каких каналов DMA поддерживает подсистема DMA PC совместимых компьютеров?
7. Каким образом подсистема DMA PC/AT формирует 24-разрядный адрес памяти?
8. Охарактеризуйте регистры контроллера DMA, входящие в состав каждого канала.
9. Какая система приоритетов каналов DMA используется в подсистеме DMA PC совместимых компьютеров?
10. По каким событиям инициируется обмен через ECP порт в режиме DMA?
11. Охарактеризуйте особенности работы ECP порта в режиме DMA.

10. ПРИЛОЖЕНИЕ

10.1.Подсистема контроллера DMA IBM PC/AT.

Архитектура PC AT предполагает наличие важного компонента: программируемых системных устройств, которые обеспечивают поддержку не только функций DOS и BIOS, но и многих прикладных и системных программ.

Знание принципов работы этих устройств и основных правил их программирования поможет написать корректную и, главное, совместимую программу.

Прямой доступ к памяти (DMA) - это метод непосредственного обращения к памяти, минуя процессор. Процессор отвечает только за программирование DMA: настройку на определенный тип передачи, задание начального адреса и размера массива обмениваемых данных. Обычно DMA используется для обмена массивами данных между системной памятью и устройствами ввода-вывода.

Обмен данными между процессором и устройствами ввода-вывода осуществляется по системной шине, "хозяином" которой является процессор. При использовании контроллера DMA на время обмена данными он должен получить управление системной шиной, т.е. стать ее "хозяином". По окончании обмена подсистема DMA возвращает процессору право управления шиной.

Архитектура процессора PC AT включает в себя подсистему DMA, состоящую из двух контроллеров DMA Intel 8237, регистра старшего адреса DMA и регистров страниц DMA. Эти контроллеры обеспечивают 7 каналов DMA. Система обеспечивает передачу данных по каналам DMA как по одному байту за цикл DMA, так и по два байта за цикл, исходя из возможностей архитектуры процессора (двухбайтной шины данных). Чтобы сохранить преемственность подсистемы DMA в PC AT с аналогичной подсистемой в PC XT каскадирование "байтного" контроллера DMA с распределением каналов XT осуществляется через "словный" контроллер DMA. В связи с тем, что в архитектуре PC AT признано целесообразным использование строковых команд ввода-вывода МП 80286, для обмена с адаптером НМД канал байтного DMA, предназначавшийся для подключения адаптера НМД, стал резервным.

Упрощенная схема подсистемы DMA в PC AT показана на рис. 2.

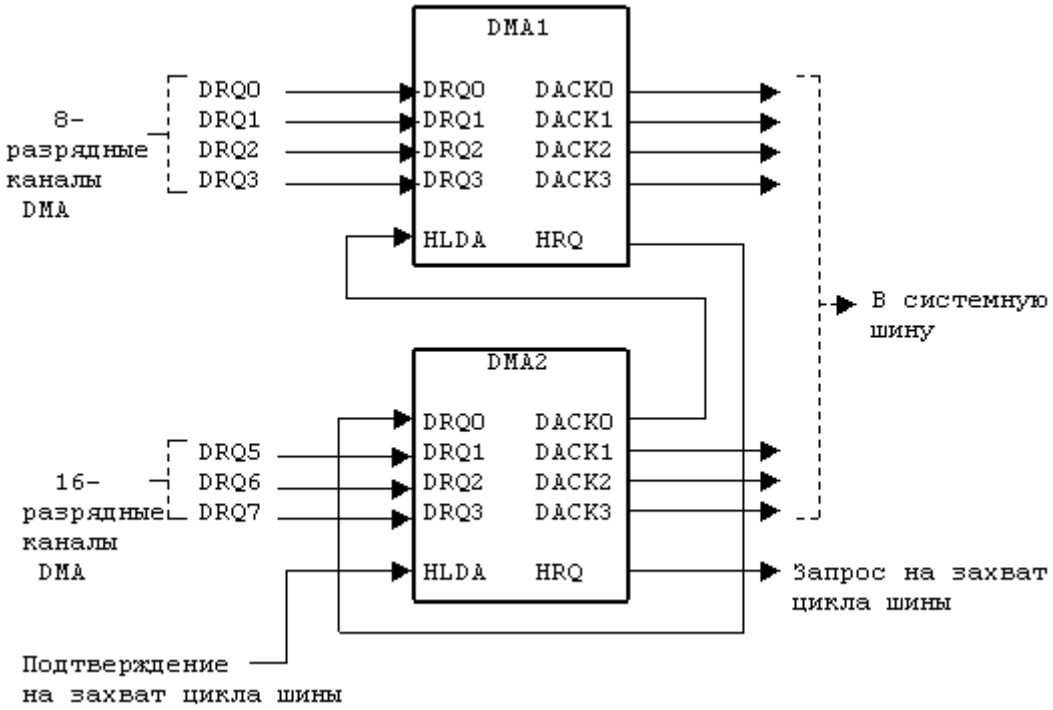


Рис.2. Подсистема DMA в PC AT

Для осуществления прямого доступа к памяти контроллер должен выполнить ряд последовательных операций:

- принять запрос (DREQ) от устройства ввода-вывода;
- сформировать запрос (HRQ) в процессор на захват шины;
- принять сигнал (HLDA), подтверждающий захват шины;
- сформировать сигнал (DACK), сообщающий устройству о начале обмена данными;
- выдать адрес ячейки памяти, предназначенной для обмена;
- выработать сигналы (MEMR, IOW или MEMW, IOR), обеспечивающие управление обменом;
- по окончании цикла DMA либо повторить цикл DMA, изменив адрес, либо прекратить цикл.

10.1.1. Формирование адреса памяти

Контроллеры DMA обеспечивают формирование только 16 младших разрядов адреса памяти. Причем старшая часть адреса (A15-A8 для DMA1 или A16-A9 для DMA2) во время цикла DMA по шине данных поступает в регистр старшего адреса DMA и далее на шину адреса, а младшая часть адреса (A7-A0 для DMA1 или A8-A1 для DMA2) выдается на шину адреса непосредственно из контроллера. Восемь старших разрядов адреса памяти содержатся в регистре страниц DMA. Разряд A16 из регистра страниц DMA запрещается, когда выбран DMA2. Разряд A0 не связан с DMA2 и всегда содержит нуль при передаче слова. Это означает, что:

- размер блока данных, который может быть передан или адресован, измеряется не байтами (8 бит), а словами (16 бит);
- слова всегда должны быть расположены на четной границе.

Таким образом, контроллер DMA и регистр страниц определяют 24-разрядный адрес, что обеспечивает передачу данных в пределах адресного пространства 16 М байт.

На рис.3 показана одна из возможных схем реализации подсистемы DMA и распределения адресных потоков для формирования адреса при обмене данными. Формирование адреса памяти при передаче байта и при передаче слова показано ниже.

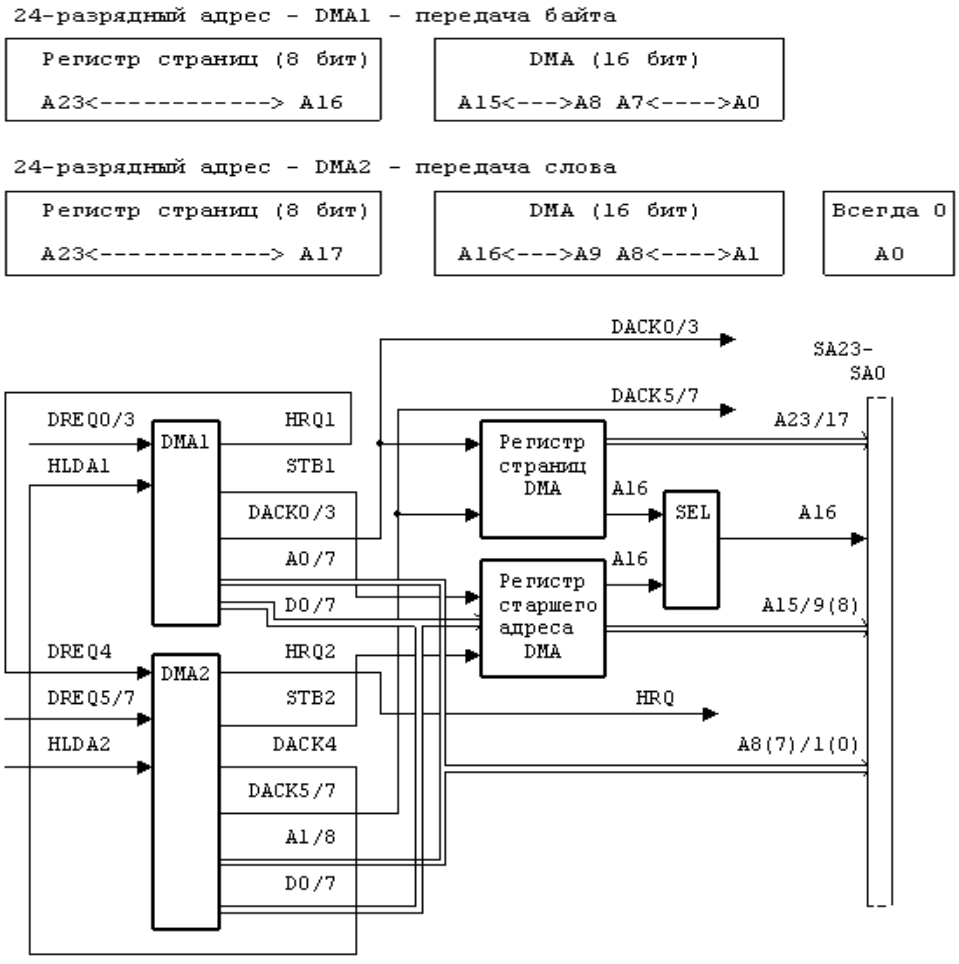


Рис.3. Подсистема DMA и распределение адресных потоков

10.1.2. Каналы контроллеров DMA

Каналы контроллера 1 (0, 1, 2 и 3) предназначены для выполнения побайтных передач DMA с 8-разрядными портами (блоками до 64 К байт). Из четырех каналов контроллера 2 (4, 5, 6, 7) каналы 5, 6 и 7 предназначены для обмена словами с 16-разрядными портами (блоками до 128 К байт). В отличие от микропроцессора подсистема DMA передает слова только по четной границе адреса. Канал 4 используется для каскадного объединения с контроллером 2. Таким образом, все запросы каналов 0-3 контроллера 1 обрабатываются через канал 4 контроллера 2. Это приводит к тому, что все каналы контроллера 1 (каналы 0, 1, 2 и 3) обладают более высоким приоритетом по сравнению с каналами контроллера 2 (каналы 5, 6, 7). Распределение каналов контроллеров DMA приведено в табл. 2.

Таблица. 2. Распределение каналов контроллеров DMA

Канал	Использование в AT
0*	Резерв
I*	Адаптер интерфейса SDLC
2*	Адаптер НГМД
3*	Резерв
4	Используется для каскадирования контроллеров
5**	Резерв
6**	Резерв
7**	Резерв

* 8-разрядные каналы.

** 16-разрядные каналы.

10.1.3. Регистры страниц

Регистр страниц находятся в памяти и содержит восемь старших разрядов 24-разрядного адреса. Вместе с контроллерами DMA он определяет полный (24-разрядный) адрес для каналов DMA. В табл. 3 приведены адреса портов регистров страниц для каналов DMA.

Таблица. 3. Адреса портов регистров страниц

Канал DMA	Адрес порта регистра страниц
0	087h
1	083h
2	081h
3	082h
4i	-
5	08Bh
6	089h
7	08Ah
Регенерация	08Fh *

* Содержимое регистра страниц в целях регенерации должно быть равно 00h.

10.1.4. Адресация портов

В табл. 4 приведены адреса портов - регистров адреса ОП и управления/состояния контроллеров DMA, а также форматы регистров.

Таблица. 4. Адреса портов

Функции регистров	Формат	Адреса портов		Чтение/запись
		DMA1	DMA2	
Регистр состояния (STAT) (Read Status Register) Регистр команд (CR) (Write Command Register)	8 8	008h	0D0h	Чтение Запись
Регистр режима (MOD) (Write Mode Register) Регистр режима (MOD) *** (Read Mode Register)	6	00Bh	0D6h	Запись Чтение
Запись одиночных разрядов регистра маски (Write Single Mask Register) Регистр команд (CR) *** (Read Command Register)	4 8	00Ah	0D4h	Запись Чтение
Запись всех разрядов маски (Write Mask Register) Регистр маски (MASK) *** (Read Mask Register)	4	00Fh	ODEh	Запись Чтение
Программный регистр запросов (REQ)** (Write Request Register) Регистр запросов *** (Read Request Register)	4	009h	0D2h	Запись Чтение

Базовый и текущий регистры адреса - канал 0	16	00h	0C0h	Запись
Текущий регистр адреса -канал 0	16	000h	0C0h	Чтение
Базовый и текущий регистры счетчика - канал 0	16	00lh	0C2h	Запись
Текущий регистр счетчика - канал 0	16	00lh	0C2h	Чтение
Базовый и текущий регистры адреса - канал 1	16	002h	0C4h	Запись
Текущий регистр адреса - канал 1	16	002h	0C4h	Чтение
Базовый и текущий регистры счетчика - канал 1	16	003h	0C6h	Запись
Текущий регистр счетчика - канал 1	16	003h	0C6h	Чтение
Базовый и текущий регистры адреса - канал 2	16	004h	0C8h	Запись
Текущий регистр адреса -канал 2	16	004h	0C8h	Чтение
Базовый и текущий регистры счетчика - канал 2	16	005h	0CAh	Запись
Текущий регистр счетчика - канал 2	16	005h	0CAh	Чтение
Базовый и текущий регистры адреса - канал 3	16	006h	0CCh	Запись
Текущий регистр адреса - канал 3	16	006h	0CCh	Чтение
Базовый и текущий регистры счетчика - канал 3	16	007h	0CEh	Запись
Текущий регистр счетчика - канал 3	16	007h	0CEh	Чтение
Временный регистр данных (TR)** (Read Temporary Register)	16	00Dh	0DAh	Чтение
Общий сброс (Master Clear)	*			Запись
Сброс F/F (Clear Byte Pointer Flip-Flop)	*	00Ch	0D8h	Запись
Установка F/F *** (Set Byte Pointer Flip-Flop)				Чтение
Сброс регистра маски (Clear Mask Register)	*	00Eh	0DCh	Запись
Сброс счетчика MODE *** (Clear Mode Counter)				Чтение

* Это не регистры, а непосредственные команды для контроллера DMA.

** Эти регистры используются только в режиме ПАМЯТЬ-ПАМЯТЬ.

*** Эти регистры и команды контроллера DMA не реализованы в контроллере 8237A и в "Периферийном контроллере" STC62C008, но реализованы в большинстве более современных комплектов процессорных БИС.

10.1.5. Передачи DMA

Подсистема DMA может осуществлять обмен данными между устройством ввода-вывода и памятью и между различными массивами самой памяти.

Передача данных из памяти в память не реализуется в архитектуре PC AT.

При передаче данных из устройств ввода-вывода в память контроллеры DMA и устройства ввода-вывода используют сигналы DREQx и DACKx для установления связи. Когда устройству ввода-вывода нужно передать байт или слово данных, оно возбуждает свою линию DREQx. После возбуждения контроллером линии DACKx и линии IOR устройство передает свои данные на шину данных для памяти. Когда устройству ввода-вывода нужен байт или слово данных из памяти, оно возбуждает свою линию DREQx. После возбуждения контроллером линии DACKx и линии IOW устройство "забирает" данные памяти из шины данных.

10.2. Контроллер DMA 8237A.

Контроллер DMA используется для прямого доступа к системной памяти по четырем независимым каналам, минуя процессор. Каждый канал может выполнять до 64 К циклов DMA. Возможно индивидуальное программирование канала на повторение цикла DMA (автоинициализацию). Три основных режима позволяют пользователю программировать тип DMA. Увеличить число каналов можно путем присоединения к основному контроллеру дополнительных контроллеров DMA (каскадирования). Режим работы каждого канала может программироваться индивидуально.

Структурная схема контроллера 8237A представлена на рис. 4.

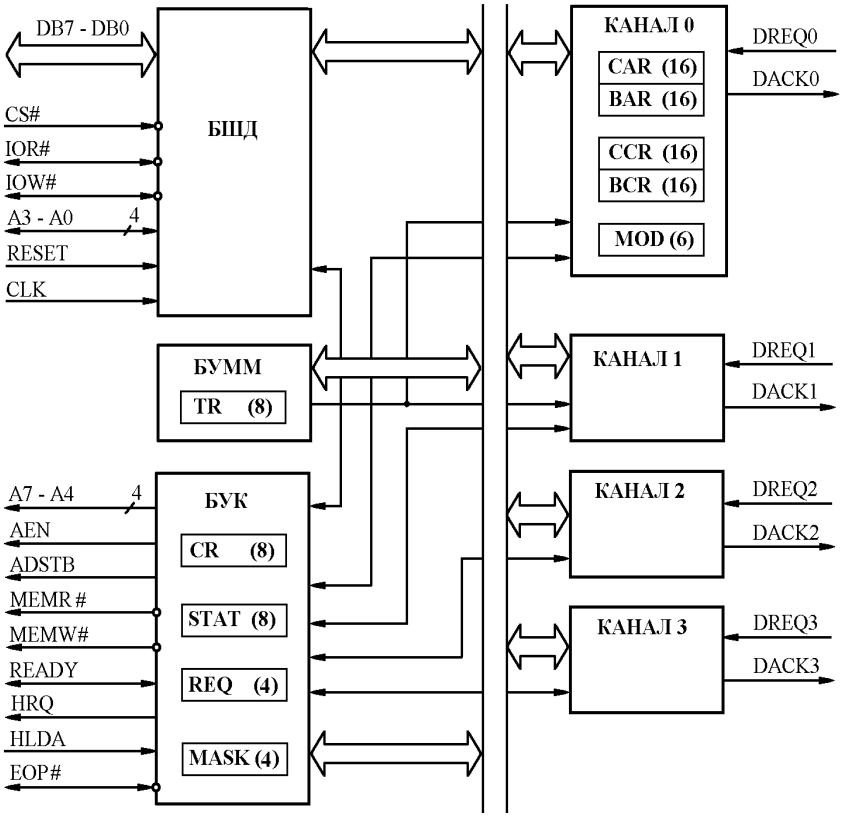


Рис. 4. Структурная схема контроллера DMA.

10.2.1. Функциональное назначение выводов контроллера DMA

В табл. 5 приведено обозначение выводов контроллера DMA и их функциональное назначение.

Таблица. 5. Обозначение выводов контроллера DMA

Обозначение вывода	Тип вывода	Функциональное назначение
CLK	Вход	ВХОД СИНХРОСИГНАЛА: синхронизирует внутренние операции, управляет скоростью передачи данных
CS#	Вход	ВЫБОР КРИСТАЛЛА: активен низким. Выбирает контроллер DMA в качестве устройства ввода-вывода в течение холостого цикла (S1). Разрешает связь с микропроцессором по шине данных
RESET	Вход	СБРОС: активен высоким. Сбрасывает регистры команды, состояния, запроса и временный, а также триггер FF и устанавливает регистр маски. После сигнала сброса контроллер DMA находится в холостом цикле
READY	Вход	ГОТОВО: вход используется для расширения импульсов чтения и записи из контроллера DMA при согласовании с медленной памятью или периферийным устройством. READY не должен изменяться в течение периода времени, необходимого для его установки/захвата. Активен высоким
HLDA	Вход	ПОДТВЕРЖДЕНИЕ ЗАХВАТА: активен высоким. Сигнал из CPU указывает, что CPU освободил системную шину
DREQ0, DREQ1, DREQ2, DREQ3	Вход	ЗАПРОС DMA: шина, по которой поступают индивидуальные асинхронные запросы от периферийных устройств на получение обслуживания подсистемы DMA. При фиксированном приоритете DREQ0 имеет высший приоритет, DREQ3 - низший. Запрос вырабатывается путем активизации шины DREQ соответствующего канала. DACK подтверждает распознавание запроса DREQ. Полярность DREQ программируется. Инициализация по RESET делает эту шину активной по высокому уровню. DREQ должен поддерживаться активным до тех пор, пока не активизируется соответствующий DACK

DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7	Вход-выход	ШИНА ДАННЫХ: это двунаправленная шина с тремя состояниями, связанная с системной шиной. Выходы разрешаются при программировании во время цикла чтения для вывода содержимого адресного регистра, регистра состояния, временного регистра или регистра счетчика слов в CPU. Выходы блокируются, а входы считываются во время цикла записи, когда управляющие регистры контроллера DMA программируются CPU. Во время циклов DMA старшие 8 разрядов адреса выдаются на шины данных и стробируются ADSTB во внешние регистры-защелки. В операциях ПАМЯТЬ-ПАМЯТЬ данные из памяти поступают в контроллер DMA по шине данных во время передачи ЧТЕНИЕ ПАМЯТИ. При передаче ЗАПИСЬ ПАМЯТИ эти данные размещаются в новых ячейках памяти
IOR#	Вход-выход	ЧТЕНИЕ ВВОДА-ВЫВОДА: активен низким уровнем. Это двунаправленная шина с тремя состояниями. В холостом цикле - это входной управляющий сигнал, используемый CPU для чтения управляющих регистров. В активном цикле - это выходной управляющий сигнал, используемый контроллером DMA для выборки данных из периферийного устройства во время передачи ЗАПИСЬ DMA
IOW#	Вход-выход	ЗАПИСЬ ВВОДА-ВЫВОДА: активен низким уровнем. Это двунаправленная шина с тремя состояниями. В холостом цикле это входной управляющий сигнал, используемый CPU для загрузки информации в контроллер DMA. В активном цикле - это выходной управляющий сигнал, используемый контроллером DMA для загрузки данных в периферийное устройство во время передачи ЧТЕНИЕ DMA
EOP#	Вход-выход	ОКОНЧАНИЕ ПРОЦЕССА: двунаправленный сигнал, активен низким. Сигнал возникает, когда завершается обслуживание подсистемы DMA. Контроллер DMA разрешает внешнему сигналу завершить обслуживание подсистемы DMA. Это происходит путем передачи сигнала низкого уровня на вход EOP#. Контроллер DMA вырабатывает сигнал, когда заканчивается окончание счета (TC) какого-либо канала. В этом случае сигнал EOP# передается на выход. Выработка EOP#, как внутреннего, так и внешнего, заставляет контроллер DMA завершить обслуживание, сбросить запрос и, если разрешена автоинициализация, записать базовые регистры в текущие регистры канала. Разряд маски и разряд TC регистра состояния устанавливаются по EOP# для текущего активного канала, если канал не программируется для автоинициализации. В этом случае разряд маски сбрасывается. Во время передачи ПАМЯТЬ-ПАМЯТЬ EOP# передается на выход, когда устанавливается TC для канала 1. EOP# следует соединить с высоким уровнем через резистор, если он не используется как вход, чтобы предотвратить ошибочные входные сигналы окончания процесса
A0, Al, A2, A3	Вход-выход	АДРЕС: эти 4 двунаправленных сигнала с тремя состояниями являются младшими разрядами адресной шины. В холостом цикле они являются входами и используются CPU для адресации управляющих регистров DMA, чтобы загрузить или считать их. В активном цикле они являются выходами и выдают 4 младших адресных разряда
A4, A5, A6, A7.	Вход-выход	АДРЕС: эти 4 выходных сигнала с тремя состояниями являются старшими разрядами адресной шины. Они выдаются только во время обслуживания DMA
HRQ	Выход	ЗАПРОС ЗАХВАТА: этот сигнал выдается в CPU и используется для управления захватом системной шины. Если соответствующий разряд маски сброшен, то наличие любого действительного DREQ заставляет DMA выдать HRQ. После выдачи HRQ требуется по крайней мере хотя бы один цикл синхронизации (TCY), чтобы появился активный HLDA
DACK0, DACK1, DACK2, DACK3	Выход	ПОДТВЕРЖДЕНИЕ DMA: эти сигналы используются для сообщения индивидуальной периферии, что одной из них разрешается цикл DMA. Активное значение этих сигналов программируется. Инициализация по сбросу, устанавливает для них низкий активный уровень

AEN	Выход	РАЗРЕШЕНИЕ АДРЕСА: этот выход разрешает выдать содержимое 8-разрядной защелки (8 старших разрядов адреса) на системную адресную шину. AEN используется в этом случае для запрещения использования системной шины другим устройством во время передач DMA. Активен высоким
ADSTB	Выход	СТРОБ АДРЕСА: активен высоким. Используется для стробирования старшего адресного байта во внешние защелки.
MEMR#	Выход	ЧТЕНИЕ ПАМЯТИ: выходной сигнал с тремя состояниями. Активен низким. Используется для выборки данных из заданной ячейки памяти во время передачи ЧТЕНИЕ DMA или передачи ПАМЯТЬ-ПАМЯТЬ
MEMW#	Выход	ЗАПИСЬ ПАМЯТИ: выходной сигнал с тремя состояниями. Активен низким. Используется для записи данных в заданную ячейку памяти во время передачи ЗАПИСЬ DMA или передачи ПАМЯТЬ-ПАМЯТЬ
PINS	Вход	Этот вход должен быть всегда высоким

10.2.2. Циклы DMA

Подсистема DMA предназначена для работы в двух основных циклах: холостом и активном. Каждый цикл - это совокупность некоторого количества ее состояний. Подсистема DMA может иметь семь состояний, каждое из которых соответствует одному периоду синхронизации.

Состояние SI - неактивное состояние. Подсистема DMA находится в этом состоянии, когда нет действительных запросов на обслуживание подсистемы DMA. Но в этом состоянии подсистема DMA может программироваться.

Состояние S0 - первое состояние обслуживания подсистемы DMA, возникающее по действительному запросу (DREQ). Подсистема DMA запрашивает разрешение у CPU на захват системной шины (HRQ), но до получения подтверждения захвата (HLDA) может еще программироваться. По HLDA подсистема DMA переходит в рабочее состояние.

Состояния S1, S2, S3 и S4 - рабочие состояния. Начинают вырабатываться после получения подтверждения из CPU.

Состояние SW - состояние ожидания. Необходимо, если для завершения передачи требуется больше времени, чем предполагается обычно. SW могут "вклиниваться" между S2 или S3 и S4, используя вход READY DMA.

10.2.3. Режимы обслуживания

В активном цикле обслуживание подсистемы DMA возможно в одном из четырех режимов. Окончание обслуживания распознается по переходу регистра счетчика слов из 0000H в FFFFH. При этом возникает сигнал окончания счета (TC), который может вызвать автоинициализацию, если она запрограммирована, или маскирование канала при ее отсутствии. Одновременно с TC вырабатывается выходной сигнал -EOP.

Во время автоинициализации первоначальные значения регистров текущего адреса и счетчика восстанавливаются из соответствующих базовых регистров. После автоинициализации канал готов выполнять другое обслуживание подсистемы DMA без вмешательства CPU, как только обнаружится достоверный DREQ.

10.2.3.1. Режим одиночной передачи (Single Transfer Mode)

В этом режиме контроллер DMA выполняет только одну передачу. Адрес и счетчик слов будут изменяться при каждой передаче. DREQ должен быть активным, пока не активизируется соответствующий DACK. Если DREQ активен на протяжении одиночной передачи, HRQ переходит в неактивное состояние по выполнении одной передачи и освобождает шину системе. HRQ снова станет активным (при активном DREQ) и по получении нового HLDA будет выполняться следующий цикл одиночной передачи. Это гарантирует CPU в системе выполнение одного полного машинного цикла между передачами DMA.

10.2.3.2. Режим передачи блока (Single Transfer Mode)

В этом режиме передается блок информации во время обслуживания подсистемы DMA. DREQ должен быть активным, пока не появиться активный DACK.

10.2.3.3. Режим передачи по требованию (Demand Transfer Mode)

В этом режиме передача данных выполняется до тех пор, пока не появится TC или внешний EOP#, либо когда DREQ станет неактивным. Таким образом, передачи могут продолжаться до тех пор, пока периферийное устройство не исчерпает объем данных.

10.2.3.4. Каскадный режим (Cascade Mode)

Этот режим использует объединение нескольких контроллеров DMA для расширения числа подключаемых каналов.

Выходы HRQ и входы HLDA от дополнительных контроллеров соединяются соответственно со входами DREQ и выходами DACK первичного контроллера DMA. Это дает возможность запросам от дополнительного устройства распространяться через сеть приоритетных цепей предшествующего устройства.

Таким образом, канал первичного контроллера DMA, к которому подключен дополнительный контроллер, программируется на выполнение каскадного режима и служит только для определения приоритета дополнительного устройства и транзита сигналов HRQ в CPU и HLDA из CPU. Все другие сигналы каскадного канала первичного контроллера DMA в формировании циклов подсистемы DMA не участвуют.

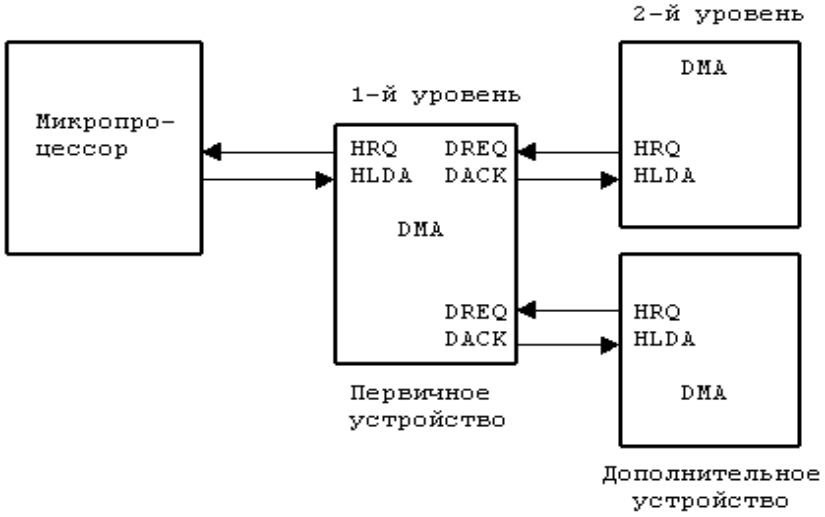


Рис. 5. Каскадное объединение контроллеров DMA

На рис. 5 показано каскадное объединение контроллеров DMA. Это двухуровневая система. Другие контроллеры DMA могут быть добавлены на остальные каналы первого уровня.

10.2.3.5. Режим память-память

Этот режим предназначен для передачи блоков данных из одного адресного пространства памяти в другое с минимальными программными и временными затратами, но в PC AT не используется. В данном режиме могут работать только нулевой и 1-й каналы контроллера. Эта передача инициируется программной установкой REQ для канала 0. Адрес ячейки памяти-источника данных задают в CAR0, а ячейки-приемника - в CAR1. Байт данных, считанный из памяти, заносится во временный регистр (TR) и затем из TR считывается в ячейку-приемник. Когда значение счетчика слов канала 1 станет равным FFFFh, обслуживание заканчивается.

10.2.4. Типы передач

Каждый из трех активных режимов предполагает выполнение трех различных передач. Это ЧТЕНИЕ, ЗАПИСЬ и ПРОВЕРКА. Передача ЧТЕНИЕ пересылает данные из памяти в периферийное устройство при активизации MEMR# и IOW#. Передача ЗАПИСЬ пересылает данные из периферийного устройства в память при активизации MEMW# и IOR#. ПРОВЕРКА - это псевдопередача. Подсистема DMA осуществляет передачи ЧТЕНИЯ или ЗАПИСИ, генерируя адреса и реагируя на EOP#, но сигналы управления памятью и периферийными устройствами остаются не активными. В режиме ПРОВЕРКИ вход READY игнорируется .

10.2.5. Приоритеты

Подсистема DMA имеет два типа приоритета, которые можно установить программно. Первый из них - фиксированный приоритет, который фиксирует каналы в последовательности, соответствующей убыванию их номеров. Низший приоритет имеет канал номер 3, а высший приоритет имеет канал 0. После выбора какого-либо канала для обслуживания запрещается вмешательство остальных каналов до тех пор, пока обслуживание не завершится.

Второй тип приоритета - циклический (рис. 6). Последний обслуженный канал становится каналом с низшим приоритетом согласно циклу. При циклическом приоритете любое устройство, требующее обслуживания, непременно будет распознано после обработки максимум трех более приоритетных обслуживаний. Это исключает монополизацию всей системы одним каналом.

Во время цикла DMA контроллер для обращения к памяти выдает в CPU младшую часть адреса (старшая часть адреса поступает из регистра страниц DMA). Причем разряды адреса A15-A8 выдаются на шину данных и далее поступают во внешние защелки, из которых они попадают на адресную шину. Разряды адреса A0-A7 выдаются на адресную шину непосредственно из контроллера DMA .

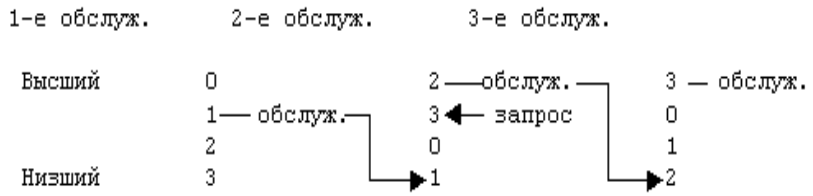


Рис. 6. Циклический приоритет

10.2.6. Генерация адреса

В течение обслуживания режимов передачи блока данных и передачи по требованию генерация адресов должна быть последовательной. Для большого количества передач данные на внешних адресных защелках не изменяются. Эти данные нуждаются в обновлении только при переносе из A7 или заеме из A8 в обычной адресной последовательности, что позволяет вырабатывать состояние S1 только тогда, когда необходимо обновление A8-A15 в защелках. Это означает, что для длительного обслуживания состояние S1 и адресный строб могут возникать только один раз на каждые 256 передач.

10.2.7. Регистры контроллера DMA

В этом параграфе приводится детальное описание регистров контроллера DMA.

10.2.7.1. Регистр текущего адреса (CAR).

Каждый канал имеет 16-разрядный регистр текущего адреса. Этот регистр хранит младшие разряды адреса, используемые во время обслуживания ПДП. Адрес автоматически уменьшается или наращивается после каждой передачи, а промежуточные значения адреса запоминаются в регистре во время передачи. Содержимое текущего регистра записывается или читается микропроцессором побайтно.

10.2.7.2. Регистр текущего счетчика слов (CCR)

Каждый канал имеет 16-разрядный регистр текущего счетчика слов. Этот регистр определяет количество передач, которые необходимо выполнить. Количество передач будет на единицу больше, чем число, программируемое в регистре (т.е. в счетчик заносится число 100, а передач будет 101). Значение счетчика уменьшается после каждой передачи, а промежуточное значение счетчика записывается в регистр во время передачи.

В режиме программирования регистр CCR может быть побайтно записан или прочитан микропроцессором. По концу обслуживания, если не задана автоинициализация, значение счетчика после TC равно FFFFh.

Между обслуживаниями, когда микропроцессору разрешается выполнять операции, промежуточные значения адреса и счетчика слов хранятся в регистрах текущего адреса и счетчика соответствующего канала.

10.2.7.3. Базовые регистры адреса и счетчика (BAR и CAR)

Каждый канал имеет два регистра: регистр базового адреса и регистр базового счетчика. Эти регистры хранят начальные значения соответствующих текущих регистров. При авто инициализации они используются для восстановления первоначальных значений текущих регистров. Базовые регистры загружаются микропроцессором поразрядно одновременно со своими текущими регистрами в режиме программирования. Считать содержимое базовых регистров нельзя.

10.2.7.4. Регистр режима (MOD - 00B, 0D6)

Каждый канал имеет 6-разрядный регистр режима (MOD). Запись в регистр производится в режиме программирования байтом, формат которого представлен на рис. 7. Разряды 0, 1 указывают, в регистр режима какого канала нужно произвести запись, а разряды 2-7 - информацию непосредственно для соответствующего регистра режима. Для регистра режима разрешена только запись информации.

10.2.7.5. Регистр команд (CR - 008, 0D0)

Этот 8-разрядный регистр управляет операциями контроллера DMA. Он программируется микропроцессором и сбрасывается либо по входу RESET либо командой MC (очистка системы). Формат байта записи представлен на рис. 8. Регистр команд не может быть считан микропроцессором.

10.2.7.6. Регистр запроса (REQ; 009, 0D2)

Контроллер DMA может реагировать на запросы по обслуживанию DMA, которые будут инициироваться программно, как и при DREQ. Каждый канал имеет свой разряд в 4-разрядном регистре запроса. Эти разряды немаскируемые, приоритет которых устанавливается шифратором приоритетов. Каждый разряд регистра сбрасывается и устанавливается под управлением программы или очищается при генерации TC или внешнего -EOP.

Весь регистр очищается по RESET. Разряд устанавливается или сбрасывается программой, которая загружает слово данных соответствующего формата, приведенного на рис. 9. Для того чтобы получить программный запрос, канал должен находиться в режиме блочного обмена.

Регистр запроса не может быть считан микропроцессором.

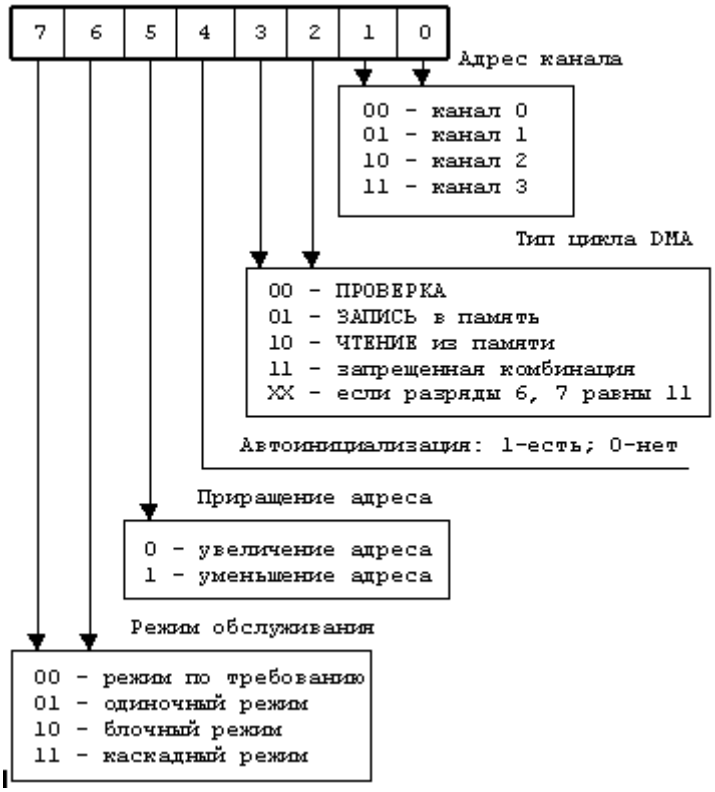


Рис. 7. Формат байта записи в регистр режима

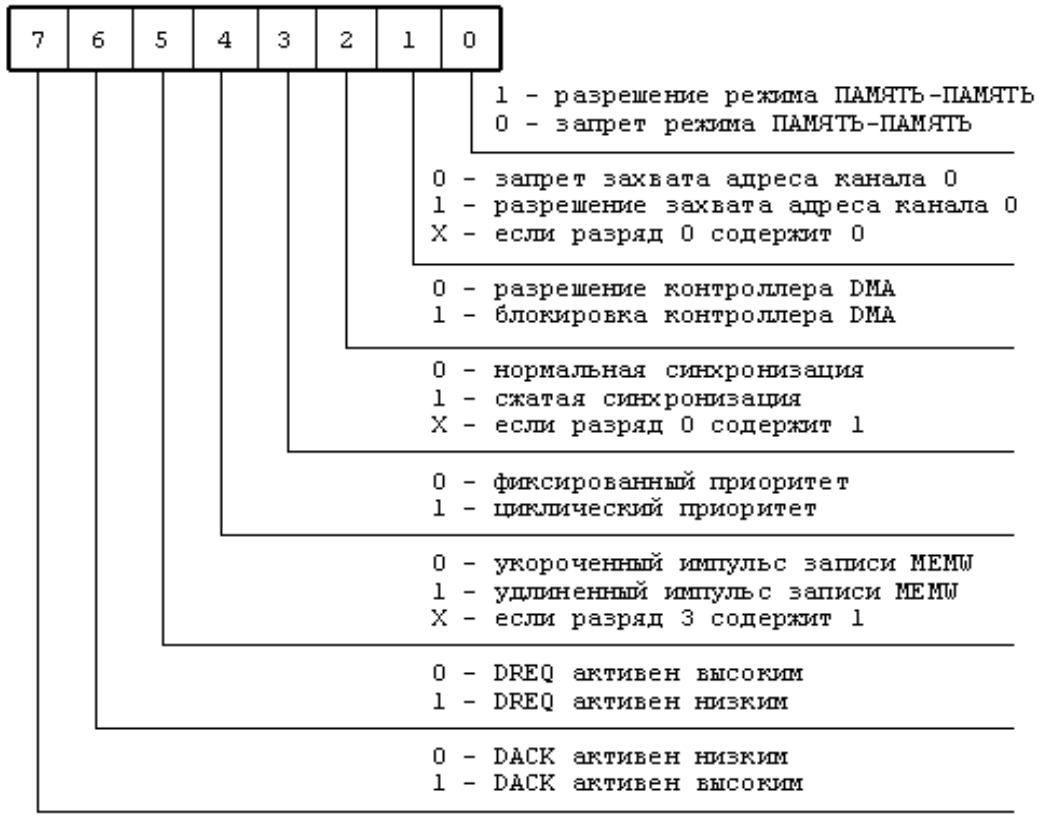


Рис. 8. Формат байта записи в регистр команд

Рис. 9. Формат слова установки разрядов регистра запросов

10.2.7.7. Регистр маски (MASK)

Каждый канал имеет свой разряд в 4-разрядном регистре маски, который может быть установлен, чтобы заблокировать приходящий DREQ. Каждый разряд маски устанавливается, когда связанный с ним канал вырабатывает -EOP, если канал не запрограммирован на автоинициализацию (в этом случае по -EOP маска не устанавливается).

Рис. 10. Формат байта для установки или сброса одного разряда MASK (00A, 0D4)

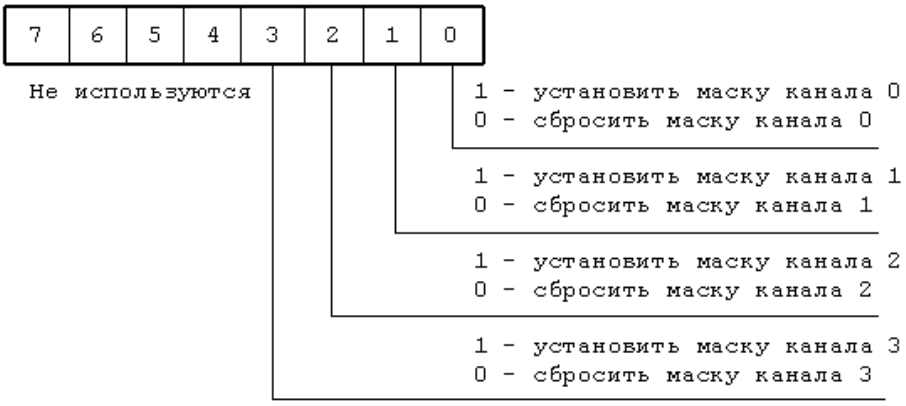


Рис. 11. Формат байта для записи информации во все разряды MASK (00F, 0DE)

Каждый разряд 4-разрядного регистра маски может быть установлен или сброшен под управлением программы. Весь регистр устанавливается по RESET. При этом блокируются все запросы DMA, пока их не разрешит команда сброса регистра маски. Команда раздельной установки/сброса маски подобна команде раздельной установки/сброса регистра запроса. Одновременно все четыре разряда маски могут быть записаны отдельной командой - Write All Mask Register Bits. Регистр маски не может быть считан микропроцессором.

При инициализации регистра маски используются байты с форматами, приведенными на рис. 10, 11.

10.2.7.8. Регистр состояний (STAT - 008, 0D0)

Содержимое регистра состояния может быть прочитано микропроцессором. Регистр содержит информацию о состоянии каналов в данный момент времени. Эта информация показывает, какие каналы достигли заполнения счетчика (т.е. сгенерировали TC) и какие каналы имеют неудовлетворенные запросы (рис. 12). Разряды 0-3 регистра устанавливаются каждый раз, как появляется TC для соответствующего канала или внешний -EOP. Эти разряды сбрасываются по RESET и при каждом чтении регистра состояния.

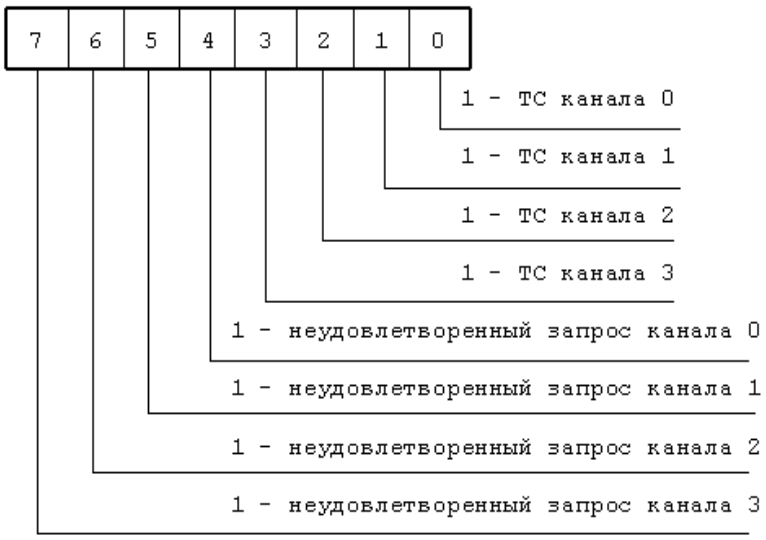


Рис. 12. Регистр состояния

Разряды 4-7 устанавливаются всякий раз, когда соответствующий канал запрашивает обслуживание.

10.2.7.9. Временный регистр (TR - 00D, 0DA)

Регистр используется для хранения данных в течение передачи ПАМЯТЬ-ПАМЯТЬ. После окончания передач последнее слово может быть считано микропроцессором.

Регистр всегда содержит последний байт, переданный в предыдущей операции ПАМЯТЬ-ПАМЯТЬ, если не сброшен по RESET.

10.2.8. Программирование контроллера

Контроллер DMA может программироваться процессором, когда HLDA не активен; это утверждение истинно, даже если активен HRQ. Ответственность процессора заключается в гарантии, что программирование контроллера и активный HLDA взаимоисключающие.

Если -CS и HLDA в низком состоянии, контроллер DMA входит в режим программирования. Выбор регистров при программировании осуществляется с помощью адресной шины A0-A3, работающей на вход, а выбор режима записи или считывания - с помощью сигналов -IOW или -IOR. При записи/считывании 16-разрядных регистров адреса или счетчика соответствующего канала необходим дополнительный разряд адреса, в качестве которого используется внутренний триггер FF. По FF=0 происходит обращение к младшему байту, а по FF=1 - к старшему байту регистра адреса или счетчика.

Кроме того, для контроллера существуют дополнительные специальные команды, которые могут быть выполнены в режиме программирования и не зависят от набора разрядов на шине данных. К ним относятся следующие команды:

1) сброс триггера FF (Clear First/Last Flip-Flop; 00C, 0D8).

Эта команда выполняется перед записью или чтением информации из регистров адреса или счетчика слов контроллера DMA. Триггер FF устанавливается таким образом, чтобы микропроцессор адресовал старший и младший байты в нужной последовательности;

2) очистка (Master Clear; 00D, 0DA).

По этой программной команде в контроллере DMA выполняются такие же действия, как и по аппаратному RESET: Очищаются регистры команд, состояния, запросов и триггер FF, а регистр маски устанавливается. После этого контроллер DMA переходит в холостой цикл;

3) сброс регистра маски (Clear Mask Register; 00E, 0DC).

По этой команде очищаются разряды масок для всех 4 каналов, что разрешает им принимать запросы DMA.

После включения питания предполагается, что все внутренние ячейки, особенно регистр MOD, будут загружены некоторым действительным значением. Это необходимо сделать, даже если некоторые каналы не используются.

Необходимые условия для программирования и управления подсистемой DMA приведены в табл. 6.

Таблица. 6. Условия для программирования и управления подсистемой DMA

FF	IOR	IOW	Адрес	A3	A2	A1	A0	Операция
0	1	0	0h	0	0	0	0	Запись мл. байта CAR и BAR канала 0
1	1	0	0h	0	0	0	0	Запись ст. байта CAR и BAR канала 0
0	0	1	0h	0	0	0	0	Чтение мл. байта CAR канала 0
1	0	1	0h	0	0	0	0	Чтение ст. байта CAR канала 0

O	1	0	1h	0	0	0	1	Запись мл. байта CCR и BCR канала 0
1	1	0	1h	0	0	0	1	Запись ст. байта CCR и BCR канала 0
0	0	1	1h	0	0	0	1	Чтение мл. байта CCR канала 0
1	0	1	1h	0	0	0	1	Чтение ст. байта CCR канала 0
0	1	0	2h	0	0	1	0	Запись мл. байта CAR и BAR канала 1
1	1	0	2h	0	0	1	0	Запись ст. байта CAR и BAR канала 1
0	0	1	2h	0	0	1	0	Чтение мл. байта CAR канала 1
1	0	1	2h	0	0	1	0	Чтение ст. байта CAR канала 1
0	1	0	3h	0	0	1	1	Запись мл. байта CCR и BCR канала 1
1	1	0	3h	0	0	1	1	Запись ст. байта CCR и BCR канала 1
0	0	1	3h	0	0	1	1	Чтение мл. байта CCR канала 1
1	0	1	3h	0	0	1	1	Чтение ст. байта CCR канала 1
0	1	0	4h	0	1	0	0	Запись мл. байта CAR и BAR канала 2
1	1	0	4h	0	1	0	0	Запись ст. байта CAR и BAR канала 2
0	0	1	4h	0	1	0	0	Чтение мл. байта CAR канала 2
1	0	1	4h	0	1	0	0	Чтение ст. байта CAR канала 2
0	1	0	5h	0	1	0	1	Запись мл. байта CCR и BCR канала 2
1	1	0	5h	0	1	0	1	Запись ст. байта CCR и BCR канала 2
0	0	1	5h	0	1	0	1	Чтение мл. байта CCR канала 2
1	0	1	5h	0	1	0	1	Чтение ст. байта CCR канала 2
0	1	0	6h	0	1	1	0	Запись мл. байта CAR и BAR канала 3
1	1	0	6h	0	1	1	0	Запись ст. байта CAR и BAR канала 3
0	0	1	6h	0	1	1	0	Чтение мл. байта CAR канала 3
1	0	1	6h	0	1	1	0	Чтение ст. байта CAR канала 3
0	1	0	7h	0	1	1	1	Запись мл. байта CCR и BCR канала 3
1	1	0	7h	0	1	1	1	Запись ст. байта CCR и BCR канала 3
0	0	1	7h	0	1	1	1	Чтение мл. байта CCR канала 3
1	0	1	7h	0	1	1	1	Чтение ст. байта CCR канала 3
-	0	1	8h	1	0	0	0	Чтение STAT
-	1	0	8h	1	0	0	0	Запись CR
-	1	0	9h	1	0	0	1	Запись REQ
-	1	0	Ah	1	0	1	0	Запись отдельных разрядов MASK
-	1	0	Bh	1	0	1	1	Запись MOD
-	1	0	Ch	1	1	0	0	Сброс триггера FF
-	1	0	Dh	1	1	0	1	Очистка
-	0	1	Dh	1	1	0	1	Чтение TR
-	1	0	Eh	1	1	1	0	Сброс MASK
-	1	0	Fh	1	1	1	1	Запись во все разряды MASK

10.2.9. Пример программирования подсистемы DMA

PC AT использует канал 2 контроллера DMA для обмена с гибким диском. Чтобы получить доступ к верному адресу памяти, необходимо загрузить соответствующий регистр страниц. В данном случае для канала 2 это адрес порта 81h.

В приводимом примере ограничимся рассмотрением установки канала 2 для чтения одного сектора (512 байт) гибкого диска, начиная со смещения внутри страницы, содержащегося в BX. Программа имеет следующий вид:

```
mov     al,46h          ;Канал 2, передача одного байта, чтение с диска
                        (4Ah для записи на диск)
out     dma+0B,al       ;Установить байтный режим
out     dma+0C,al       ;Сбросить триггер FF
mov     al,bl           ;Загрузить смещение адресов в текущий регистр
                        страниц DMA
out     dma+4,al
mov     al,bh
out     dma+4,al
mov     al,0            ;Загрузить счетчик=512
out     dma+5,al
mov     al,2
out     dma+5,al
```

```
out       dma+0A,al     ;Размаскировать канал 2 и читать сектор
```

Примечание: `dma` - это базовый адрес, равный, в данном случае, 00h (ведомый контроллер DMA).

В приложении использованы материалы из [5, 10].

2.7. Лабораторная работа 5 (описание) “Использование параллельных портов для обмена между компьютерами (передача/прием)“

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы “Использование параллельных портов для обмена между компьютерами (передача/прием)“ является ознакомление с принципами работы двунаправленного параллельного порта, принципами его программирования, с приемами непосредственной программной настройки порта на передачу и прием данных и принципами программной реализации различных протоколов взаимодействия и обмена.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА , ИСПОЛЬЗУЕМЫЕ В РАБОТЕ

В лабораторной работе используется ПК Pentium с параллельным портом соответствующим стандарту IEEE 1284, осциллограф С1-81, кроссовая пата и интерфейсные кабели.

3. КРАТКОЕ ОПИСАНИЕ ПРИНЦИПОВ ОБМЕНА ДАННЫМИ ЧЕРЕЗ ПАРАЛЕЬНЫЙ ПОРТ

Для реализации обмена данными между компьютерами через пораллельный порт необходима реализация минимум трех типов протоколов взаимодействия:

1. протокола взаимодействия процессора и контроллера порта;
2. протокола взаимодействия портов между собой на сигнальном уровне через среду интерфейса;
3. протокол взаимодействия между собой программ, обеспечивающих обмен данными между оперативной памятью компьютеров.

Протокол первого типа определяется правилами программной настройки контроллера порта на заданный режим работы и правиами приема и передачи данных через порт в заданных режимах его работы.

Протокол второго типа определяется правилами интерфейса взаимодействия портов и средствами портов, испоьзуемыми дя его реализации. В зависимости от режима работы порта этот протокол сигнального уровня может реализовываться программно через регистры управления и состояния портов (например, в режиме Bi-directional), или аппаратно - средствами контроллера (например, в режимах EPP и ECP).

Протокол третьего типа определяется испоьзуемыми способами синхронизации работы программ передачи и приема (по прерыванию или по опросу фаговых разрядов), способами управления потоком данных и т.д.

В данной лабораторной работе рассматривается организация приема и передачи данных параллельным портом в следующих режимах:

- Bi–directional
- ECP
- ECP с использованием DMA(ПДП)
- EPP в комбинации с Bi–directional.

Для реализации обмена в перечисленных режимах необходимо написание двух программ для каждого режима: программу-передатчик и программу-приемник (для EPP режима - программу ведущего и ведомого комиьютеров). Эти программы должны в том или ином виде реализовывать процедуры всех трех типов протоколов взаимодействия.

4. УКАЗАНИЯ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНОЙ РАБОТЕ

При подготовке к лабораторной работе, ее выполнении и оформлении отчета необходимо следовать рекомендациям раздела 2.2 “Общие указания по выполнению и оформлению лабораторных работ”.

Данная лабораторная работа призвана не только ознакомить студента с принципами и способами организации взаимодействия компьютеров между собой через двунаправленный параллельный порт, но и закрепить те знания и навыки, которые были получены при подготовке, выполнении и защите предыдущих работ этого цикла. Поэтому при подготовке к данной лабораторной работе необходимо повторно изучить описание контроллера параллельного ECP порта (раздел 1.3.4), принципов его программирования (раздел 1.9.2), принципов работы и программирования подсистемы DMA (см. приложение к лабораторной работе 4 - раздел 10). Необходимо также изучить принципы работы и программирования порта в режиме Bi-directional (Byte Mode) (раздел 1.3.2) и в режиме EPP (раздел 1.3.3), а также ознакомиться с примерами программ, приведенными в данном описании.

Перед началом работ необходимо убедиться в том, что средствами BIOS Setup установлен ECP или ECP+EPP режимы работы контроллера параллельного порта с поддержкой DMA (канал 3). Необходимо также соединить порты компьютеров кабелями типа AMAM через соответствующую кроссовую плату, разводка соединений которой для режимов Bi-directional, ECP и ECP с поддержкой DMA, приведена в табл. 2.2 раздела 2.1.2. При работе портов в режиме EPP не возможно непосредственное соединение двух компьютеров даже с использованием кроссовой платы. Это связано с тем, что в режиме EPP выполняются так называемые вложенные циклы ввода/вывода в которых порт всегда выступает ведущим устройством, как в циклах вывода, так и в циклах ввода. В этом случае объединение компьютеров можно выполнить через специальный адаптер, выполняющий функцию ведомого устройства для обоих портов или на одном из компьютеров порт перевести в режим Bi-directional и программно эмулировать ведомое устройство, поддерживающее протокол EPP. Во втором случае для объединения компьютеров можно пользоваться теми же кабелями и кроссовой платой, что и для режимов Bi-directional и ECP.

5. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Подключить к разъему параллельного порта ведущего ПК через соответствующую кроссовую плату и интерфейсные кабели параллельный порт другого компьютера.
2. Отладить программы передачи и приема данных через параллеьные порты в режиме Byte Mode (Bi-directional) (см. раздел 6.1.1 и 6.1.2).
3. Снять осциллограммы сигналов HostBusy (14), HostClc (1), Data (2,3), PerClc (10), PerBusy (11) с привязкой к переднему фронту сигнала HostBusy (14). Сигналы снимать с контактов разъема передающего компьютера, расположенного на кроссовой плате. Номера контактов, на которых наблюдается соответствующий сигнал, приведены в скобках.
4. Сравнить полученные осциллограммы с временными диаграммами, приведенными на рис. 1 и 2.
5. Получить индивидуальное задание от преподавателя и выполнить его (см. раздел 6.1.3 данного описания).
6. Отладить программы передачи и приема через ECP порт без поддержки DMA (раздел 6.2.).
7. По разделу 6.2 получить индивидуальное задание от преподавателя и выполнить его.
8. Отладить программы передачи и приема через ECP порт в режиме DMA (раздел 6.3.).
9. По разделу 6.3 получить индивидуальное задание от преподавателя и выполнить его.
10. Отладить программы, реализующие обмен данными и адресами через EPP порт с компьютером, эмулирующим EPP устройство через порт Bi-directional (раздел 6.4).

11. По разделу 6.4 получить индивидуальное задание от преподавателя и выполнить его.

6. ПРИМЕРЫ ПРОГРАММ

В этом разделе рассмотрены примеры программ, реализующих различные протоколы обмена данными между двумя компьютерами через двунаправленные параллельные порты, работающие в режиме Bi-directional, ECP, ECP с поддержкой DMA и EPP. Для соединения компьютеров через порты во всех случаях используется одна и та же кроссовая плата, разводка которой соответствует табл. 2.2 раздела 2.1.2.

6.1.Обмен данными между ПК в режиме Bi-directional

Передача/прием данных в режиме Bi-directional рассматривается на примере фрагментов программ (программа-передатчик и программа-приемник), реализующих протокол передачи байтов данных через двунаправленный порт Bi–directional из одного ПК в другой с использованием двух типов квитирующих сигналов. Один определяет готовность устройств к обмену. Второй тип сигналов со стороны передатчика сопровождает передаваемые данные и указывает приемнику момент считывания информации с линий данных, а со стороны приемника - подтверждает прием данных приемником (см. рис. 1).

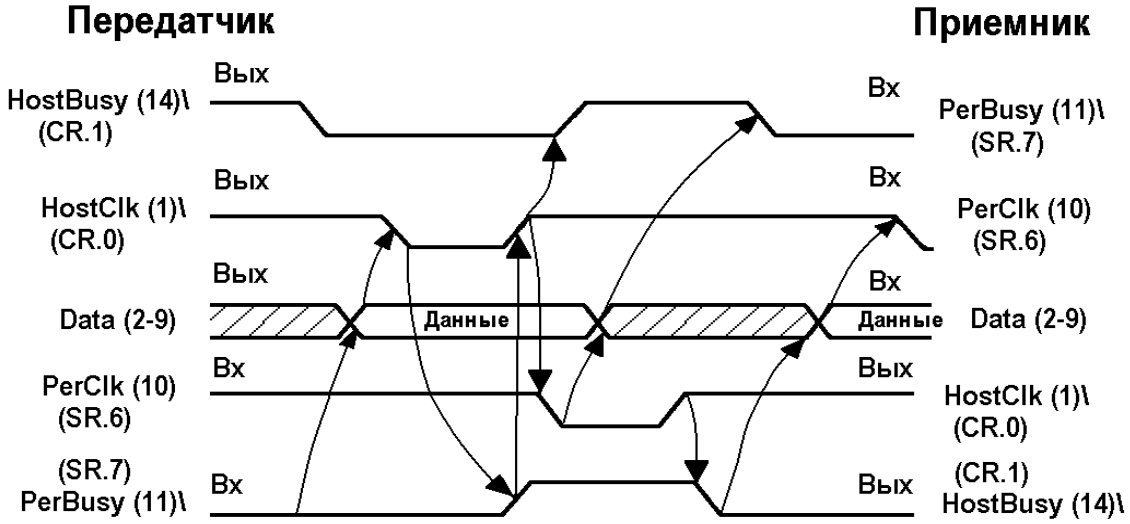


Рис. 1. Временная диаграмма протокоа передачи данныз через двунаправленные параллельные порты между ПК в режиме Bi–directional

6.1.1. Программа-передатчик

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.1 и правилами программирования порта программа-передатчик выполняет следующие действия:

1. Устанавливает Bi-directional режим работы порта записью константы 34h в основной регистр ECR (077Ah)
2. В регистр управления стандартного порта (037Ah) записывает константу 02h – готовность передатчика (низкий уровень сигнала HostBusy на выводе 14 разъема)
3. Тестирует сигнал PerBusy (контакт 11 - состояние занятости прямого канала); разряд SR.7 = 1, если состояние линии PerBusy имеет низкий уровень, т.е. устройство не занято и готово принять очередной байт.
4. Передает байт данных через регистр данных стандартного порта (378h)
5. В регистр управления стандартного порта (037Ah) записывает константу 03h, устанавливая PerClk в низкий уровень (индикация действительности данных на линиях DATA [7:0]) и подтверждая готовность передатчика
6. Ожидание состояния высокого уровня сигнала PerBusy - подтверждение начала процесса приема байта.

7. В регистр управления CR (037Ah) записывает константу 00h – сброс сигнала PerClk - разрешение приема по заднему фронту, и сброс активности передатчика.
8. Ожидание подтверждения приема данных: низкий уровень сигнала PerClk (SR.6=0 ?).
9. Сброс шины данных - занесение 00h в RD.
10. В регистр управления стандартного порта (037Ah) записывает константу 02h – готовность передатчика
11. Переход на пункт 3 - новый цикл приема байта данных.

Примечание: Программа передачи должна запускаться только после того, как на принимаемом компьютере будет запущена программа-приемник.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Программа 1. (Передатчик)

```
        MOV     DX,077A     ;Установка режима Bi-directional
        MOV     AL,34
        OUT     DX,AL
        MOV     DX,037A     ;Устанавливается готовность передатчика
        MOV     AL,02
        OUT     DX,AL
A:      MOV     DX,0379     ;Ожидание появления
        IN      AL,DX       ;низкого уровня сигнала PerBusy -
        TEST    AL,80       ;готовность устройства к приему байта
        JZ      A
        MOV     CX,000F     ;Задержка
B:      LOOP    B
        MOV     DX, 0378    ;Передача данных - байт AAh передается в шину
        MOV     AL, AA      ;данных через регистр DR (адрес 378h)
        OUT     DX, AL
        MOV     DX, 037A    ;Устанавливается низкий уровень HostClk и
        MOV     AL, 03      ;подтверждается низкий уровень HostBusy
        OUT     DX, AL
        MOV     DX, 0379    ;Ожидание высокого уровня сигнала PerBusy,
B1:     IN      AL, DX      ;подтверждающего начало цикла
        TEST    AL, 80      ;приема приемника
        JNZ     B1
        MOV     CX, 001F    ;Программная задержка
B2:     LOOP    B2
        MOV     DX, 037A    ;Переключение HostClk и HostBusy в высокий
        MOV     AL, 00      ; уровень - разрешение приема по заднему фронту
        OUT     DX, AL      ;сигнала HostClk и неготовность передатчика
        MOV     DX, 0379    ;Ожидание низкого уровня сигнала PerClk,
B3:     IN      AL, DX      ;подтверждающего прием байта
        TEST    AL, 40      ;приемником (SR.6=0 ?)
        JNZ     B3
        MOV     DX, 0378    ;Сброс шины данных
        MOV     AL, 00      ;через регистр DR (адрес 378h)
        OUT     DX, AL
        MOV     CX,000F     ;Задержка
C:      LOOP    C
        MOV     DX,037A
        MOV     AL,02       ;Устанавливается готовность передатчика
```

```
        OUT     DX,AL
        JMP     A       ;Переход на новый цикл передачи
```

6.1.2. Программа-приемник

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.1, программа-приемник выполняет следующие действия:

1. Установка Bi-directional режима работы порта записью константы 34h в основной регистр режима ECP порта ECR (адрес 077Ah)
2. В регистр управления стандартного порта (037Ah) записывает константу 22h – режим приема и готовность приемника.
3. Ожидание низкого уровня сигнала PerBusy - готовность передатчика к передаче
4. Ожидание низкого уровня сигнала PerClk - начало цикла передачи
5. В регистр управления стандартного порта (037Ah) записывает константу 20h – начало цикла приема байта
6. Ожидание высокого уровня сигнала PerClk - разрешение приема байта и высокого уровня сигнала PerBusy - передатчик не готов.
7. Прием байта данных через регистр данных стандартного порта (378h) и запись его в регистр AHh
8. В регистр управления стандартного порта (037Ah) записывает константу 21h – установка в низкий уровень PerClk (подтверждение приема байта данных)
9. Программной задержкой формирует длительность низкого уровня сигнала PerClk.
10. В регистр управления стандартного порта (037Ah) заносит константу 20h – установка в высокий уровень PerClk - прием закончен
11. В регистр управления стандартного порта (037Ah) записывает константу 22h – готовность приемника к приему очередного байта
12. Переход на начало нового цикла приема байта данных - пункт 4.

Примечание: Программа приема данных должна запускаться первой, и только после ее запуска на передающем компьютере может запускаться программа-передатчик.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Программа 2. (Приемник)

```
        MOV     DX, 077A    ;Установка режима Bi-directional
        MOV     AL, 34
        OUT     DX, AL
        MOV     DX, 037A    ;Готовность приемника в режиме приема
        MOV     AL, 22
        OUT     DX, AL
A:      MOV     DX, 0379    ;Ожидание низкого уровня
        IN      AL, DX      ; сигнала PerBusy - готовность передатчика
        TEST    AL, 80
        JZ      A
        MOV     CX, 000F    ;Программная задержка
B:      LOOP    B
C:      MOV     DX, 0379    ;Ожидание низкого уровня сигнала PerClk -
        IN      AL, DX      ;данные выставлены на шину данных
        TEST    AL, 40
        JNZ     C
        MOV     CX, 000F    ;Задержка
E:      LOOP    E
        MOV     DX, 037A    ;Формирование высокого уровня сигнала HostBusy -
```

```
        MOV     AL, 20      ;подтверждение начала цикла приема
        OUT     DX, AL
        MOV     DX, 0379
D:      IN      AL, DX
        TEST    AL, 40      ;Ожидание высокого уровня сигнала PerClk -
        JZ      D           ;разрешение приема данных
        TEST    AL, 80      ;и высокого уровня сигнала PerBusy - передатчик
        JNZ     D            не готов
        MOV     CX, 000F    ;Задержка
F:      LOOP    F
        MOV     DX, 0378    ;Прием данных
        IN      AL, DX
        MOV     AH, AL      ;Запись принятых данных в AH
        MOV     CX, 000F    ;Задержка
F1:     LOOP    F1
        MOV     DX, 037A    ;Подтверждение приема - низкий уровень сигнала
        MOV     AL, 21      ;HostClk
        OUT     DX, AL
        MOV     CX, 02FF    ;Задержка, формирующая длительность низкого
F2:     LOOP    F2          ;уровня сигнала HostClk (при меньшем значении
                            ;для Pentium 333 мГц передатчик на Pentium
                            ; 133мГц зацикливается на участке метки B3:)
        MOV     DX, 037A    ;Сброс сигнала HostClk
        MOV     AL, 20
        OUT     DX, AL
        MOV     CX, 000F    ;Задержка
G:      LOOP    G
        MOV     DX, 037A    ;Готовность приемника - низкий уровень HostBusy
        MOV     AL, 22
        OUT     DX, AL
        JMP     A           ;Переход на новый цикл приема (выход из
                            ;бесконечного цикла по комбинации клавиш Ctrl+Esc).
```

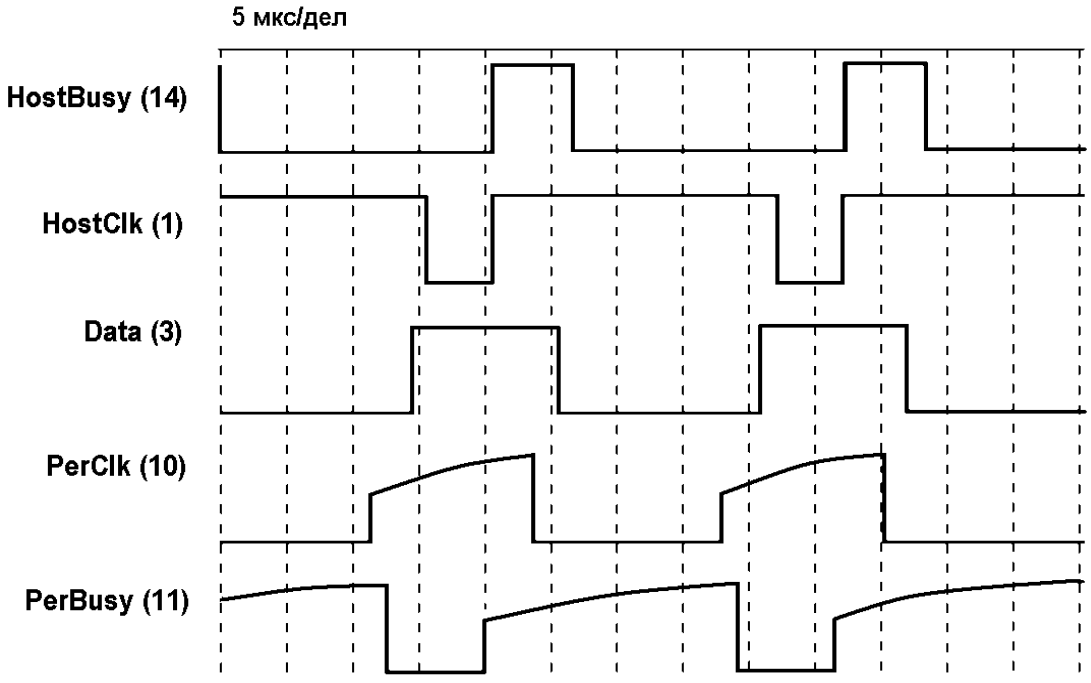


Рис.2. Осциллограммы протокольных сигналов передачи данных в режиме Bite Mode.

На рис. 2 представлены экспериментально полученные осциллограммы, иллюстрирующие процесс передачи байтов данных между двунаправленными портами компьютеров в соответствии с протоколом, временная диаграмма которого представлена на рис.1. Программа передатчик запускалась в среде отладчика AFD на компьютере с процессором Pentium 133 мГц, а программа приемник - на Pentium II 333 мГц.

6.1.3. Другие протоколы передачи данных между компьютерами в режиме Bi-directional

В режиме Bi-directional сигнальный протокол обмена данными между компьютерами реализуется программным путем с помощью регистров состояния SR (адрес 0379h) и управления CR (адрес 037Ah). В зависимости от используемых сигналов управления обменом и правил их формирования можно реализовать различные протоколы обмена. Ниже приводятся описания четырех из возможных протоколов обмена через двунаправленный порт.

6.1.3.1. Первый протокол

Первый протокол является наиболее сложным. Он предусматривает генерацию двух типов управляющих сигналов, как со стороны передатчика, так и со стороны приемника. Первый тип сигналов определяет готовность устройств к обмену. Второй тип со стороны передатчика сопровождает передаваемые данные и указывает приемнику момент считывания информации с линий данных, а со стороны приемника - квитирует, т.е. подтверждает прием данных приемником (см. рис. 3). В отличие от ранее рассмотренного протокола (см. рис. 1) в этом протоколе длительности стробирующего сигнала HostClk# и квитирующего сигнала PerClk# формируются программными задержками независимо.

Алгоритмы программ, реализующих функции передатчика и приемника приведены ниже.

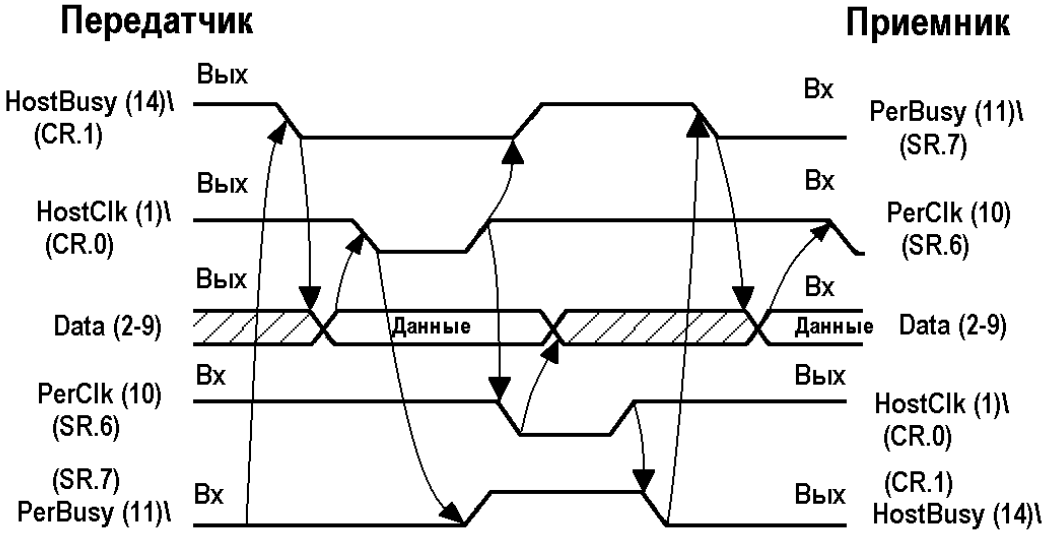


Рис. 3. Временные диаграммы первого протокола

Программа-передатчик

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.3 программа-передатчик выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. Анализирует состояние линии PerBusy (контакт 11 - состояние занятости прямого канала (приемник не готов)): если сигнал имеет высокий уровень (SR.7 = 0) - организуется цикл ожидания низкого уровня этого сигнала.
3. В регистр управления стандартного порта CR (адрес 037Ah) записывает константу

02h. Эта константа устанавливает сигнал HostBusy на контакте 14 в низкий уровень (разряд CR.1=1) – передатчик готов к передаче, сигнал HostClk на контакте 1 устанавливает в высокий уровень (разряд CR.0=0), в разряд CR.5 регистра записывается 0, настраивая порт на передачу данных

4. Передачу байта данных через регистр данных стандартного порта DR (адрес 0378h)
5. В регистр управления стандартного порта (037Ah) записывает константу 03h, подтверждая низкий уровень сигнала HostBusy (14) и устанавливает HostClk (1) в низкий уровень (индикация действительности данных на линиях DATA [7:0])
6. Программной задержкой формирует длительность сигнала HostClk (1)
7. В регистр управления стандартного порта (037Ah) записывает константу 02h, подтверждая низкий уровень сигнала HostBusy (14) и устанавливая сигнал HostClk (1) в высокий уровень
8. В регистр управления стандартного порта (037Ah) записывает константу 00h, устанавливая высокий уровень сигнала HostBusy (14)
9. Тестирует состояние линии PerClk (10) на установку низкого уровня (ноль в разряде SR.6 регистра состояния стандартного порта (адрес 0379h) - подтверждение приема байта данных)
10. Сброс в ноль шины данных (2-9) - запись константы 00h в регистр данных стандартного порта (адрес 0378h)
11. Переход на новый цикл передачи - переход на пункт 2 алгоритма

Программа-приемник

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.3 программа-приемник выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта (037Ah) записывает константу 22h – готовность приемника и режим приема (низкий уровень сигнала HostBusy (14) (CR.1=1), высокий уровень сигнала HostClk (1) (CR.0=0) и единица в разряде CR.5 - устанавливает режим приема)
3. Тестирование сигнала PerBusy (11) на низкий уровень (SR.7=1) (готовность передатчика к передаче)
4. Тестирование линии PerClk (10) на низкий уровень (SR.6=0) (данные выданы на линию)
5. В регистр управления стандартного порта (037Ah) записывает константу 20h – занятость приемника приемом данных (высокий уровень сигнала HostBusy (14) (CR.1=0))
6. Тестирование линии PerClk (10) на высокий уровень (SR.6=1)
7. Прием байта данных через регистр данных стандартного порта (378h)
8. В регистр управления стандартного порта (037Ah) записывает константу 21h – установка в низкий уровень сигнала HostClk (1) (подтверждение приема) и подтверждает высокий уровень сигнала HostBusy (14)
9. Организует программную задержку, формирующую длительность отрицательного сигнала HostClk (1)
10. В регистр управления стандартного порта (037Ah) записывает константу 20h – установка высокого уровня сигнала HostClk (1) и подтверждение высокого уровня PerBusy (14)
11. В регистр управления стандартного порта (037Ah) записывает константу 22h – установка низкого уровня HostBusy (14) (готовность приемника к приему)
12. Переход на тестирование PerBusy (готовность передатчика к передаче) (пункт 3),

организуя новый цикл приема байта данных.

Примечание: при программной реализации алгоритмов работы передатчика и приемника первой должна активизироваться программа приемника.

6.1.3.2. Второй протокол

Второй протокол предусматривает генерацию одного управляющего сигнала со стороны передатчика - сигнал сопровождения данных HostClk, и двух сигналов со стороны приемника: сигнал готовности приемника HostBusy (поступает на вход PerBusy передатчика) и квитирующий сигнал HostClk (поступает на вход PerClk передатчика) (См. рис. 4).

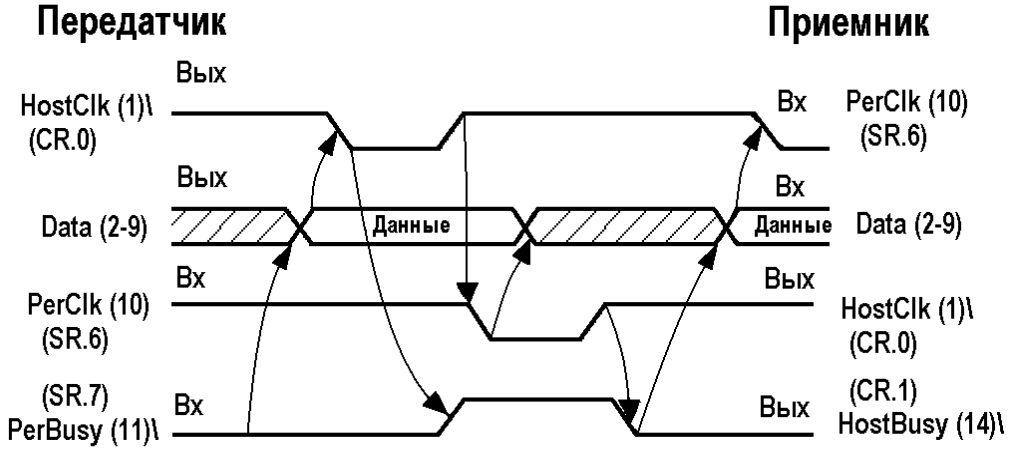


Рис. 4. Временные диаграммы второго протокола

Программа-передатчик

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.4 программа-передатчик выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта CR (адрес 037Ah) записывает константу 00h (сигнал HostClk на контакте 1 устанавливается в высокий уровень (разряд CR.0=0) , в разряд CR.5 регистра записывается 0, настраивая порт на передачу данных)
3. Тестирует сигнал PerBusy (контакт 11 - состояние занятости прямого канала (приемник не готов)) через регистр состояния стандартного порта SR (адрес 0379h, разряд SR.7). SR.7 передает инверсное состояние линии, т.е. низкий уровень представляется единицей. SR.7=1 - приемник готов к приему
4. Передает байт данных через регистр данных стандартного порта DR (адрес 0378h)
5. В регистр управления стандартного порта (037Ah) записывает константу 01h, устанавливая сигнал HostClk (1) в низкий уровень (индикация действительности данных на линиях DATA [7:0])
6. Программной задержкой формирует длительность сигнала HostClk (1)
7. В регистр управления стандартного порта (037Ah) записывает константу 00h, устанавливая высокий уровень сигнала HostClk (1)
8. Тестирует состояние линии PerClk (10) на установку низкого уровня (ноль в разряде SR.6 регистра состояния стандартного порта (адрес 0379h) - подтверждение приема)
9. Сброс данных (2-9) - запись константы 00h в регистр данных стандартного порта (адрес 0378h)
10. Переход на тестирование низкого уровня сигнала PerBusy (11) (пункт 3)

Программа-приемник

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.4 программа-приемник выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта (037Ah) записывает константу 22 h – готовность приемника и режим приема (низкий уровень сигнала HostBusy (14) (CR.1=1), высокий уровень сигнала HostClk (1) (CR.0=0) и единица в разряде CR.5 - режим приема)
3. Тестирование линии PerClk (10) на низкий уровень (SR.6=0) (данные выданы на линию)
4. В регистр управления стандартного порта (037Ah) записывает константу 20h – занятость приемника приемом данных (высокий уровень сигнала HostBusy (14) (CR.1=0))
5. Тестирование линии PerClk (10) на высокий уровень (SR.6=1)
6. Прием байта данных через регистр данных стандартного порта (378h)
7. В регистр управления стандартного порта (037Ah) записывает константу 21h – установка в низкий уровень сигнала HostClk (1) (подтверждение приема) и подтверждение высокого уровня сигнала HostBusy (14)
8. Организует программную задержку, формирующую длительность отрицательного сигнала HostClk (1)
9. В регистр управления стандартного порта (037Ah) записывает константу 20h – установка высокого уровня сигнала HostClk (1) и подтверждение высокого уровня PerBusy (14)
10. В регистр управления стандартного порта (037Ah) записывает константу 22h – установка низкого уровня HostBusy (14) (готовность приемника к приему)
11. Переход на новый цикл приема (пункт 3).

6.1.3.3. Третий протокол

Третий протокол аналогичен второму. Его отличительной особенностью является использование квитирующего сигнала PerClk (10) еще и для запроса передачи следующего байта информации (задний фронт сигнала). Этот протокол наиболее близок протоколу интерфейса Centronics стандартного параллельного порта (SPP). Временная диаграмма протокола представлена на рис. 5.

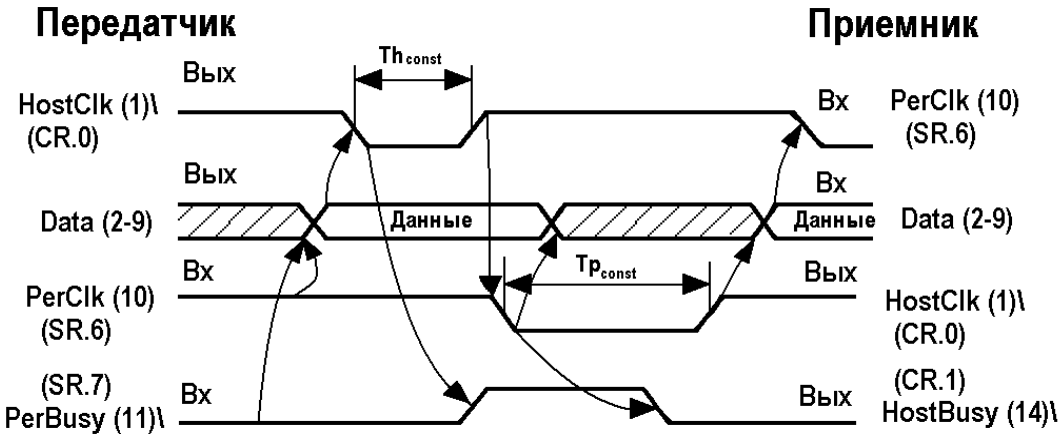


Рис. 5. Временные диаграммы третьего протокола.

Программа-передатчик

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.5 программа-передатчик должна реализовывать следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта CR (адрес 037Ah) записывает константу 00h (сигнал HostClk на контакте 1 устанавливается в высокий уровень (разряд CR.0=0) , в разряд CR.5 регистра записывается 0, настраивая порт на передачу данных)
3. Тестирует на низкий уровень сигнал PerBusy (контакт 11 - состояние занятости прямого канала (приемник не готов)) через регистр состояния стандартного порта SR (адрес 0379h, разряд SR.7). SR.7 передает инверсное состояние линии, т.е. низкий уровень представляется единицей. SR.7=1 - приемник готов к приему
4. Тестирует наличие высокого уровня сигнала PerClk (10) (переход сигнала с низкого уровня на высокий и сам высокий уровень сигнала рассматривается как запрос на передачу очередного байта)
5. Передача байта данных через регистр данных стандартного порта DR (адрес 0378h)
6. В регистр управления стандартного порта (037Ah) записывает константу 01h, устанавливая сигнал HostClk (1) в низкий уровень (индикация действительности данных на линиях DATA [7:0])
7. Программной задержкой формирует длительность сигнала HostClk (1)
8. В регистр управления стандартного порта (037Ah) записывает константу 00h, устанавливая высокий уровень сигнала HostClk (1)
9. Тестирует состояние линии PerClk (10) на установку низкого уровня (ноль в разряде SR.6 регистра состояния стандартного порта, адрес 0379h)
10. Сброс данных (2-9) - запись константы 00h в регистр данных стандартного порта (адрес 0378h)
11. Переход на новый цикл передачи (пункты 3 или 4)

Программа-приемник

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.5 программа-приемник выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта (037Ah) записывает константу 22h – готовность приемника, запрос передатчику на передачу байта и режим приема (низкий уровень сигнала HostBusy (14) (CR.1=1), высокий уровень сигнала HostClk (1) (CR.0=0) и единица в разряде CR.5)
3. Тестирование линии PerClk (10) на низкий уровень (SR.6=0) (данные выданы на линию)
4. В регистр управления стандартного порта (037Ah) записывает константу 20h – занятость приемника приемом данных (высокий уровень сигнала HostBusy (14) (CR.1=0))
5. Тестирование линии PerClk (10) на высокий уровень (SR.6=1)
6. Прием байта данных через регистр данных стандартного порта (378h)
7. В регистр управления стандартного порта (037Ah) записывает константу 21h – установка в низкий уровень сигнала HostClk (1) (подтверждение приема) и подтверждение высокого уровня сигнала HostBusy (14)
8. В регистр управления стандартного порта (037Ah) записывает константу 23h – подтверждение низкого уровня сигнала HostClk (1) и установка низкого уровня сигнала HostBusy (14) (готовность приемника)
9. Организация программной задержки, формирующей длительность отрицательного сигнала HostClk (1)

10. В регистр управления стандартного порта (037Ah) записывает константу 22h – установка высокого уровня сигнала HostClk (1) (запрос на передачу следующего байта) и подтверждение низкого уровня PerBusy (14)
11. Переход на новый цикл приема (пункт 3).

6.1.3.4. Четвертый протокол

Четвертый протокол позволяет организовывать блочный обмен между компьютерами. Такой протокол обычно применяется при обмене данными между буферизированными устройствами. Согласно этому протоколу (см. рис. 6) передача блока данных инициируется низким уровнем сигнала, поступающего на вход PerClk передатчика при низком уровне сигнала PerBusy. Передача каждого байта данных ведется в сопровождении сигнала HostClk. В качестве квитирующего сигнала используется сигнал, поступающий на вход PerBusy передатчика. После заполнения буфера данных приемник устанавливает на входах PerClk и PerBusy передатчика высокий уровень. При такой реализации протокола сигнал PerClk может используется для генерации сигнала запроса прерывания параллельного порта (IRQ 5 или IRQ 7), а программа обработки прерывания реализует передачу блока данных пока сигнал PerClk имеет низкий уровень. После того, как принимающий компьютер освободит буфер приема, может быть инициирована передача следующего блока данных установкой низкого уровня сигналов PerBusy и PerClk на разъеме передатчика.

Временная диаграмма протокола представлена на рис. 6. В этом протоколе, как и в предыдущих, длительность стробирующих и квитирующих сигналов формируется программными задержками и для различных компьютеров должна подбираться индивидуально.

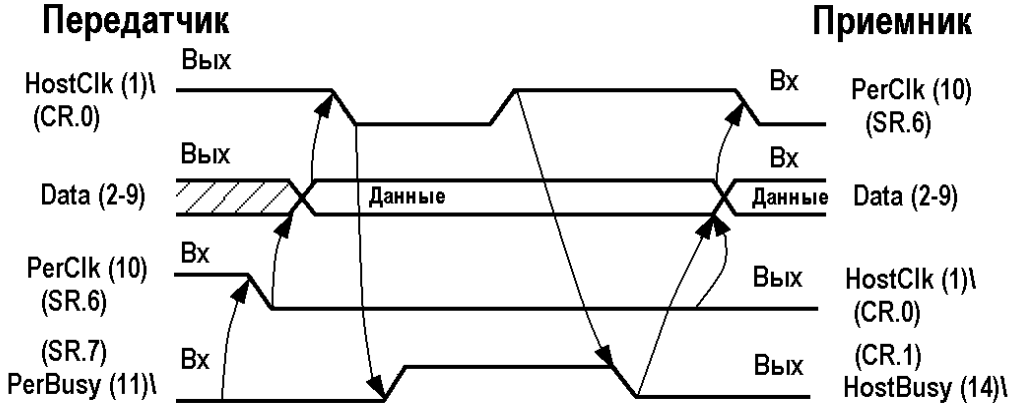


Рис. 6. Временные диаграммы четвертого протокола

Программа-передатчик

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.6 программа-передатчик выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта CR (адрес 037Ah) записывает константу 00h (сигнал HostClk на контакте 1 устанавливается в высокий уровень (разряд CR.0=0), в разряд CR.5 регистра записывается 0, настраивая порт на передачу данных)
3. Тестирует линии PerBusy (11) и PerClk (10) на наличие низкого уровня через регистр состояния стандартного порта SR (адрес 0379h, разряды SR.7 и SR.6 соответственно). SR.7 передает инверсное состояние линии PerBusy, т.е. низкий уровень представляется единицей. SR.7=1 а SR.6 - состояние линии PerClk, т.е. низкий уровень представляется нулем. Если линии PerBusy (11) и PerClk (10) имеют низкий уровень, то переходит на пункт 5, если линия PerClk (10) имеет

высокий уровень, то выполняет переход на пункт 9 - остановка по прерыванию INT 3 с передачей управления отладчику

4. Передача байта данных через регистр данных стандартного порта DR (адрес 0378h)
5. В регистр управления стандартного порта (037Ah) записывает константу 01h – устанавливает HostClk (1) в низкий уровень (индикация действительности данных на линиях DATA [7:0])
6. Программной задержкой формирует длительность сигнала HostClk (1)
7. В регистр управления стандартного порта (037Ah) записывает константу 00h – устанавливает сигнал HostClk (1) в высокий уровень
8. Переход на пункт 3
9. Передача управления отладчику командой INT 3.

Программа-приемник

В соответствии с временной диаграммой протокола передачи данных, приведенной на рис.6 программа-приемник выполняет следующие действия:

1. Устанавливает двунаправленный (Bi-directional) режим работы порта записью константы 34h в регистр ECR (главный управляющий регистр порта ECP) по адресу 077Ah
2. В регистр управления стандартного порта (037Ah) записывает константу 22h – готовность приемника и режим приема (низкий уровень сигнала HostBusy (14) (CR.1=1), высокий уровень сигнала HostClk (1) (CR.0=0) и единица в разряде CR.5 - режим приема)
3. Устанавливает счетчик числа принятых байт данных
4. Устанавливает низкий уровень сигнала HostClk (1) (запрос на прием), записывая единицу в разряд CR.0 (CR.0=1), и подтверждает низкий уровень HostBusy (14) и режим приема (в порт 037Ah записывает константу 23h)
5. Тестирует линию PerClk (10) на низкий уровень (SR.6=0) (данные выданы на линию)
6. В регистр управления стандартного порта (037Ah) записывает константу 21h – занятость приемника приемом данных (высокий уровень сигнала HostBusy (14) (CR.1=0)) и подтверждение низкого уровня сигнала HostClk (1) (CR.0=1).
7. Тестирование линии PerClk (10) на высокий уровень (SR.6=1)
8. Прием байта данных через регистр данных стандартного порта (378h) и пересылку его в буфер данных
9. Уменьшает содержимое счетчика числа принятых данных и сравнивает с нулем. Если его содержимое больше нуля, то переходит на пункт 4 - продолжение приема, в противном случае на пункт 10 - сброс запроса на прием и остановка по INT 3 с передачей управления отладчику
10. Записывает в регистр управления CR константу 20h, устанавливая высокий уровень сигналов HostBusy (14) и HostClk (1) и комендой INT 3 предает управление отладчику.

6.2. Обмен данными между ПК в режиме ECP без поддержки DMA

В протоколе ECP цикл квитирования передачи данных осуществляется аппаратными средствами контроллера ECP порта. Начало цикла передачи инициируется записью в регистр данных или команд байта информации при условии, что принимающее устройство готово к приему, т.е. сигнал PeriphAck имеет низкий уровень (см. рис. 7). Поэтому программные затраты на организацию обмена сведены к минимуму и заключаются в контроле за буфером данных, определении типа принимаемой информации и изменении направления передачи.

Для соединения компьютеров через ECP порты используются те же кабели и та же кроссовая плата, что и при обмене через двунаправленные порты Bi-directional, однако имеются различия в использовании линий интерфейса.

На рис. 7 приведены временные диаграммы прямой передачи данных между компьютерами через ECP порты. Стрелками указаны причинно-следственные связи между контроллерами ECP портов поддерживаемые на аппаратном уровне в рамках сигнального протокола квитирования. Причинно-следственная цепочка начинается после записи байта в регистр данных или команд передающего порта. Тип передаваемой через интерфейс информации определяет состояние линии HostAck: высокий уровень соответствует передаче байта данных, а низкий - байта команды.

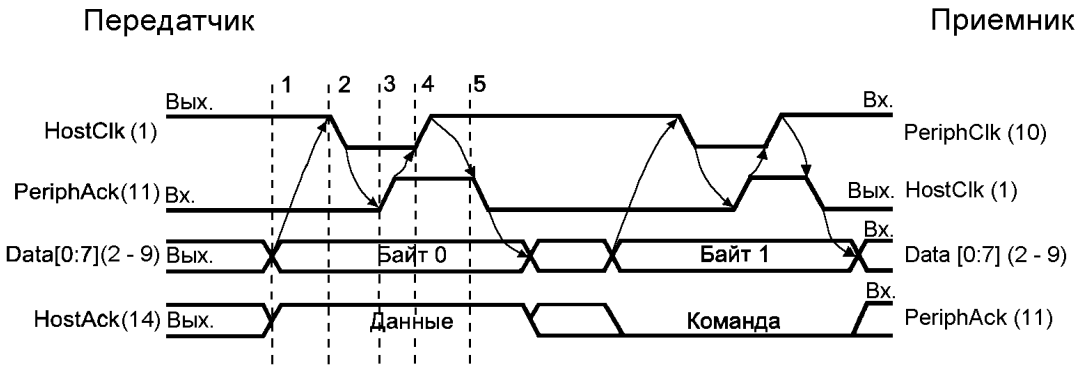


Рис. 7. Передача данных и команд в режиме ECP

На рис. 8 приведены временные диаграммы обратной передачи данных и команд. Обратная передача устанавливается после согласования переключения компьютеров на обратную передачу. Согласование производится с помощью сигналов ReversRequest# и AckRevers#. Поскольку выходы этих сигналов на участвующих в обмене компьютерах соединены перекрестно, то инициатором смены направления передачи может являться передающий или принимающий компьютер. Обратная передача должна начинаться только после того, как на обеих линиях устанавливается низкий уровень сигналов. В общем случае понятие прямой и обратной передачи весьма условно, поскольку состояние этих линий контролируется программно (или по прерыванию) и переключение направления также осуществляется программно.

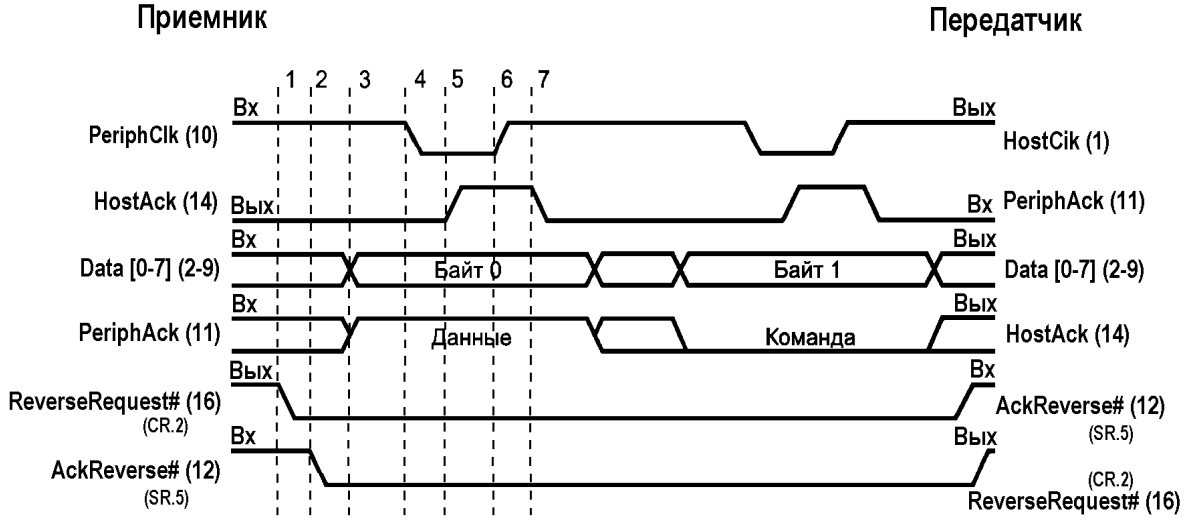


Рис.8. Прием данных и команд в режиме ECP

Для реализации обмена между компьютерами через ECP порты на них должны работать соответственно программы, управляющие передачей и приемом байтов в соответствии с требованиями протокола ECP. В приведенном ниже примере программы передачи

и приема данных реализуют передачу и прием данных (или команд, при изменении адресов соответствующих регистров) блоками по 16 байт в каждом.

Программа-передатчик

В соответствии с временной диаграммой, приведенной на рис.7 и правилами организации передачи данных и команд через ECP порт программа-передатчик выполняет следующие действия:

1. В регистр управления стандартного порта(037Ah) записывает константу 04h – режим передачи (CR.5=0), высокий уровень на линии ReversRequest# (CR.2=1) и высокий уровень на линии 1284Activ (CR.3=0).
2. Устанавливает ECP режим работы порта без поддержки DMA записью константы 74h в регистр ECR(77Ah)
3. Устанавливает величину блока данных записью константы 000Fh в регистр CX
4. В регистр аккумулятора AL записывает 00h - первую константу для передачи ее через регистр очереди данных (адрес 778h)
5. Передает байт данных в регистр очереди данных (адрес 778h)
6. Увеличивает значение AL на 1
7. Повторяет передачу данных заданное количество раз (переход на пункт 5 пока CX>0)
8. Организует программную задержку и возобновляет выполнение программы с пункта 3, организуя бесконечный цикл (в AFD выход из такого цикла – Ctrl+Esc).

В алгоритме программы не предусмотрена процедура переключения направления передачи.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Программа 3. (Передатчик)

```
        MOV     DX, 037A    ;Установка направления
        MOV     AL, 04      ;передачи данных - передача
        OUT     DX, AL
        MOV     DX, 077A    ;Установка режима ECP без
        MOV     AL, 74      ;поддержки DMA
        OUT     DX, AL
A:      MOV     CX, 000F    ;Количество передаваемых байт данных
        MOV     AL, 00      ;Задание начального значения AL
B:      MOV     DX, 0778    ;Передача данных
        OUT     DX, AL
        INC     AL          ;Увеличение значения AL на 1
        LOOP    B           ;Переход на передачу следующего байта
        MOV     CX, 0FFF    ;Задержка
C:      LOOP    C
        JMP     A           ;Переход на новый цикл передачи блока данных. При
                            ;однократном выполнении программы вместо
                            ;команды JMP A необходимо применить команду
                            ;INT3 - возврат в AFD.
```

Программа-приемник

В соответствии с временной диаграммой, приведенной на рис.7 и правилами организации передачи данных и команд через ECP порт программа-приемник выполняет следующие действия:

1. В регистр управления стандартного порта(037Ah) записывает константу 24h – режим приема (CR.5=1), высокий уровень на линии ReversRequest# (CR.2=1) и высокий уровень на линии 1284Activ (CR.3=0)

2. Устанавливает ECP режим работы порта без поддержки DMA записью константы 74h в регистр ECR(77Ah)
3. Тестирует флаг заполнения входного буфера (ECR.1=1 - ?)
4. Чтение содержимого буфера (16 байт) и запись его в память сегмента данных (DS: 0000h - DS:0010h)
5. Зацикливание программы переходом на пункт 1

В алгоритме программы не предусмотрена процедура переключения направления передачи.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Программа 4. (Приемник)

```
A0:     MOV     DI, 0000        ;Обнуление регистра DI
        MOV     AX, DS          ;Запоминание значения
        MOV     ES, AX          ;регистра DS в ES
        MOV     DX, 037A        ;Настройка на
        MOV     AL, 24          ;прием данных
        OUT     DX, AL
        MOV     DX, 077A        ;Установка режима ECP
        MOV     AL, 74          ;без поддержки DMA(ПДП)
        OUT     DX, AL
A:      MOV     DX,077A         ;Тест на заполнение буфера
        IN      AL,DX
        TEST    AL,02           ;Проверка: ECR.1=1 ?
        JZ      A
        MOV     DX,0778
B:      IN      AL, DX          ;Чтение байта данных
        MOV     ES:[DI], AL     ;и запись его по адресу ES:[DI]
        INC     DI
        TEST    DI, 0010        ;Проверка: DI=10h ?
        JZ      B
        JMP     A0              ;Зацикливание. При однократном выпол-
                                ;нении программы вместо команды JMP A0
                                ;необходимо применить команду INT3 -
                                ;возврат в AFD.
```

6.3. Обмен данными между ПК в режиме ECP с поддержкой DMA

При установке средствами Setup BIOS режима ECP или ECP+EPP параллельному порту выделяется третий канал DMA. В лабораторной работе 4 (раздел 2.6) уже рассматривались вопросы, связанные с передачей данных через ECP порт с поддержкой режима DMA. В ней рассмотрен пример передачи блока данных из ПК через ECP порт, работающий в режиме DMA на не поддерживающий режим DMA ECP порт другого ПК. В данном разделе рассмотрен обмен данными между двумя ПК через ECP порты с поддержкой DMA, т. е. как передача данных, так и их прием ведутся в режиме прямого доступа (в режиме DMA).

Перед началом работы на передающем и принимающем ПК средствами Setup BIOS следует установить ECP или ECP+EPP режимы параллельных портов и проверить номер выделенных им каналов DMA (ПДП). Как правило, система выделяет 3-й канал DMA.

Для реализации обмена между компьютерами через ECP порты на них должны работать программы, управляющие соответственно передачей или приемом байтов в режиме DMA. В приведенном ниже примере программы передачи и приема данных реализуют передачу и прием данных блоками по 16 байт в каждом.

Программа-передатчик

В соответствии с временной диаграммой, приведенной на рис.7 и правилами организации передачи данных через ECP порт в режиме DMA (см. раздел 2.6 пособия) программа-передатчик выполняет следующие действия:

1. 1.Переключает порт ECP в режим Bi-directional (запись константы 34h в регистр ECR).
2. Записью константы 04h в регистр управления стандартного порта CR (адрес 37Ah) устанавливает на выводах 16 и 17 разъема порта высокие уровни сигналов ReverseRequest# и 1284Active, а также настраивает порт на передачу (см. раздел 1.1.2 и табл. 1.7 раздела 1.3.4.1)
3. Производит инициализацию третьего канала прямого доступа к памяти DMA1:
   3.1. В регистр режима ведомого контроллера DMA1 с адресом 00Bh (см. табл. 4 раздела 9.1.4 описания) записывает конфигурационную константу 4Bh, настраивающую канал 3 DMA1 на режим одиночного вывода с инкрементацией счетчика адреса (см. формат регистра режима канала DMA в приложении описания лабораторной работы №4 - пункт 9.2.7.4).
   3.2. Преобразует 16-разрядное значение сегментного регистра DS в 20-разрядный код, указывающий на базовый адрес блока памяти, из которого в режиме DMA выводятся данные через ECP порт. Значение 20-разрядного адреса формируется в регистрах AX и DL умножением содержимого DS на 10h (16 в десятичном виде).
   3.3. Записывает два младших байта 20-разрядного базового адреса из AX в 16-разрядный счетчик адреса канала 3 контроллера DMA1. Запись производится по адресу 006h (см. табл. 4 в разделе 9.1.4. описания лабораторной работы №4) сначала младший байт из AL, а затем второй - из AH.
   3.4. Записывает третий байт базового адреса из DL в регистр страниц канала 3 подсистемы прямого доступа к памяти по адресу 82h (см. табл. 3 в разделе 9.1.3.описания лабораторной работы №4).
   3.5. Записывает количество циклов DMA (в нашем случае 0Fh) в регистр счетчика числа циклов DMA канала 3 по адресу 007h (см. табл. 4 раздела 9.1.4. описания лабораторной работы №4). Записывается сначала младший байт (0Fh), а затем - старший - (00h).
   3.6. Сбрасывает разряд маски канала 3 DMA1, записывая константу 03h (см. раздел 9.2.7.7 описания лабораторной работы №4) по адресу 00Ah (см. табл. 4 раздела 10.1.4. описания лабораторной работы №4).
1. Устанавливает режим ECP порта с разрешением DMA записью в регистр ECR по адресу 77Ah (см. табл. 1.9 раздела 1.3.4.2) константы 78h (см. табл. 1.8 описания лабораторной работы №4 и описание регистра ECR в разделе 1.3.4.2). Установка этого режима инициирует начало передачи данных в режиме DMA.
2. Организует тестирование регистра состояния контроллера DMA1 (адрес 0008h) на окончание циклов DMA по каналу 3 (разряд D3 регистра состояния равен 1 - счетчик переполнен, т.е. циклы DMA окончены).
3. Переключает ECP порт в режим запрещения DMA (константа 74h заносится в регистр ECR по адресу 77Ah)
4. Организует программную задержку на время, определяемое содержимым регистра CX и временем выполнения команды LOOP. Эта задержка позволяет отделить 16-байтовые передачи в режиме прямого доступа, выполняемые за каждый цикл выполнения программы 1 для удобства контроля процесса передачи осциллографом.
5. Зацикливает выполнение программы (переход на пункт 1) для предоставления возможности контроля состояния линий интерфейса с помощью обычного электронного осциллографа. При выполнении программы в среде AFD выход из цикла осуществляется комбинацией клавиш Ctrl+Esc. При однократном выполнении про-

граммы вместо перехода на начало программы необходимо применить команду INT 3 - возврат в AFD.

В алгоритме программы не предусмотрена процедура переключения направления передачи.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Текст программы-передатчика

```
A:      MOV     DX, 077A    ;Установка режима Bi-directional порта ECP с
        MOV     AL, 34      ;запретом служебных прерываний и DMA
        OUT     DX, AL
        MOV     DX, 037A    ;Установка порта в режим передачи
        MOV     AL, 04      ;1284Aktive - H-уровень и Init# (ReverseRequest# в
        OUT     DX, AL      режиме ECP) - H-уровень
        MOV     DX, 000B    ;Адрес регистра режима DMA1 заносится в DX
        MOV     AL, 4B      ;Установка режима работы канала 3 DMA: -
        OUT     DX, AL      ;одиночный вывод через канал 3 с инкрементом адреса
                            ;Установка счетчика адреса и регистра страниц на адрес
                            ;блока памяти из которого ;будут выдаваться в порт
                            ;данные в режиме DMA:
        MOV     AX, DS      ;Содержимое DS пересылается в AX
        MOV     BX, 10      ;Множитель 10h (16) записывается в BX
        MOV     DX, 0000    ;Обнуляем регистр DX
        MUL     BX          ;Умножение содержимого AX на 16 (сдвиг влево на 4
                            разряда с переносом в DX четырех старших разрядов
        OUT     [06],AL     Младший байт адреса в счетчик адреса канала 3 DMA1
        MOV     AL, AH      ;Запись второго байта адреса в счетчик адреса
        OUT     [06], AL    ;канала 3 DMA1
        MOV     AL, DL      ;Третий байт адреса из DL пересылается а AL
        OUT     [82], AL    ;и записывается в регистр страниц канала 3 DMA

        MOV     DX, 0007    ;Запись количества циклов (слов) DMA(ПДП)
        MOV     AL, 0F      ;в счетчик числа циклов канала 3 DMA1 (16 циклов)
        OUT     DX, AL      ;младший байт
        MOV     AL, 00
        OUT     DX, AL      ;старший байт
        MOV     DX, 000A    ;Сброс маски канала 3 DMA(ПДП) записью константы
        MOV     AL, 03      ;03 по адресу 00Ah контроллера DMA1
        OUT     DX, AL
        MOV     DX, 077A    ;Установка режима ECP с разрешением
        MOV     AL, 78      ;обмена по каналу DMA (ПДП) (инициируется обмен
        OUT     DX, AL      ;через порт в режиме DMA)
        MOV     DX, 0008    ;Тест на заполнение счетчика канала 3 DMA(ПДП)
C:      IN      AL, DX      ;чтением регистра состояния каналов DMA1
        TEST    AL, 08      ;и проверкой разряда конца циклов канала 3 DMA1
        JZ      C
        MOV     DX, 077A    ;Режим ECP с запрещением
        MOV     AL, 74      ;обмена по каналу DMA(ПДП)
        OUT     DX, AL
        MOV     CX, FFFF    ;Формирование величины программной задержки
B:      LOOP    B           ;Программная задержка

        JMP     A           ;Зацикливание программы; выход из цикла - Ctrl+Esc
```

```
;При однократном выполнении программы вместо
;команды JMP A необходимо применить команду INT 3
;- возврат в AFD
```

Программа-приемник

В соответствии с временной диаграммой, приведенной на рис.7, правилами организации обмена через ECP порт в режиме DMA (см. раздел 1.3.4) и правилами настройки каналов DMA (см. раздел 9.1 описания лабораторной работы №4) программа-приемник выполняет следующие действия:

1. Переключает порт ECP в режим Bi-directional (запись константы 34h в регистр ECR).
2. Записью константы 24h в регистр управления стандартного порта CR (адрес 37Ah) устанавливает на выводах 16 и 17 разъема порта высокие уровни сигналов ReverseRequest# и 1284Active, а также настраивает порт на передачу (см. раздел 1.1.2 и табл. 1.7 раздела 1.3.4.1)
3. Производит инициализацию третьего канала прямого доступа к памяти DMA1:
   3.1. В регистр режима ведомого контроллера DMA1 с адресом 00Bh (см. табл. 4 раздела 9.1.4 описания лабораторной работы №4) записывает конфигурационную константу 47h, настраивающую 3-й канал DMA1 на режим одиночного ввода с инкрементацией счетчика адреса (см. формат регистра режима в приложении описания лабораторной работы №4 - пункт 9.2.7.4).
   3.2. Преобразует 16-разрядное значение сегментного регистра DS в 20-разрядный код, указывающий на базовый адрес блока памяти, в который в режиме DMA вводятся данные через ECP порт. Значение 20-разрядного адреса формируется в регистрах AX и DL умножением содержимого DS на 10h (16 в десятичном виде).
   3.3. Записывает два младших байта 20-разрядного базового адреса из AX в 16-разрядный счетчик адреса канала 3 контроллера DMA1. Запись производится по адресу 006h (см. табл. 4 в разделе 9.1.4. описания лабораторной работы №4) сначала младший байт из AL, а затем второй - из AH.
   3.4. Записывает третий байт базового адреса из DL в регистр страниц канала 3 подсистемы прямого доступа к памяти по адресу 82h (см. табл. 3 в разделе 9.1.3.описания лабораторной работы №4).
   3.5. Записывает количество циклов DMA (в нашем случае 0Fh) в регистр счетчика числа циклов DMA канала 3 по адресу 007h (см. табл. 4 раздела 9.1.4. описания лабораторной работы №4). Записывается сначала младший байт (0Fh), а затем - старший - (00h).
4. Устанавливает режим ECP порта с разрешением DMA записью в регистр ECR по адресу 77Ah (см. табл. 1.9 раздела 1.3.4.2) константы 47h (см. табл. 1.8 описания лабораторной работы №4 и описание регистра ECR в разделе 1.3.4.2). Установка этого режима инициирует начало приема данных из линии связи во входной буфер.
5. Тестирует признак заполнения входного буфера (ECR.1=1 ?).
6. Сбрасывает разряд маски канала 3 DMA1, записывая константу 03h (см. раздел 9.2.7.7 описания лабораторной работы №4) по адресу 00Ah (см. табл. 4 раздела 10.1.4. описания лабораторной работы №4). Это инициирует процесс переписи в режиме DMA содержимого входного буфера в оперативную память.
7. Организует тестирование регистра состояния контроллера DMA1 (адрес 0008h) на окончание циклов DMA по каналу 3 (разряд D3 регистра состояния равен 1 - счетчик переполнен, т.е. циклы DMA окончены).
8. Переключает ECP порт в режим запрещения DMA (константа 74h заносится в регистр ECR по адресу 77Ah)

9. Зацикливает выполнение программы (переход на пункт 1) для предоставления возможности контроля состояния линий интерфейса с помощью обычного электронного осциллографа. При выполнении программы в среде AFD выход из цикла осуществляется комбинацией клавиш Ctrl+Esc. При однократном выполнении программы вместо перехода на начало программы необходимо применить команду INT 3 - возврат в AFD.

В алгоритме программы не предусмотрена процедура переключения направления передачи.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Текст программы-приемника

```
A:      MOV     DX, 077A    ;Установка режима Bi-directional порта ECP с
        MOV     AL, 34      ;запретом служебных прерываний и DMA
        OUT     DX, AL
        MOV     DX, 037A    ;Установка порта в режим приема
        MOV     AL, 24      ;1284Aktive - H-уровень и Init# (ReverseRequest# в
        OUT     DX, AL      режиме ECP) - H-уровень
        MOV     DX, 000B    ;Адрес регистра режима DMA1 заносится в DX
        MOV     AL, 47      ;Установка режима работы канала 3 DMA: -
        OUT     DX, AL      ;одиночный ввод через канал 3 с инкрементом адреса
                            ;Установка счетчика адреса и регистра страниц на адрес
                            блока памяти из которого ;будут выдаваться в порт
                            данные в режиме DMA:
        MOV     AX, DS      ;Содержимое DS пересылается в AX
        MOV     BX, 10      ;Множитель 10h (16) записывается в BX
        MOV     DX, 0000    ;Обнуляем регистр DX
        MUL     BX          ;Умножение содержимого AX на 16 (сдвиг влево на 4
                            разряда с переносом в DX четырех старших разрядов
        OUT     [06],AL     Младший байт адреса в счетчик адреса канала 3 DMA1
        MOV     AL, AH      ;Запись второго байта адреса в счетчик адреса
        OUT     [06], AL    ;канала 3 DMA1
        MOV     AL, DL      ;Третий байт адреса из DL пересылается в AL
        OUT     [82], AL    ;и записывается в регистр страниц канала 3 DMA

        MOV     DX, 0007    ;Запись количества циклов DMA(ПДП)
        MOV     AL, 0F      ;в счетчик числа циклов канала 3 DMA1 (16 циклов)
        OUT     DX, AL      ;младший байт
        MOV     AL, 00
        OUT     DX, AL      ;старший байт
        MOV     DX, 077A    ;Установка режима ECP с разрешением
        MOV     AL, 78      ;обмена по каналу DMA (ПДП)
        OUT     DX, AL
A1:     MOV     DX, 077A    ;Тест на заполнение буфера (проверка первого
        IN      AL, DX      ;разряда регистра ECR)
        TEST    AL, 02
        JZ      A1
        MOV     DX, 000A    ;Сброс маски канала 3 DMA(ПДП) записью константы
        MOV     AL, 03      ;03 по адресу 00Ah контроллера DMA1 (инициализация
        OUT     DX, AL      ;обмена между буфером порта и ОП в режиме DMA)
        MOV     DX, 0008    ;Тест на заполнение счетчика канала 3 DMA(ПДП)
C:      IN      AL, DX      ;чтением регистра состояния каналов DMA1
        TEST    AL, 08      ;и проверкой разряда конца циклов канала 3 DMA1
```

```
JZ      C
MOV     DX, 077A    ;Установка режима ECP с запрещением
MOV     AL, 74      ;обмена по каналу DMA(ПДП)
OUT     DX, AL

JMP     A           ;Зацикливание программы; выход из цикла - Ctrl+Esc
                    ;При однократном выполнении программы вместо
                    ;команды JMP A необходимо применить команду INT 3
                    ;- возврат в AFD
```

6.4. Обмен данными между ПК в режиме EPP

Обмен информацией между компьютерами через параллельный порт, работающий в режиме EPP, значительно отличается от подобных операций, выполняемых через Bi-directional или ECP порты. Это связано с тем, что в режиме EPP выполняются так называемые вложенные циклы чтения/записи с ведомым устройством, поддерживающим протокол EPP с несимметричным набором синхронизирующих и квитирующих сигналов, генерируемых контроллерами портов на аппаратном уровне (см. раздел 1.3.3). Эта несимметричность аппаратно реализуемого квитирующего протокола не позволяет осуществлять непосредственное перекрестное соединение линий интерфейса между EPP портами компьютеров. Поэтому в этом разделе рассмотрена организация обмена между компьютерами, один из которых использует параллельный порт в режиме EPP, а второй - в режиме Bi-directional. В этом случае второй компьютер программно эмулирует ведомое устройство, поддерживающее протокол EPP и обеспечивающее прием и выдачу данных и адресов по требованию ведущего компьютера. Компьютеры соединяются между собой через кабели и кроссовую плату, используемые при организации обмена в режимах ECP и Bi-directional, описание которых приведено в разделе 2.1.2.

Для реализации обмена между компьютерами через EPP порт с одной стороны и Bi-directional порт с другой, на них должны работать программы, управляющие передачей или приемом байтов соответственно в режиме EPP и Bi-directional (Bi-Di). В приведенном ниже примере программ передачи и приема реализуется передача и прием байта данных и адреса и помещение их в регистры AH и BH соответственно. В качестве адреса передается константа AAh, а в качестве данных - 55h.

Перед началом работ необходимо убедиться в том, что средствами BIOS Setup установлен ECP+EPP режим работы контроллера параллельного порта. Эти режимы поддерживают программное переключение режима работы контроллера порта в любой из режимов, предусмотренных стандартом IEEE 1284. В этом режиме BIOS на стадии POST устанавливает на выводах 1, 14 и 16 разъема порта высокий уровень, а на выводе 17 - низкий (содержимое регистра CR = CCh.

Описание алгоритмов программ ведется на основании временных диаграмм протокола передачи и приема данных и адреса в режиме EPP с одной стороны и в режиме Bi-Di с другой. На диаграмме указаны контакты разъемов портов, направление сигналов, соответствующие разряды регистров состояния (SR) и управления (CR), через которые доступны эти сигналы в режиме Bi-Di. Временная диаграмма передачи данных в режиме EPP представлена на рис.9 (передача адреса реализуется аналогично, только вместо строба данных DataStrobe# портом EPP генерируется сигнал AddrStrobe#), а приема адреса - на рис. 10. (прием данных реализуется аналогично, только вместо строба адреса AddrStrobe# портом EPP генерируется сигнал DataStrobe#)

На временных диаграммах цифры в окружностях обозначают номера контактов разъемов параллельных портов; в скобках указаны разряды регистров, через которые доступны соответствующие сигналы в режиме Bi-Di; косая черта (\) в обозначение разряда указывает инверсию, т.е. высокому уровню сигнала(+5.В) соответствует значение "0", низкому уров-

ню (0.В) соответствует значение "1"; стрелка над обозначением сигнала указывает на направление прохождения этого сигнала через контакты разъемов портов.

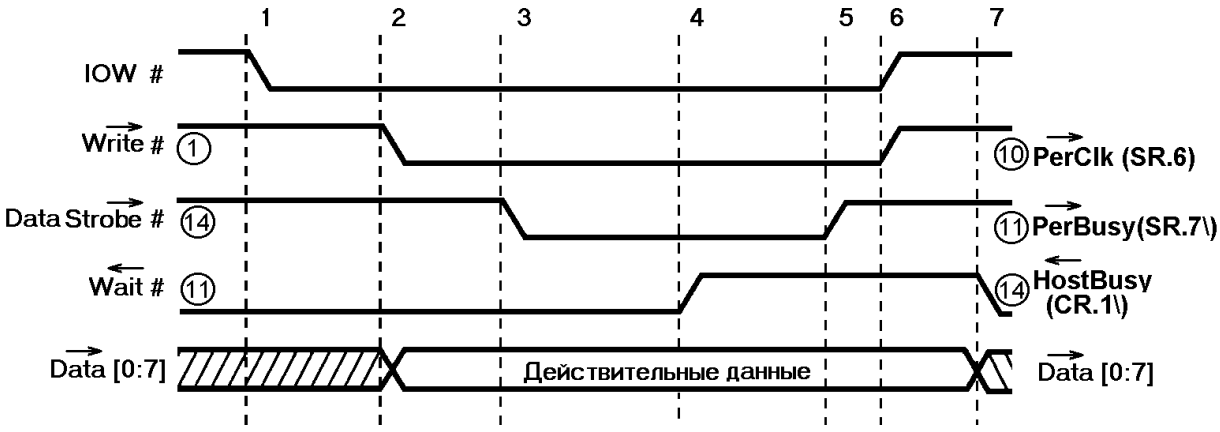


Рис. 9. Временная диаграмма передачи данных портом EPP и приема этих данных портом Bi-Di.

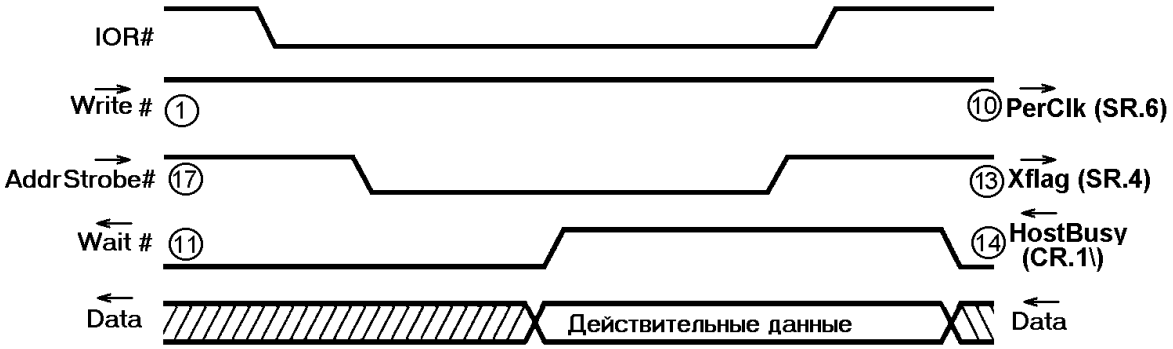


Рис. 10. Временная диаграмма приема адреса портом EPP и передачи этого адреса портом Bi-Di.

1. Устанавливает режим Bi-directional занесением константы 34h в регистр ECR (адрес 77Ah).
2. Устанавливает EPP режим записью в регистр ECR константы 94h (см. табл.1.8 и описание формата регистра ECR в разделе 1.3.4 или 1.9.2.1).
3. Через регистр CR (37Ah), обнулением разрядов CR.0 - CR.5, устанавливает низкий уровень сигнала на выводе 16 (CR.2 Reset), указывая ведомому компьютеру на готовность ведущего к обмену, высокий уровень на выводах 1, 14 и 17 (Write#, DataSrobe# и AddrStrobe#), разрешая аппаратно управлять этими линиями контроллеру EPP-порта в циклах обмена, CR.4=0 запрещает прерывания. Разряд CR.5 не влияет на работу EPP-порта.
4. Задает число циклов выполнения программы (в регистре CX)
5. Сохраняет содержимое CX в стеке
6. Через регистр данных EPP порта выдает константу 55h (адрес 37Ch).
7. Организует программную задержку на несколько микросекунд.
8. Через регистр адреса EPP порта (адрес 37Bh) выдает на линии D7 - D0 интерфейса константу AAh.
9. Организует программную задержку на несколько микросекунд.
10. Через регистр данных EPP порта (адрес 3FCh) читает байт данных из компьютера, эмулирующего устройство, поддерживающее протокол EPP и записывает его в регистр AH.
11. Организует программную задержку на несколько микросекунд.

12. Через регистр адреса EPP порта (адрес 37Bh) принимает байт адреса из компьютера-эмулятора и помещает его в регистр BH.
13. Организует программную задержку на несколько микросекунд.
14. Извлекает из стека значение CX и организует условную передачу управления на новый цикл обмена (пункт 4): циклы будут повторяться до обнуления содержимого регистра CX (команда LOOP).
15. Записью в CR.2 единицы сообщает приемнику об окончании работы.
16. Останавливает работу программы командой INT 3 (передача управления отладчику).

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Текст программы, работающей с EPP портом **(eppbda.bin)**

```
        MOV     DX,077A   ;Установка Bi-directional режима через ECR -
        MOV     AL,34     ;главный управляющий регистр ECP
        OUT     DX,AL
        MOV     DX,077A   ;Установка EPP режима записью в регистр ECR
        MOV     AL,94     ;константы 94
        OUT     DX,AL
        MOV     DX,037A   ;Обнуление разрядов CR.0 - CR.5, которые
        IN      AL,DX     ;устанавливают пассивный высокий уровень
        AND     AL,C0     ;на контактах 1, 14, 17 (Write#, DataSrobe#,
        OUT     DX,AL     ;AddrStrobe#) и низкий уровень
                          ;на контакте 16 (готов).
        MOV     CX,0100   ;В CX - число циклов выполнения программы

        PUSH    CX        ;Содержимое CX заносится в стек
A0:     MOV     DX,037C   ;Через регистр данных EPP выдает на линии
        MOV     AL,55     ;AD [0:7] (контакты 2-9) константу 55h
        OUT     DX,AL
        MOV     CX,02FF   ;Программная задержка
B:      LOOP    B
        MOV     AL,AA
        MOV     DX,037B   ;Через регистр адреса EPP выдаем на линии
        OUT     DX,AL     ;AD [0:7] (контакты 2-9) константу AAh
        MOV     CX,02FF   ;Программная задержка
C:      LOOP    C
        MOV     DX,037C   ;Через регистр данных EPP читается
        IN      AL, DX    ;байт данных и записывается
        MOV     AH,AL     ;в регистр AH
        MOV     CX,02FF   ;Программная задержка
B1:     LOOP    B1
        MOV     DX,037B   ;Через регистр адреса EPP читается с линии
        IN      AL,DX     ;AD [0:7] (контакты 2-9) байт адреса
        MOV     BH, AL    ;и записывается в регистр BH
        POP     CX        ;Зацикливание программы пока содержимое CX
        LOOP    A5        ;не обнулится

        MOV     DX,37A    ;Восстановление
        IN      AL,DX     ;состояния разрядов
        OR      AL,04     ;CR.2=1 (не готов).
        OUT     DX,AL
```

```
        INT3                ;Передача управления отладчику

A5      PUCH    CX
        MOV     CX,02FF     ;Программная задержка
C1:     LOOP    C1
        JMP     A0          ;На новый цикл программы.
```

Программа, работающая с Bi-directional портом

В соответствии с временными диаграммами, иллюстрирующими процедуры передачи и приема данных и адресов через EPP порт, приведенных на рис.9 и 10 описания и в соответствии с правилами использования регистров SPP, EPP и Bi-directional портов программа управления обменом выполняет следующие действия:

1. Устанавливает режим Bi-directional занесением константы 34h в регистр ECR (адрес 77Ah) и через регистр CR (37Ah) устанавливает константой 02h признак готовности компьютера к обмену (формирует низкий уровень сигнала Wait на входе 11 EPP порта через выход Host Busy (14) своего Bi-Di порта).
2. Ожидание готовности ведущего компьютера к обмену (SR.5=0 ?) - сигнал Reset# (ReverseRequest#) активен (низкий уровень). Доступен через вход AckDataReq (контакт 12) (см. табл. 1.4, раздел 1.3.2 и табл. 1.13 раздела 1.6).
3. Определяет состояние линий стробов адреса и данных через разряды SR.4 и SR.7\ регистра состояния SPP-порта:
   - если SR.4=0, то переход на обмен адресами (пункт 4);
   - если SR.7\=1, то переход на обмен данными (пункт 5);
   - в противном случае на новый цикл определения состояния линий.
4. Обмен адресами.

4.1. Определение состояния линии Write# через SR.6:
   - SR.6 = 1 - передача адреса через Bi-Di порт в EPP порт ведущего компьютера.
   - SR.6 = 0 - прием адреса от EPP порта ведущего компьютера.

4.2. Прием адреса

4.2.1. Переключение порта на прием байта установкой разряда CR.5 в состояние "1".

4.2.1. Формирование высокого уровня сигнала Wait# через разряд CR.1\ = 0.

4.2.2. Ожидание высокого уровня сигнала AddrStrobe# контролем состояния разряда SR.4: если SR.4 = 0, то новый цикл контроля.

4.2.3. Чтение адреса через регистр данных Bi-Di порта и запоминание его в регистре BH.

4.2.4. Установка низкого уровня сигнала Wait# через разряд CR.1\ = 1.

4.2.5. Программная задержка несколько микросекунд.

4.2.6. Проверка готовности ведущего компьютера через разряд SR/5:
   - если SR.5 = 1, то на завершение программы (пункт 6);
   - если SR.5 = 0, то на новый цикл (пункт 3).

4.3. Передача адреса.

4.3.1. Переключение порта на передачу байта установкой разряда CR.5 в состояние "0".

4.3.2. Передача адреса через регистр данных Bi-Di порта.

4.3.3. Формирование высокого уровня сигнала Wait# через разряд CR.1\ = 0.

4.3.4. Ожидание высокого уровня сигнала AddrStrobe# контролем состояния разряда SR.4: если SR.4 = 0, то новый цикл контроля.

4.3.5. Обнуление адреса

4.3.6. Переход на пункт 4.2.4.

5. Обмен данными.

5.1. Определение состояния линии Write# через SR.6:

- SR.6 = 1 - передача данных через Bi-Di порт в EPP порт ведущего компьютера (5.3).
- SR.6 = 0 - прием данных от EPP порта ведущего компьютера (5.2).

5.2. Прием данных
5.2.1. Переключение порта на прием байта установкой разряда CR.5 в состояние "1".
5.2.2. Формирование высокого уровня сигнала Wait# через разряд CR.1\ = 0.
5.2.3. Ожидание высокого уровня сигнала DataStrobe# контролем состояния разряда SR.7\: если SR.7\ = 1, то новый цикл контроля.
5.2.4. Чтение данных через регистр данных Bi-Di порта и запоминание его в регистре AH.
5.2.5. Переход на пункт 4.2.4.
5.3. Передача данных.
5.3.1. Переключение порта на передачу байта установкой разряда CR.5 в состояние "0".
5.3.2. Передача данных через регистр данных Bi-Di порта.
5.3.3. Формирование высокого уровня сигнала Wait# через разряд CR.1\ = 0.
5.3.3. Ожидание высокого уровня сигнала DataStrobe# контролем состояния разряда SR.7\: если SR.7\ = 1, то новый цикл контроля.
5.3.4. Обнуление данных.
5.3.5. Переход на пункт 4.2.4.
6. Окончание программы.

Пример реализации выше приведенного алгоритма в мнемокоде команд процессора i8086, приведен ниже:

Текст программы, работающей с портом Bi-directional **(eppbdp.bin)**

```
A0:     MOV     DX,077A     ;Установка Bi-directional режима через ECR -
        MOV     AL,34       ;главный управляющий регистр ECP
        OUT     DX,AL
        MOV     DX, 037A    ;Формирование признака готовности к обмену -
        MOV     AL, 02      ;установкой низкого уровня сигнала Wait#
        OUT     DX, AL
AH:     MOV     DX, 0379    ;Ожидание готовности ведущего компьютера
        IN      AL, DX
        TEST    AL, 20
        JNZ     AH
START:  MOV     DX,0379     ;Определение строба на линиях DataStrobe#
        IN      AL, DX      ; и AddrStrobe#  тестированием
        TEST    AL, 10      ;разрядов SR.4 и SR.7.
        JZ      A           ;Переход на обмен адресами (SR.4=0).
        TEST    AL,80
        JNZ     D           ;Переход на обмен данными (SR.7=1).
        JMP     START       ;На новый цикл опроса.
                            ;Обмен адресами.
A:      MOV     DX, 0379    ;Определение состояния линии Write#
        IN      AL, DX      ;через SR.6
        TEST    AL,40
        JNZ     TRADR
                            ;Прием адреса
RSADR:  MOV     DX,037A     ;Установка в единицу разряда CR.5 (прием)
        IN      AL, DX      ;и CR.1=0 (высокий уровень Write#)
        OR      AL,20       ;регистра
        AND     AL,FD       ;управления
```

```
        OUT     DX,AL   ;SPP-порта (037Ah).
        MOV     DX,0379 ;Ожидание высокого
A1:     IN      AL,DX   ;уровня сигнала
        TEST    AL,10   ;AddrStrobe#:
        JZ      A1      ;при SR.4=0 на новый цикл ожидания.
        MOV     DX,0378 ;Чтение адреса
        IN      AL,DX   ;и запоминание его
        MOV     BH,AL   ;в регистре BH.
A4:     MOV     DX,037A
        IN      AL,DX   ;Формирование низкого
        OR      AL,02   ;уровня сигнала Wait#
        OUT     DX,AL   ;установкой CR.1=1.
        MOV     CX, 007F ;Программная
A2:     LOOP    A2      ;задержка.
        MOV     DX, 0379 ;Проверка готовности
        IN      AL,DX   ;ведущего компьютера
        TEST    AL, 20  ;тестированием разряда SR.5:
        JZ      START   ;если SR.5=0, то новый цикл обмена,
        INT3            ;иначе - окончание программы
                        ;Передача адреса.
TRADR:  MOV     DX,037A ;Переключение порта
        IN      AL,DX   ;в режим передачи
        AND     AL,DF   ;установкой
        OUT     DX, AL  ;разряда CR.5=0.
        MOV     DX,0378 ;Передача константы AAh как адрес
        MOV     AL,AA   ;через регистр
        OUT     DX,AL   ;данных порта Bi-Di.
        MOV     DX,037A ;Формирование высокого
        IN      AL,DX   ;уровня сигнала Wait#
        AND     AL,FD   ;установкой
        OUT     DX,AL   ;разряда CR.1=0.
        MOV     DX,0379 ;Ожидание высокого уровня
A3:     IN      AL,DX   ;сигнала AddrStrobe# тестированием SR.4=0?
        TEST    AL,10   ;если SR.4=0,
        JZ      A3      ; то новый цикл тестирования.
        MOV     DX,0378 ;Обнуление
        MOV     AL,00   ;адресной
        OUT     DX,AL   ;информации.
        JMP     A4      ;Переход на окончание цикла обмена.
                        Обмен данными
D:      MOV     DX, 0379 ;Определение состояния линии Write#
        IN      AL, DX  ;через SR.6
        TEST    AL,40
        JNZ     TRDAT
                        Прием данных
RSDAT:  MOV     DX,037A ;Установка в единицу разряда CR.5 (прием)
        IN      AL, DX  ;и CR.1=0 (высокий уровень Write#)
        OR      AL,20   ;регистра
        AND     AL,FD   ;управления
        OUT     DX,AL   ;SPP-порта (037Ah).
        MOV     DX,0379 ;Ожидание высокого
D1:     IN      AL,DX   ;уровня сигнала
```

```
        TEST    AL,80       ;DataStrobe#:
        JNZ     D1          ;при SR.7\=1 на новый цикл ожидания.
        MOV     DX,0378     ;Чтение байта данных
        IN      AL,DX       ;и запоминание его
        MOV     AH,AL       ;в регистре AH.
        JMP     A4          ;Переход на окончание цикла обмена.
                            Передача данных
TRDAT:  MOV     DX,037A     ;Переключение порта
        IN      AL,DX       ;в режим передачи
        AND     AL,DF       ;установкой
        OUT     DX, AL      ;разряда CR.5=0.
        MOV     DX,0378     ;Передача константы 55h как данных
        MOV     AL,55       ;через регистр
        OUT     DX,AL       ;данных порта Bi-Di.
        MOV     DX,037A     ;Формирование высокого
        IN      AL,DX       ;уровня сигнала Wait#
        AND     AL,FD       ;установкой
        OUT     DX,AL       ;разряда CR.1=0.
        MOV     DX,0379     ;Ожидание высокого уровня
D3:     IN      AL,DX       ;сигнала DataStrobe# тестированием SR.7\=0?
        TEST    AL,10       ;если SR.7\=1,
        JZ      D3          ; то новый цикл тестирования.
        MOV     DX,0378     ;Обнуление
        MOV     AL,00       ;шины данных
        OUT     DX,AL       ;интерфейса.
        JMP     A4          ;Переход на окончание цикла обмена.
```

Примечание:

При неработоспособности ведомого компьютера команды обмена с регистрами адреса и данных контроллера EPP-порта оканчиваются по таймауту за 10 мкс. При работе программы, эмулирующей EPP устройство через порт Bi-Di на компьютере с тактовой частотой 500 МГц, а программы, управляющей EPP портом - на компьютере с частотой 166 МГц наблюдалось неустойчивое выполнение команд приема адреса и данных EPP портом при программных задержках между командами меньше 5 мкс (CX меньше 2FFh).

7. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

При подготовке к лабораторным работам и к их защите рекомендуется пользоваться информацией, представленной в основном разделе учебно-методического пособия и в дополнительной литературе [1, 3, 6, 10].

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы правила соединения параллельных портов компьютеров при обмене байтовой информацией меюжу ними в режимах Bi-Di и ECP?
2. Каковы особенности обмена между ПК через EPP порты?
3. Каковы особенности обмена в режиме ECP с использованием DMA?
4. В каком режиме работы параллельных портов можно программно реализовывать различные протоколы обмена сигнального уровня (физического уровня)?
5. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол, временная диаграмма которого представлена на рис. 1 (пункты 6.1.1. и 6.1.2).
6. Объясните временные диаграммы, представленные на рис. 2.
7. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол, временная диаграмма которого представлена на рис. 3 (пункты 6.1.3.1).

8. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол, временная диаграмма которого представлена на рис. 4 (пункты 6.1.3.2).
9. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол, временная диаграмма которого представлена на рис. 5 (пункты 6.1.3.3).
10. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол, временная диаграмма которого представлена на рис. 6 (пункты 6.1.3.4).
11. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол обмена между компьютерами через параллельные порты, работающие в режиме ECP без поддержки DMA (см. рис.7 и рис. 8, пункт 6.2).
12. Охарактеризуйте программу передатчик и программу приемник, реализующие протокол обмена между компьютерами через параллельные порты, работающие в режиме ECP с поддержкой DMA (пункт 6.3).
13. Охарактеризуйте программы, реализующие обмен между компьютером с EPP портом и компьютером с портом, работающим в режиме Bi-Di (пункт 6.4).

3. ПРИЛОЖЕНИЕ

3.1. Основные команды отладчика AFD (Advanced Fullscreen Debugger)

L имя_файла {,адрес1} - загрузить файл в память. Адрес загрузки можно задавать параметром (адрес1). По умолчанию - адрес1 = CS:0100. Количество отображаемых байтов помещается в BX, CX после окончания команды. Параметр имя_файла - это имя файла с обязательным указанием расширения (расширение по умолчанию .exe). Если у фила нет расширения, то параметр имя_файла должен оканчиваться (.).
Пример: L aaa.txt, CS:0140

W имя_файла, адрес, количество_байт_- записать данные в файл. Адресный сегмент по умолчанию - DS, параметр количество_байт - определяет количество записываемых в файл байт (4-х разрядное шестнадцатеричное число).
Пример: W aaa.txt ,0100,0136

{R} регистр = содержимое регистра - установить регистр. При выполнении команды FL=содержимое регистра, флаговый регистр установится как 16 -разрядный регистр. Для раздельного доступа к флаговым битам нужно использовать их идентификаторы OF, DF, IF, SF, ZF, AF, PF, CF.

D адрес - показать код. Указатель устанавливается в окне дизассемблера на этот адрес. Сегмент по умолчанию CS.

M n [регистр:] адрес - показать окно памяти (**n = 1** или **n = 2**). Сегмент по умолчанию тот, который показан в соответствующем окне. FS можно использовать для фиксации памяти. Для косвенной адресации, содержимое любого регистра может использоваться, как адрес.
Пример: M2 100
M1 CS:100

G {нач_адрес} {,кон_адрес} - выполнить программу с текущего адреса или со стартового адреса нач_адрес; кон_адрес - точка останова. Сегмент по умолчанию - текущий CS.
Пример: G 2aaa,2baa

A {адрес1} - переход в ассемблерный режим, в котором происходит редактирование программ. Инструкции вводятся по команде "Enter", клавиши управления курсором используются для перемещения между командами в области кодов вверх и вниз. Параметр адрес1 - адрес входа (по умолчанию - текущий адрес).

P адрес, строка - «взлом » памяти. Сегмент по умолчанию CS.
Пример: P 100, 12EF 'STR' - помещает коды 12h, Efh и коды КОИ-8 символов S, T и R в память, начиная с ячейки CS:0100.

F адрес, число повторений, строка - заполнить память указанной строкой. Адресный сегмент по умолчанию DS. Число повторений определяет, сколько раз строка будет записана в память.

S {адрес1} {,строка1} - поиск данных в памяти, начиная с адреса {адрес1}. Если адрес не указан, поиск начинается с CS:00. Когда данные обнаружены, экран M2 отобразит эту область памяти. Команда S без параметров начинает повторный поиск. Строка1 - строка символов, любой набор значений, 1234, CD, 'ASCII', и т.д.

C адрес1, адрес2, длина - сравнение двух областей памяти. Если обнаружено несовпадение, то на экране M1 будет отражена область памяти, адресуемая первым параметром, а на экране M2-область памяти, адресуемая вторым параметром. Сегмент по умолчанию DS.

CO адрес1, адрес2, количество_байт_- копирование данных из области памяти с адресом адрес1 в область памяти с адресом адрес2.

I адрес - отображение содержимого порта ввода/вывода. Адрес - любой 8-ми или 16-ти разрядный адрес или название регистра.
Пример: i 2fb
i ax

O адрес, константа - вывод 8-разрядной константы в порт ввода/вывода, параметр адрес - смотрите команду **I**.
Пример: O 2fb,00

T{B}- показать буфер трассировки. Если параметр B не указан, использован раздельный метод отображения.

BW имя_файла - записать установки точки останова в файл.

BL имя_файла - считать установки точки останова из файла.

PH адрес, длина{, имя_файла} - распечатать данный в шестнадцатеричном и ASCII формате. Сегмент по умолчанию DS. Длина определяет количество байт. По умолчанию вывод на принтер.

PD адрес1, длина{,имя_файла} - распечатать дизассемблированный код. Сегмент по умолчанию CS. Параметр длина - количество команд, которые будут распечатаны. Параметр имя_файла - имя файла, в который записываются ASCII коды дизассемблированной программы. Если этот параметр не указывается, то программа распечатывается на принтере.
Пример: PD 100, 10, aaa.txt

PT{смещение, длина{, имя_файла}} - распечатать содержимое буфера трассировки. Смещение определяет смещение первой печатаемой команды. Число печатаемых инструкций определяется длиной или действительным числом записанных команд. По умолчанию - все команды.

286 ON
OFF -переключение режима ассемблера и дизассемблера для i80286. Переключатель установлен на PC тип (PC/AT).

MO{DE} M{ONO}
C{OLOR}
A{LTERN} ON
OFF - установить режим экрана.
M - черно-белый адаптер.
C - цветной адаптер.
Если указано **A ON**, то выполнение программы будет сопровождаться отображением выводимых ею на экран данных (так называемый «теневой экран»). F6 - переключение между основным и теневым экраном. Команда без параметров возвращает текущие установки.

BE{EP} ON
OFF - включение и выключение звука.

XT - перейти в режим обучения. Все нажатия клавиш хранятся в буфере и могут быть записаны в файл или выполнены. Режим обучения завершается, когда буфер переполнен или по Ctrl+Break.

XX {имя_файла} - выполнить записанные в режиме обучения нажатия клавиш. Если указано имя файла, то данные считываются из файла и выполняются.

XW имя_файла - записать нажатия клавиш, хранящихся в буфере в файл.

XL имя_файла - заполнить буфер клавиатуры данными из файла.

CTRL-HOME - выход из ассемблерного редактора в интерактивный режим для ввода команд.

CTRL-ESC - выход из бесконечного цикла (созданного, например командой JMP).

QUIT - выход из AFD и возврат в DOS.

F1 - выполнение команды
F2 - выполнение процедуры
F3 - восстановление команды
F4 - помощь
F5 - просмотр окна трассировки
F7 - перемещение вверх
F8 - перемещение вниз
F9 - перемещение влево
F10 - перемещение вправо

3.2. Полноэкранный отладчик реального режима. Руководство пользователя

3.2.1. ВВЕДЕНИЕ

Полноэкранный отладчик SDT (аналогичен AFD) обеспечивает режим интерактивного доступа ко всем ресурсам микро-ЭВМ (ПЭВМ) и удобный интерфейс для отладки программ.

Во время работы SDT на экране отображается содержимое всех регистров процессора, четыре верхних элемента стека и до девяти строк дисассемблированного кода. Кроме того имеется два независимых окна на память, которые позволяют отображать содержимое ячеек оперативной памяти в шестнадцатеричном виде и в коде КОИ-8. Дисассемблированный код или данные в шестнадцатеричном виде или в коде КОИ-8 могут быть выведены на печать или в файл. Интерфейс с пользователем организован очень продуманно, обеспечивая максимальное удобство в работе.

Наиболее часто используемые функции могут быть выполнены с помощью функциональных клавиш. Команды имеют длину от одного до двух символов и для проверки их синтаксиса можно, не прерывая работы, вызывать вывод на экран справочной информации. Шаг выполнения программы может быть выполнен нажатием единственной клавиши. Даже процедуры, которые вызываются отлаживаемой программой с помощью CALL или INT могут быть выполнены нажатием единственной клавиши. Любая ошибка ввода вызывает печать сообщения об ошибке и курсор позиционируется на символ, где во время разбора команды была обнаружена ошибка.

Введенные команды могут быть повторно извлечены из стека команд для повторения операции с помощью функциональной клавиши.

Команды и другая информация могут быть записаны в файл или во внутренний буфер для выполнения в качестве макрокоманды. Это позволяет выполнять запуск SDT из командного файла для установки точек останова по условиям или просто для выполнения последовательности команд с помощью макроопределения. Код загруженной программы дается на экране в дисассемблированном виде, при этом поддерживается мнемоника всех команд микропроцессора типа I8088 и I80286, что позволяет читать код так же легко, как и исходный код с помощью экранного редактора текстов.

Машинные инструкции в коде программы могут быть легко изменены с помощью встроенного в SDT ассемблера. Для трансляции новой инструкции курсор устанавливается в окне дисассемблированного текста на инструкцию, которую нужно заменить, инструкция может быть изменена вводом новых символов поверх дисассемблированного текста.

Прикладные программы, использующие вывод данных на экран, могут использовать для этого другой дисплейный адаптер и экран, чтобы избежать смешения выходных данных программы с выводом на экран, выполняемым SDT. Это возможно, если на микро-ЭВМ установлены как монохромный, так и цветной графический адаптеры. При работе на ЭВМ с одним экраном пользователь может выбрать режим альтернативного экрана для разделения данных, выводимых SDT и прикладной программой. Переключение между экранами выполняется с помощью нажатия единственной клавиши.

3.2.2. НАЗНАЧЕНИЕ ПРОГРАММЫ

Полноэкранный отладчик SDT существенно облегчает отладку и анализ программного обеспечения для микро и персональных ЭВМ типа ЕС1840, Искра 1031, IBM PC/AT и др. (в реальном режиме адресации).

Эта программа предназначена в первую очередь для отладки программ, написанных на языке ассемблера, но может быть использована и для отладки программ, написанных на языках высокого уровня.

SDT рекомендуется использовать при изучении системы команд и программирования на языке ассемблера, так как он предоставляет возможность иметь в интерактивном режиме полный доступ ко всем регистрам процессора и ячейкам памяти.

Посредством SDT могут быть загружены для отладки файлы типа EXE или COM.

На пошаговое выполнение или трассировку процедур ДОС или процедур обработки прерываний BIOS не накладывается никаких ограничений.

Широкие возможности определения точек останова позволяют прерывать выполнение программы по заданному адресу, а также обеспечивают проверку указанных условий и, в случае их истинности, выполнение определенных действий, таких как трассировка, останов или сброс счетчика проходов точки останова. Можно определять до восьми точек останова, указывая их адреса, несколько условий выполнения заданных действий, объединенных операцией логического "И" (AND), а также определяя значение счетчика проходов через точку останова и сами действия, которые необходимо выполнить.

Определения точек останова могут быть сохранены в файле и впоследствии вновь использованы.

Отладчик предоставляет возможность трассировки программы. Режим трассировки может быть задан так, чтобы не протоколировались инструкции, выполняемые в процедурах обработки прерываний. Записи трассировки содержат выполненные инструкции, текущее содержимое регистров и содержимое

верхних четырех элементов стека. Записанные данные трассировки могут быть просмотрены в полноэкранном режиме, либо выведены в файл или на печать.

При работе отладчика в резидентном режиме он может быть вызван по запросу пользователя с клавиатуры или нажатием кнопки немаскируемого прерывания (NMI), если она установлена аппаратно. Это позволяет получить управление даже, если отлаживаемая программа зависла, установив флаг запрещения прерываний, и система не воспринимает ввод с клавиатуры.

Если SDT больше не используется, память, занятая SDT, может быть возвращена системе при условии, что выше SDT в памяти нет других резидентных программ.

3.2.3. УСЛОВИЯ ПРИМЕНЕНИЯ

Условия, необходимые для выполнения программы, определяются требованиями к составу технических средств и программному обеспечению.

3.2.3.1. Требования к составу технических средств

Отладчик предназначен для функционирования на 16-разрядных микро-ЭВМ на базе микропроцессорного набора К1810 ВМ 86 (или эквивалентного), таких как ЕС-1840, ЕС-1839 ("Правец-16"), Искра 1031, IBM PC/AT и др.

Минимальная конфигурация должна включать:

- 16-разрядный центральный процессор;
- оперативную память объемом не менее 256 Кбайт;
- 1 накопитель на жестком диске (необязательно);
- 1 накопитель на гибком магнитном диске (или 2 в случае отсутствия накопителя на жестком диске);
- алфавитно-цифровой дисплей или видеосистема ПК класса MDA, CGA и выше.

3.2.3.2. Требования к программному обеспечению

Отладчик можно эксплуатировать под управлением операционной системы MS-DOS версии 3.10 и выше или совместимой с ней.

3.2.4. ОБРАЩЕНИЕ К ПРОГРАММЕ

Для запуска SDT его необходимо загрузить в память с помощью ввода команды: {устр:} SDT {фспец} {парам} {"команда SDT} в ответ на подсказку ДОС (Например,'C:>').

В фигурных скобках указаны необязательные параметры. Параметр 'устр:' должен быть указан и определять дисковод, на котором находится SDT.COM (AFD.COM), если он отличен от дисковода, используемого по умолчанию. Если указан параметр 'фспец.', то для указанного файла после запуска SDT выполняется команда загрузки 'L' (подробнее она описана в п. 3.2.6.11).

После спецификации файла, если нужно, могут быть заданы параметры, которые помещаются в префиксный сегмент (PSP) отлаживаемой программы. Используя двойную кавычку (") отладчику могут быть переданы дополнительные команды. Если в SDT передается более одной дополнительной команды, они должны быть разделены двойной кавычкой ("). Обычно таким образом передается команда выполнения макрокоманды для автоматической установки параметров отладки (подробнее описание команды 'XX' дано в подразделе 3.2.6.28).

Для того, чтобы передать в SDT указание об использовании функциональной клавиши, нужно ввести '#n', где n -это номер выбранной функциональной клавиши. После указания функциональной клавиши не должно следовать знака двойной кавычки ("), так как (") транслируется в код клавиши Enter.

Соответствующие функциональные клавиши вводятся в следующем виде:

```
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10
#1 #2 #3 #4 #5 #6 #7 #8 #9 #10
```

Если SDT запускается без перечисленных необязательных параметров, то выводится экран с названием программы. Для продолжения работы необходимо нажать какую-нибудь клавишу, после чего появляется основной экран SDT (AFD).

В случае, если необязательные параметры заданы, экран с названием программы не выводится, а сразу появляется основной экран SDT общий вид которого приведен на рис. 3.1.

Если при запуске SDT не передано ни одной команды перемещения курсора, то курсор устанавливается в поле командной строки (на рисунке в командной строке набрана команда g,CS:236). (Курсор - это полностью заполненное мерцающее знакоместо, если редактирование экрана SDT выполняется в режиме замены, или половина знакоместа в режиме вставки). Все команды вводятся в этом режиме. В дальнейшем мы будем ссылаться на эту часть экрана как на *командную строку*.

Поле над командной строкой показывает содержимое всех регистров процессора и четыре верхних элемента стека. Эту часть экрана в дальнейшем будем называть *областью регистров*. Имеются два регистра сегмента - HS и FS, которые не реализованы в процессоре аппаратно. Эти два регистра могут служить как вспомогательные. Они используются также для идентификации сегмента после выполнения

команд поиска или сравнения.

Регистр HS (вспомогательный регистр сегмента) используется в командах поиска и сравнения. FS (фиксированный регистр сегмента) используется только в команде сравнения, если для адресации не может быть использовано содержимое никакого другого сегментного регистра.

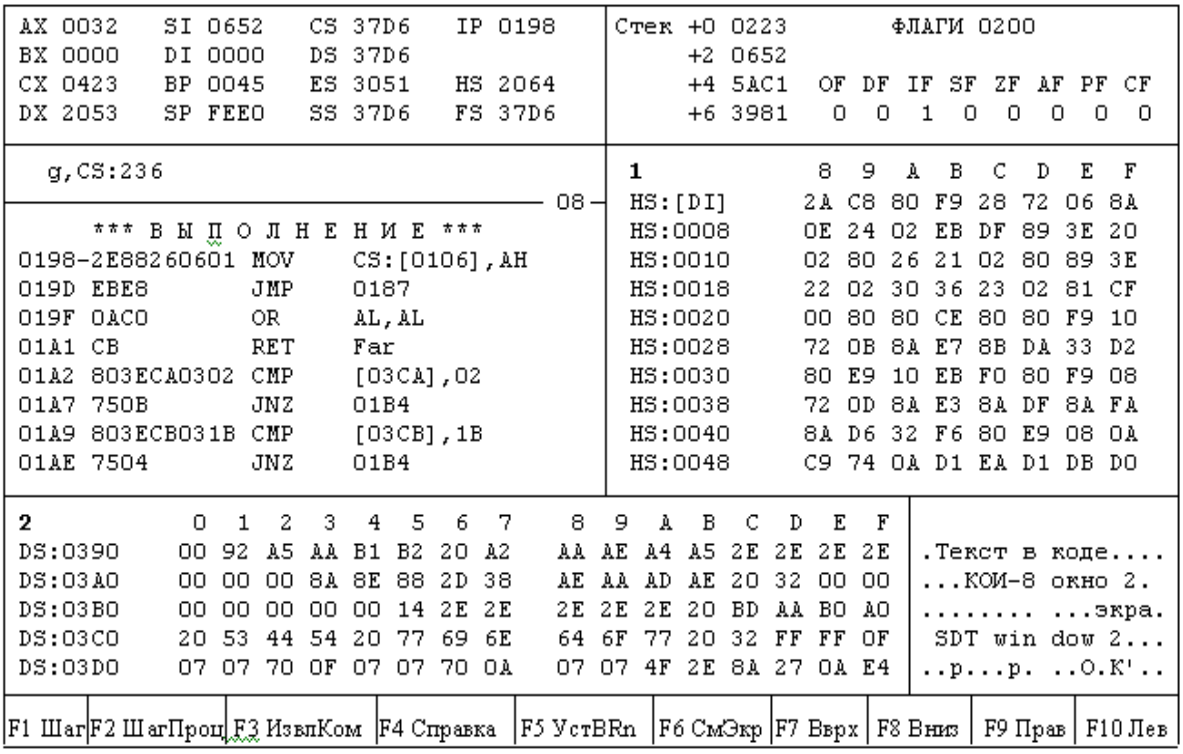


Рис 3.1. Основной экран SDT

Кроме того, эти регистры могут использоваться при определении значения регистра сегмента для адресации окна в памяти, которое не должно меняется даже, если отлаживаемая программа изменяет содержимое регистров сегментов. Пользователь может обращаться к ним так же, как к другим регистрам.

Показанные флаги процессора имеют два представления. Первое - в виде 16-разрядного слова и второе, когда каждый из восьми флаговых разрядов (OF, DF, IF, SF, ZF, AF, PF и CF) представлен двоичным значением. Все неиспользуемые разряды слова флагов установлены в ноль для облегчения интерпретации 16-разрядного слова флагов. Флаг трассировки используется SDT и поэтому он недоступен пользователю. Флаги NT и IOPL в случае микропроцессора типа i80286 не имеют значения при работе с реальными адресами, используемыми операционной системой ДОС. Поэтому их содержимое не показано отдельными значениями, а может быть проконтролировано только при интерпретации флагового регистра, представленного в виде 16-разрядного слова.

Область ниже командной строки содержит текст отлаживаемой программы в дисассемблированном виде, начиная с ячейки CS:IP. Самое левое поле в каждой строке содержит смещение адреса, следующие шестнадцатеричные значения представляют объектный код каждой инструкции. Дисассемблированный текст выглядит аналогично ассемблерному листингу, за исключением того, что не используются символические имена.

Активная строка дисассемблированной области показывается на экране в обратном фоне. Инструкция в этой строке называется *текущей инструкцией*, которая выполняется следующей по команде 'G' или 'Шаг' (функциональная клавиша F1). При выполнении команды 'Шаг' выполненная инструкция сдвигается на одну строку вверх. Строка над строкой в обратном фоне называется строкой предыдущей команды.

Эта верхняя строка дисассемблированной области (строка предыдущей команды) перекрывается выводимыми сообщениями о состоянии системы и сообщениями об ошибках, когда они появляются. Поэтому на эту строку в зависимости от информации, которая в ней представлена, ссылаются так же, как на строку состояния.

Дисассемблируемая область может быть сдвинута вверх и вниз с помощью клавиш ↑ ('курсор вверх'), ↓ ('курсор вниз'), клавиш PgUp и PgDn. При сдвиге дисассемблируемой области для отображения кода с соответствующего адреса используются все 9 строк (включая строку предыдущей команды). При сдвиге дисассемблируемой области с помощью клавиш 'курсор вверх' или PgUp в область, не содержащую кода программы, может появляться сообщение:

'Нельзя дисассемблировать назад - данные'

Если это сообщение выводится при использовании режима автоповтора клавиатуры, буфер клавиатуры может быть очищен с помощью ввода Ctrl/Break.

В случае если текущая инструкция ссылается на ячейку памяти, содержимое этой ячейки показывается в обратном фоне в конце строки, расположенной после командной строки. В зависимости от типа инструкции может быть показано значение слова или байта.

Область справа от командной строки - это окно памяти номер 1, а область под дисассемблируемой областью - окно памяти номер 2. Каждое окно в первой колонке содержит адреса, а содержимое памяти выводится в шестнадцатеричном виде. Смещение от начального адреса каждой строки показано в верхней строке каждого окна.

В каждой строке окна 1 в шестнадцатеричном виде показаны восемь байтов.

Окно 2 разделено на две области, которые используют одни и те же адреса. Левая часть отображает содержимое памяти также как и окно 1, но по 16 байтов в строке. Правая часть показывает содержимое той же области памяти, что и левая, но в коде КОИ-8 (ASCII). Адреса окон могут отличаться друг от друга.

Все символы, выведенные с повышенной яркостью, могут быть изменены пользователем вводом на их место новых значений. Курсор можно свободно передвигать в каждом окне. Для перехода из одного окна в другое нужно использовать клавиши F7-F10, как это описано в п.п.3.2. 7.3.7 - 3.2.7.3.10.

Если в команде вызова SDT указано имя файла, то выполняется загрузка этого файла, а если есть дополнительные команды, они передаются в SDT в порядке, указанном пользователем. Эти команды, а также команда загрузки остаются в стеке сохраненных команд, что позволяет выполнить их последующий просмотр для проверки.

При загрузке SDT по умолчанию вывод сообщения об ошибке сопровождается звуковым сигналом. Он может быть выключен с помощью команды BEEP OFF.

SDT проверяет также тип процессора, на котором он выполняется. В соответствии с этим устанавливается режим дисассемблера и ассемблера, определяющий используемое подмножество мнемоники машинных инструкций. Этот режим может быть изменен с помощью команды '286'.

Если по включению питания SDT нужно резидентно установить в системе, то в качестве последней команды в файл AUTOEXEC.BAT должна быть включена команда:

SDT "QUIT R

Эта команда загружает SDT и немедленно завершает его работу, оставляя загруженным в резидентном режиме. Все другие резидентные программы должны загружаться перед SDT для того, чтобы впоследствии позволить освободить память, используемую отладчиком.

3.2.5. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

SDT представляет собой COM-файл, загружаемый для выполнения операционной системой ДОС. После запуска SDT резервирует память, начиная с последнего используемого адреса, для загрузки тестируемой прикладной программы. Управление передается прикладной программе либо по команде, либо по нажатию функциональной клавиши. Если не включена трассировка, прикладная программа выполняется в реальном времени, до тех пор, пока управление не возвращается в SDT. Выполнение программы может быть прервано пользователем нажатием клавиш Ctrl/Esc, нажатием кнопки немаскируемого прерывания (NMI) (если она аппаратно установлена) или посредством определения точки останова.

Точки останова определяются пользователем с помощью меню определения точек останова или включением инструкции INT3 в код отлаживаемой программы. При обнаружении точки останова, определенной вне SDT, после останова программы выводится сообщение 'BRx'.

Когда работа SDT завершается по команде 'QUIT', вся память возвращается в общесистемный пул свободной памяти, кроме случая, когда SDT запущен в резидентном режиме по команде 'QUIT RESIDENT'. В этом случае только область памяти после SDT возвращается в систему. Для возвращения управления обратно в отладчик может быть использована кнопка NMI, клавиши Ctrl/Esc или активные точки останова.

Примечание. Процедуры ДОС не являются реентерабельными (повторно входимыми). При отладке или прерывании процедур ДОС не должно запускаться никаких функций SDT для работы с файлами или функций печати, так как SDT использует для выполнения файловых операций некоторые из функций ДОС, что в этом случае может привести к непредсказуемым результатам.

При работе в резидентном режиме отлаживаемая программа не может быть загружена с помощью команды загрузки SDT. Загрузка может быть выполнена с помощью загрузчика ДОС или посредством повторной загрузки SDT, пока одна копия выполняется в резидентном режиме. Если после SDT нет загруженных резидентных программ, то отладчик может быть выгружен из резидентного режима по команде 'QUIT' используемая память при этом возвращается в систему.

Примечание. Для передачи управления клавиатуре используется вектор прерывания 9. Поэтому драйвер клавиатуры, используемый, например, для поддержки символов кириллицы, должен быть загружен перед SDT. В общем случае вектора прерываний 1, 2, 3 и 9 не должны изменяться пока SDT

активен.

Ниже приводится краткий список функций и возможностей SDT. Подробнее они описаны в соответствующих разделах данного руководства:

- легкость использования и высокая функциональность;
- полноэкранное отображение всей необходимой информации;
- изменение содержимого регистров и памяти в режиме полноэкранного редактирования;
- автоматическая установка параметров и выполнение по командам из командного файла;
- поддержка клавиатурных макрокоманд;
- возможность получения справочной информации без прерывания отладки;
- функция извлечения ранее введенных команд;
- дисассемблирование машинных инструкций;
- встроенный ассемблер машинных инструкций;
- поддержка теневого или альтернативного экрана для вывода данных отлаживаемой программой;
- поддержка как монохромного, так и цветного адаптера;
- пошаговое выполнение программы и возможность выполнения за один шаг целиком процедуры или подпрограммы;
- резидентный режим работы и возможность последующего освобождения используемой памяти;
- возможность прерывания программы посредством:
  - ввода с клавиатуры Ctrl/Esc;
  - условных точек останова;
  - программных прерываний (INT3);
  - сигнала немаскируемого прерывания (NMI);
- определение до 8 точек останова по условиям;
- реализацию действий, выполняемых в случае истинности условий:
  - останов;
  - включение/выключение трассировки;
  - обнуление счетчика проходов;
- возможность подавления трассировки процедур INT;
- быстрая трассировка по условиям, во время которой записываются инструкции и содержимое регистров;
- одна дополнительная точка останова по указанному адресу;
- команды корректировки содержимого памяти и поиска заданных строк;
- команды копирования и сравнения областей данных;
- вывод дисассемблированного кода, данных или записей трассировки на печать или в файл;
- вычисление шестнадцатеричных арифметических выражений.

3.2.6. ОПИСАНИЕ ОСНОВНЫХ ФУНКЦИЙ

К основным функциям SDT (AFD) пользователь может обратиться с помощью функциональных клавиш и указывая в командной строке соответствующие команды полноэкранного отладчика.

В данном разделе описываются все команды SDT (AFD), которые можно вводить в командной строке. Все 28 команд описываются в алфавитном порядке. Большинство команд имеют длину 1 или 2 символа и могут быть введены как строчными, так и промежуточными буквами. Команду QUIT необходимо вводить полностью, что позволяет избежать нежелательного завершения программы.

Все символы, показанные в описании команд большими буквами, должны быть введены. Некоторые команды допускают сокращение, например, команда BEEP. Символы, которые можно опускать, заключены в скобки ( ), например BE(EP). Параметр OFF можно задавать и с одним F.

Параметры, задаваемые в командах, объяснены в п.3.2.7.1.1.

3.2.6.1. Разрешение или запрещение команд микропроцессора типа i80286

Формат команды: **286 (ON)**
(OFF)

Примеры: **286**
286 ON
286 OFF

Команда '**286**' может быть использована для разрешения или запрещения дисассемблирования дополнительных команд микропроцессора типа i80286. Ассемблер находится всегда в том же режиме, что и дисассемблер. При запуске SDT проверяет тип процессора, на котором он выполняется и соответственно выбирает режим работы.

Для определения режима работы SDT команда **286** вводится без параметров. SDT указывает состояние (ON-включен, OFF-выключен) в командной строке и подсказывает пользователю о необходимости нажать клавишу для продолжения работы.

Примечание. Когда режим 286 разрешен при работе на микро-ЭВМ (ПЭВМ) с процессором, не поддерживающим дополнительные команды, эти команды только отображаются на экране. Они не могут быть выполнены.

Особое внимание следует обратить на привилегированные команды, которые выполняются на микро-ЭВМ на базе микропроцессора типа i80286. Большинство этих команд должны использоваться только в защищенном режиме, IOPL=0. SDT подразумевает, что работа идет в режиме реальных адресов. Изменение этого режима может привести к непредсказуемым результатам.

3.2.6.2. Использование встроенного ассемблера.

Формат команды: **A {адрес}**
Пример: **A**
A CS:100

Ассемблер может быть вызван с помощью команды '**A**'. Если параметр 'адрес' не указан, то ассемблирование начинается с текущей команды. В противном случае - с указанного адреса. Выбранная ячейка должна находиться в оперативной памяти.

После ввода этой команды SDT переходит в режим ассемблирования. Курсор устанавливается на строку дисассемблированного кода, выведенную в обратном фоне, и символы в этой строке выводятся с повышенной яркостью, как во всех входных полях SDT. Командная строка заменяется на сообщение о том, что SDT находится в режиме ассемблера и для выхода из этого режима необходимо нажать Ctrl/Home.

В дополнение к обычным клавишам, поддерживающим редактирование командной строки, Tab может использоваться для перемещения курсора с поля мнемоники на поле операнда и обратно.

Клавиши ↑ ('курсор вверх') и ↓ ('курсор вниз') используются для сдвига дисассемблируемого окна, как и во время обычного ввода команды. Все введенные символы преобразуются в символы верхнего регистра.

Все функциональные клавиши поддерживаются так же как в обычном режиме ввода команд. Все изменения в ассемблируемой строке игнорируются, если они не были закончены нажатием клавиши Enter. Команды могут быть изменены или заменены как в поле мнемоники, так и в поле операндов. Введенная команда ассемблируется и, если ошибок не обнаружено, содержимое ячейки памяти изменяется на новое значение. Кроме того, окно дисассемблера перемещается, устанавливая курсор на следующую команду.

Примечание. Заменяемая команда может иметь длину в байтах отличную от длины новой команды. Это может привести к тому, что следующие инструкции будут интерпретироваться иначе.

В зависимости от режима, заданного командой **'286'** может быть использована мнемоника всех команд или только мнемоника основных команд. Если дополнительные команды не поддерживаются, выдается сообщение об ошибке:

'Ошибка мнемоники'

Поддерживается полная мнемоника инструкций, даже синонимы (такие как JB/JC). Для синонимов генерируется одинаковый код. Однако дисассемблер отображает только одно представление синонимов.

Из-за ограниченной длины строки для представления указателей выбрана специальная мнемоника. Индикатор "WORD PTR" отображается как "W/", а "BYTE PTR" - как "B/". Если для одноопернадной команды, которая указывает на ячейку памяти (например INC [1234]), не задан указатель, то по умолчанию подразумевается операция над словами.

3.2.6.3. Разрешение/запрещение звукового сигнала

Формат команды: **BE(EP) (ON)**
(OFF)
Пример: **BE**
BE ON
BEEP OFF

Эта команда разрешает и запрещает звуковой сигнал, сопровождающий вывод сообщений об ошибках. По умолчанию после запуска SDT выдача звукового сигнала разрешена.

Для ввода команды достаточно ввести первые два символа команды и параметр.

При вводе команды без параметров в командной строке выводится текущее состояние и подсказка о том, что для продолжения работы нужно нажать какую-либо клавишу.

3.2.6.4. Загрузка определений точек останова из файла

Формат команды: **BL фспец**
Пример: **BL A:TEST.SET**

Эта команда используется для чтения с диска файла определений точек останова. Может быть использовано любое имя файла, но рекомендуется в качестве расширения имени использовать ".SET". Во время выполнения дисковых операций в строке состояния выводится сообщение:

'Чтение'

и курсор исчезает.

С именем файла нельзя указывать маршрут. Файл должен находиться в текущем каталоге или на указанном диске.

3.2.6.5. Запись определений точек останова в файл

Формат команды: **BW фспец**
Пример: **BW TEST.SET**

Эта команда используется для записи текущих определений точек останова в дисковый файл. Может быть использовано любое имя файла, но рекомендуется в качестве расширения имени использовать ".SET". Во время выполнения дисковых операций в строке состояния выводится сообщение:

'Запись'

и курсор исчезает.

С именем файла нельзя указывать маршрут. Файл должен находиться в текущем каталоге или на указанном диске.

3.2.6.6. Сравнение двух областей памяти

Формат команды: **C адр.1, адр.2, длина**
Пример: **C DS:100,F000:123,%100**
C 100,HS:0123, CX*2

Эта команда сравнивает две области памяти. Для обоих адресных параметров в качестве умолчания используется текущий сегмент данных - DS. Количество байтов, которые нужно сравнить, определяется параметром 'длина'. Если указанная длина выходит за границы сегмента, адрес не изменяется циклически на первый адрес этого сегмента, а переходит на следующий сегмент.

Если все байты указанных областей совпадают, то в строке состояния выводится сообщение:

'Все байты равны'

и командная строка очищается.

В случае различия между двумя областями для вывода содержимого используются оба окна памяти. В окне 1 выводятся данные, адресуемые первым параметром, начиная с первого несовпадающего байта. Окно 2 используется для отображения второй области. Если задано абсолютное значение сегментного регистра, которое отличается от всех текущих значений сегментных регистров, то для адресации окна 1 будет использоваться регистр HS. Если HS и регистры сегмента процессора не могут быть использованы для адресации окна 2, то будет использоваться регистр FS и модифицироваться для второго окна.

Командная строка не очищается и курсор указывает на начало второго операнда. Это позволяет в случае неправильного ввода изменить указанный адрес. Для ввода новой команды строка должна быть очищена нажатием клавиши Esc.

3.2.6.7. Копирование данных из одной области памяти в другую

Формат команды: **CO исх.адр., пр.адр., длина**
Примеры: **CO DS:100, 5023:123, %100**
CO 100, HS:0123, CX*2

Содержимое ячеек памяти, указанных исходным адресом (исх.адр.), копируется в область, определенную адресом приемника (пр.адр.). Направление такое же, как для команды COPY операционной системы - из первого параметра во второй. Параметр 'длина' определяет количество байтов, которые нужно скопировать. Если указанная длина выходит за границы сегмента, адрес не изменяется циклически на первый адрес этого сегмента, а переходит на следующий сегмент.

Во время выполнения команды курсор исчезает, а командная строка содержит введенную команду. Если все байты успешно скопированы, командная строка очищается и выводится сообщение:

'Байты скопированы'

Если адрес приемника находится не в оперативной памяти, то в строке состояния выводится соответствующее сообщение об ошибке, а командная строка не очищается.

3.2.6.8. Определение начального адреса дисассемблирования

Формат команды: **D адр.**
Примеры: **D 120**
D *
D FS:100
D 123:AX+SI
D IP

Команда '**D**' используется для установки адреса начала дисассемблируемой области. Этот адрес может быть задан как сегмент и смещение.

Для указания сегмента или смещения могут быть использованы арифметические выражения. Если значение сегмента не указано, то по умолчанию используется значение регистра сегмента кода,

отображаемого в данный момент. Если сегмент указан, то это значение будет использовано для установки регистра CS, который показан в области регистров. Однако, если сегмент кода был изменен какой-либо командой, инструкция, которая должна выполняться после последней выполненной инструкции, может быть повторно выведена на экран командой **'D *'**.

Указатель команд - регистр IP не изменяется командой **D**. Значение указателя команд может быть использовано для определения смещения в виде 'IP' или '*'.

Указанный адрес должен быть началом инструкции. SDT не выполняет никакого автоматического выравнивания. После указания адреса, который не содержит первый байт допустимой инструкции, последующий код интерпретируется так, как если бы эта ячейка была началом инструкции. Неизвестный код инструкции представляется в виде псевдооперации определения байта (DB xx).

При вводе команды **D**, верхняя строка в дисассемблируемой области (строка предыдущей команды) очищается. В дисассемблируемой области отображаются 8 дисассемблированных инструкций, начиная с указанного адреса. Первая инструкция отображается в поле текущей инструкции.

При сдвиге дисассемблируемой области вверх или вниз используются все 9 строк окна дисассемблера.

3.2.6.9. Заполнение области памяти

Формат команды: **F адр, повтор, строка**
Примеры: **F 1234,1B,23 456A 'текст' 11 'еще текст'**
F CS:DI+SI, %100, 20

Команда заполнения используется для заполнения области памяти указанной строкой. Начальный адрес может быть задан с указанием сегмента или без него. Для обеих частей могут применяться арифметические выражения. Если значение сегмента не указано, то в качестве сегмента по умолчанию используется текущее содержимое регистра сегмента данных.

Параметр 'повтор' указывается с помощью четырех шестнадцатеричных цифр или десятичного значения. Он определяет сколько раз нужно повторить в памяти указанную строку.

'Строка' может быть любой комбинацией байтов, слов или строк в КОИ-8. Строки в коде КОИ-8 заключаются в одиночные кавычки. Если нет закрывающей кавычки, то все символы до конца строки считаются символами строки. Слова хранятся в оперативной памяти в таком порядке: младший байт, затем старший.

Во время выполнения команды курсор становится невидимым и в строке состояния выводится информационное сообщение. С помощью этой команды можно изменить очень большие области памяти, что может потребовать определенного времени. Операция заполнения немедленно прекращается при нажатии любой клавиши.

Если в качестве адреса указана область не оперативной памяти, то выводится сообщение об ошибке и курсор устанавливается на параметр 'адр.'. Нужно исправить адрес и выполнить команду снова или использовать клавишу Esc для очистки командной строки.

3.2.6.10. Команда G (выполнить)

Формат команды: **G {нач.адр.} {адр.ост}**
Пример: **G**
G *
G CS:100
G 123, 1100
G ,1200

Команда G (выполнить) используется для запуска отлаживаемой программы или для выхода обратно в прерванную программу, если SDT выполняется в резидентном режиме. Необходимо, чтобы адресные параметры ссылались на ячейки, которые содержат первый байт допустимых инструкций, в противном случае результат выполнения команды непредсказуем.

Начальный адрес 'нач.адр.' может быть указан как сегмент и смещение. Если значения сегмента не указано, то используется значение регистра сегмента кода, отображаемого в данный момент. Если CS: указан явно, то значением кода сегмента становится то, которое использовала последняя выполненная инструкция. Оно может отличаться от того, которое показано в области регистров экрана, если была использована команда '**D**' для отображения другого сегмента.

Команда '**G**' без указания начального адреса начинает выполнение с инструкции, которая отображена в дисассемблируемой области в обратном фоне.

Кроме восьми точек останова, из определенных с помощью меню точек останова, может быть непосредственно указана дополнительная точка останова. Если указанный адрес останова находится не в оперативной памяти, выводится сообщение об ошибке и запуск программы не выполняется. Адрес точки останова может равняться начальному адресу. Это часто используется для проверки циклов. Выполнение программы прекращается, когда указанный адрес достигается во второй раз.

При запуске в строке состояния выводится сообщение:

'* В Ы П О Л Н Е Н И Е ***'**

и курсор становится невидимым. Все установленные точки останова активизируются. При прохождении одной из 8 отдельно определенных точек останова в первой позиции строки состояния выводится номер данной точки останова. При каждом прохождении точки останова интенсивность вывода этого номера изменяется с повышенной яркости на нормальную и наоборот. Если достигнута точка останова, включающая трассировку, то в конце строки состояния выводится сообщение 'TRACE'. Этот текст удаляется из строки состояния, когда трассировка снова выключается.

Выполнение отлаживаемой программы прекращается, когда достигается дополнительная точка останова, определенная в команде '**G**' (BR0), или какая-либо из восьми точек останова, определенных через меню, в которой задано действие STOP. Номер точки останова, в которой прекратилось выполнение программы, указывается в строке состояния.

Выполнение программы прекращается также по запросу пользователя с клавиатуры или (если имеется) с помощью кнопки, вызывающей немаскируемое прерывание (NMI). Для останова программы с клавиатуры нужно одновременно нажать клавиши Ctrl и Esc.

Если в отлаживаемой программе встречается инструкция INT3, выполнение также останавливается в этой точке. SDT сообщает об этом событии, выдавая сообщение:

'Останов на BRn'

Текущей инструкцией является инструкция INT3. Ее необходимо пропустить или заменить перед тем, как выполнение программы может быть продолжено.

Когда управление возвращается в SDT, все области экрана корректируются и отображают текущие значения. Могут быть введены новые команды или старые команды могут быть извлечены из стека команд. Если SDT выполняется в резидентном режиме, это указывается в строке над командной строкой.

Когда отлаживаемая программа, загруженная с помощью SDT, завершается и возвращает управление в DOS с помощью прерывания, все регистры сбрасываются в их первоначальное состояние и все ячейки точек останова восстанавливаются и содержат свои исходные коды инструкций. В дисассемблируемой области выводится начало программы и появляется сообщение.

'Программа окончена нормально'

Примечание. Указатель стека и сегмент стека должны быть допустимыми и, по крайней мере, 6 байтов должны быть доступны для команды **G**. В противном случае результаты будут непредсказуемы.

3.2.6.11. Ввод из порта

Формат команды: **I адр**.
Примеры: **I 3EC**
I 3EC+SI
I DX
I DX-AX

Команда ' **I** ' используется для чтения любого 8-разрядного порта ввода/вывода и отображения текущего значения на экране. Адрес 'адр.' может быть арифметическим выражением.

После выдачи команды курсор исчезает и командная строка завершается знаком равенства, за которым следует шестнадцатеричное значение из порта ввода/вывода.

В строке состояния выводится сообщение:

'Нажатие клавиши - продолжение'

После нажатия клавиши командная строка и строка состояния очищаются и курсор устанавливается на начало командной строки.

Примечание. Допустимый диапазон адресов 0-3FF.

3.2.6.12. Загрузка файла в память

Формат команды: **L фспец {парам.}, {адр.}**
Примеры: **L TEST**
L TEST.COM
L DATA.DAT, DS:1234

Команда загрузки используется для загрузки в память файлов любого типа. Если расширение имени файла не указано, по умолчанию используется расширение ".EXE". Если имя файла, который нужно загрузить не имеет расширения, то имя должно заканчиваться точкой (.), чтобы избежать генерации расширения по умолчанию. Когда параметры вводятся после имени файла, эти данные помещаются в префиксный сегмент программы (PSP). По умолчанию адрес, с которого загружаются эти файлы, CS:0100. Если адрес указан без сегмента, по умолчанию используется текущее значение сегмента кода.

Во время дисковых операций курсор становится невидимым и в строке состояния выводится сообщение:

'Чтение'

Примечание. Файлы EXE и COM не могут быть загружены командой 'L', когда SDT работает в

резидентном режиме, так как это может привести к противоречиям с диспетчером памяти операционной системы. Программы должны загружаться обычным загрузчиком ДОС, либо SDT может быть вызван повторно, пока одна копия SDT уже резидентна в памяти.

Если отлаживаемая программа скомпонована программой LINK с вариантом HIGH, она не может быть загружена отладчиком SDT. Для проверки таких программ SDT должен быть запущен в резидентном режиме, после чего нужно использовать загрузчик операционной системы. Для доступа к указанной точке кода программы пользователь может поместить в эту точку инструкцию INT3.

Примечание. После загрузки программы SS:SP должен указывать на допустимую ячейку памяти перед тем, как программа будет запущена. SDT использует при передаче управления отлаживаемой программе для ее выполнения 3 верхних слова в стеке.

3.2.6.13. Установка адреса окна памяти

Формат команды: **Mn адр.**

Примеры:
M1 DS:1230
M1 ES
M2 FS:SI+1000-DI
M1 2340:[SI]
M2 SI+AX+10
M1 [SI]

Эта команда используется для установки начального адреса обоих окон памяти. 'n' может быть 1 или 2 (номер окна). Если введено имя регистра, за которым следует двоеточие (:), текущее имя регистра отображенное в указанном окне заменяется на вновь введенное имя.

Если для определения сегмента адреса используется выражение или значение, то регистр HS устанавливается равным этому значению и вместо имени сегмента в указанной области окна записывается HS. Если HS уже используется для другого окна, данная область также будет указана с помощью HS. Если это не желательно, то для окна, которое не должно изменяться, следует использовать регистр FS.

Команда '**M**' может быть использована и для изменения только сегмента. В этом случае никакого смещения вводить не нужно.

Если значение сегмента пропущено, изменяется только смещение в указанном окне. Смещение может быть любым выражением или одним значением, которое не выходит из диапазона 16-ти разрядного слова.

Для указания смещения можно также использовать имя любого 16-ти разрядного регистра в квадратных скобках (например, [DI]). Когда используется этот способ, окно памяти корректируется в соответствии с текущим значением указанного регистра. Когда проверяются строковые операции, этот режим очень удобен, так как адрес окна всегда равен содержимому используемого регистра. Этот режим будет сброшен в нормальный, когда область памяти сдвигается вверх и вниз с помощью клавиш ↑ ('курсор вверх') и ↓ ('курсор вниз') или указано другое окно.

Адрес окна памяти может быть также изменен прямой заменой символов в поле адреса окна.

3.2.6.14. Выбор режима отображения

Формат команды: **MO(de)**
MO(de) M(ono)
MO(de) C(olor)
MO(de) A(lternative) ON
MO(de) A(lternative) OFF

Команда **MODE** используется для выбора одного из 4 режимов отображения или для получения информации о том, куда выводятся данные отлаживаемой программой. Если команда **MODE** вводится без параметров, текущий режим указывается в строке состояния.

При вводе команды могут использоваться любые сокращения слов, указанных в скобках.

Первоначально при запуске SDT отлаживаемая программа и SDT используют один и тот же экран. Текущий экран системы, с которого запускается SDT, используется обеими программами.

В этом режиме вывод на экран, выполняемый отлаживаемой программой, будет смешиваться с информацией экрана SDT. Это означает, что любые данные, выводимые отлаживаемой программой, будут тут же заменяться данными с экрана SDT при каждом возврате управления. Однако этот режим очень полезен, когда отлаживаемая программа не выполняет операций отображения на экране, так как выполнение программы по шагам не вызывает переключения на альтернативный экран.

Независимо от текущего режима экрана при использовании параметра MONO в команде **MODE**, SDT переключается в режим монохромного отображения. Команды выбора режима не влияют на использование экрана отлаживаемой программой. Параметр COLOR заставляет SDT использовать цветной экран в режиме 80×25 символов. Если выбранный адаптер не установлен, выводится сообщение об ошибке и режим экрана не изменяется.

В системах, которые оборудованы только одним монитором, монохромным или цветным, альтернативный или теневой экран назначается для использования отлаживаемой программой.

Когда введена команда ALTERNATE ON, SDT и отлаживаемая программа используют один и тот же экран. Содержимое экрана и состояние курсора сохраняются и восстанавливаются каждый раз, когда управление передается в отлаживаемую программу. Вместе с разрешением использования альтернативного экрана может использоваться функциональная клавиша F6 для переключения основного экрана SDT на дополнительный и наоборот, если управление находится в SDT.

Примечание. Когда альтернативный режим используется с цветным графическим адаптером, SDT для обработки альтернативной функции использует страницу 3 этого адаптера. Для запрещения альтернативного экрана введите команду **MODE** с параметром ALTERNATE OFF.

Когда для SDT и отлаживаемой программы используются разные экраны и выполняемая программа переключается на экран, который используется SDT, альтернативный режим включается автоматически и, когда SDT получает управление, выводится информационное сообщение.

Когда SDT запущен в резидентном режиме, альтернативный режим экрана включается автоматически.

3.2.6.15. Вывод данных в порт

Формат команды: **O адр., знач.**
Пример: **O 3EC, 12**
O 3EC+SI,1230
O DX, AL
O DX-AX, AX+BX*2

Эта команда используется для посылки байта в порт ввода/вывода. Адрес 'адр.' и значение 'знач.' могут быть заданы арифметическим выражением или просто значением, не выходящим из диапазона 16-ти разрядного слова.

Значение, посылаемое в порт ввода/вывода, может быть байтом или словом. Когда значение выражения может быть представлено как байт, выполняется вывод байта, в противном случае - слова.

Примечание. Диапазон адресов портов ввода/вывода 0 - 3FF.

3.2.6.16. Корректировка содержимого памяти.

Формат команды: **P адрес, строка**
Примеры: **P 110, 12 3EC 'текст' SI**
P DS:1103, AX 1234
P *, 90

Команда корректировки используется для изменения содержимого памяти. Адрес 'адрес' ячейки, которую нужно изменить, может состоять из сегмента, и смещения. Если значение сегмента пропущено, то по умолчанию используется текущее значение сегмента кода (обычно команда корректировки используется для модификации области кода).

'Строка' может быть любой комбинацией байтов, слов и строк в КОИ-8.

Примечание. Строка в КОИ-8 заключается в одиночные кавычки ('). Если закрывающей кавычки нет, то все символы до конца командной строки считаются символами строки в КОИ-8.

При выполнении команды корректировки все области экрана (регистров, памяти и дисассемблируемая область) корректируются. При любом изменении содержимого ячеек памяти верхняя строка дисассемблируемой области (строка предыдущей команды) очищается.

При попытке изменить ячейку, которая находится не в оперативной памяти, в строке состояния выводится сообщение об ошибке и курсор устанавливается на параметр адреса в команде.

3.2.6.17. Печать дисассемблированного кода

Формат команды: **PD адр., длина {,фспец.}**
Примеры: **PD*, %100**
PD DS:1103, CX, A:TST.PRT

С помощью этой команды дисассемблированный код может быть распечатан на бумаге. Если в команде не указан сегмент в параметре 'адр.' по умолчанию используется содержимое регистра CS. Этот адрес определяет начало текста для вывода на печать. Параметр 'длина' определяет число инструкций, которые нужно распечатать. Длина может быть любым значением, которое можно представить с помощью 16-ти разрядного слова. Когда достигается конец сегмента, адрес циклически изменяется на адрес начала сегмента.

Задавая дополнительно спецификацию файла, вывод может быть направлен в файл для последующей проверки или распечатки. Для обращения к другим каталогам в спецификации файла можно указывать маршрут.

Распечатка начинается с текста '>> вывод SDT', затем печатается текущая дата и время.

Если принтер не готов или попытка создать файл закончилась неудачей, в строке состояния выводится сообщение, но введенная строка команды не стирается. Команда может быть отредактирована

или выдана повторно после приведения в готовность устройства печати.

После запуска печати на экран выводится информационное сообщение. Печать может быть прервана нажатием клавиши Esc. В случае возникновения ошибки в процессе вывода, операция завершается.

3.2.6.18. Печать данных в шестнадцатеричном виде и в коде КОИ-8.

Формат команды: **PH адр., длина {,фспец}**
Примеры: **PH *, %100**
PH DS:1103, CX, A:TST.PRT

Эта команда распечатывает данные из памяти в том же формате, в котором они выводятся в окне 2 основного экрана. Формат команды и ее действие аналогичны команде PD, описанной в подразделе 3.2.6.17. Единственная разница состоит в том, что 'длина' в этой команде определяет количество байтов, которые нужно распечатать, и в качестве сегмента по умолчанию используется содержимое регистра сегмента данных.

3.2.6.19. Распечатка записей трассировки

Формат команды: **PT {начало, длина {,фспец}}**
Примеры: **PT**
PT 10, %20
PT %10,5\WORK\TST.PRT

Эта команда используется для распечатки предварительно записанных записей трассировки. Формат печати аналогичен формату печати команды '**TB**', описанной в подразделе3.2.6.23. Команда может быть введена без параметров для распечатки всех записей трассировки, находящихся в данный момент в буфере. Если данных трассировки нет, операция завершается и в строке состояния выводится сообщение об ошибке.

Параметр 'начало' может быть указан для определения смещения до первой записи трассировки, которую нужно распечатать. Если это значение больше максимального номера записей в буфере, операция завершается.

Количество записей, которое нужно распечатать, определяется параметром 'длина'. Если это значение указывает номер записи больше максимального из записанных, выдается сообщение об ошибках. Печать прекращается, когда достигнут конец буфера трассировки.

Указывая спецификацию файла, вывод можно направить в дисковый файл. Для размещения файла в другом каталоге в спецификацию файла можно включить информацию о маршруте.

В начале операции печати основной экран заменяется отображением информации о трассировке. После печати каждой записи, изображение сдвигается вверх на одну запись. Печать может быть прервана пользователем нажатием клавиши Esc.

3.2.6.20. Завершение работы и возврат в ДОС или переход в резидентный режим

Формат команды: **QUIT {R{ESIDENT}}**

Эта команда используется для завершения работы SDT или для перевода его в резидентный режим. В любом случае определение всех точек останова сбрасывается и управление передается обратно в операционную систему. Память, занятая SDT, освобождается только тогда, когда в команде не указан параметр 'R(esident)'.

Команда **QUIT** должна быть введена полностью для того, чтобы избежать непреднамеренного завершения работы SDT. Параметр может быть сокращен до одного символа.

При завершении работы SDT перед возвратом управления в ДОС восстанавливается состояние курсора и экрана прикладных программ. При переходе в резидентный режим включается режим использования дополнительного экрана (см. подраздел 3.2.6.14.) для того, чтобы при следующем вызове SDT сохранить экран прикладной программы.

После того, как SDT сделан резидентным с помощью команды **QUIT R**, он остается активным даже, если выполняется другая программа. В этом случае STD может быть вызван в любой момент для отладки пользовательской программы. Если SDT используется в резидентном режиме, то около 64 Кбайтов памяти, которые он занимает, становятся недоступны для операционной системы.

В резидентном режиме программы не могут загружаться с помощью SDT. Если необходимо загрузить программу с помощью SDT, пользователь может запустить SDT еще раз, в то время как резидентный вариант SDT по-прежнему будет находиться в памяти.

Если память выше резидентного SDT не используется другой программой, то резидентный вариант SDT может быть "выгружен" с помощью команды **QUIT**. Это позволяет освободить используемую память.

При работе в резидентном режиме SDT может быть вызван по запросу пользователя с помощью одновременного нажатия клавиш 'Ctrl' и 'Esc, или нажатия кнопки немаскируемого прерывания (NMI), если она установлена. Если в отлаживаемой программе выполняется инструкция INT3, управление также

передается SDT.

Возможность определения условий для точек останова является очень полезной для отладки прикладных программ. При работе в резидентном режиме эта возможность должна использоваться с большой осторожностью. Если прикладная программа выполняется и SDT вызывается для установки точек останова, то оригинальный код по адресу точки останова запоминается. Эти байты будут восстановлены, когда SDT передает управление обратно. Если управление передается в прикладную программу, точки останова остаются активны. Если текущая программа заменяется на другую, и загружается SDT, то SDT пытается восстановить байты кода по адресам, указанным для точек останова.

Это может изменить код новой прикладной программы, что вызовет непредсказуемые результаты при ее выполнении.

SDT может помочь пользователю даже в той ситуации, когда SDT определяет, что код был изменен после последней команды "**G**", но сохраненный код не будет восстановлен по данному адресу и точка останова будет запрещена. Когда это происходит, выводится сообщение:

'Код изменен в BRn'

и поле счетчика для данной точки останова сбрасывается в 0. В сообщении указывается наибольший номер точки останова.

3.2.6.21. Команда установки регистра

Формат команды: **{R} регистр=значение**

Примеры:
R AX=1250
DS=AX+SI
CF=1
BX=AL*3+D-CX/AH
CX=%20

Команда установки регистров используется для установки в регистры процессора указанных значений. Для упрощения ввода команды '**R**' может быть пропущена.

Можно изменить содержимое 16-ти, 8-ми или 1-но разрядного регистра. Тип регистра и данных с другой стороны знака равенства должны совпадать, в противном случае выводится сообщение об ошибке и содержимое регистра не изменяется. Однако содержимое 16-ти разрядного регистра может быть заменено на байт. В этом случае старший байт заменяется на нулевой.

В 1-но разрядные регистры можно устанавливать только значения 0 или 1. Любые другие назначения вызывают ошибку.

Для арифметических выражений справа от знака равенства могут быть использованы знаки плюс, минус, умножение и деление ("+", "-", "*" и "/"). Все вычисления выполняются строго слева направо над 16-разрядными беззнаковыми числами. Переполнение игнорируется.

Два регистра HS и FS, не реализованные аппаратно, трактуются так же, как все остальные 16-ти разрядные регистры. Они могут быть использованы для хранения результата присвоения или преобразования десятичного числа в шестнадцатеричное.

3.2.6.22. Команда поиска

Формат команды: **S {адр., строка}**

Примеры:
S 0:0,12 3EC 'текст в КОИ-8' SI
S DS:1103, AX 1234
S

Команда поиска используется для нахождения указанной строки в памяти. Начальный адрес может быть задан в виде сегмента и смещения. Оба могут быть заданы любым арифметическим выражением. Если значение сегмента не введено, то используется текущее значение сегмента кода.

'Строка' может быть любой комбинацией байтов, слов и строк в КОИ-8.

Примечание. строки КОИ-8 заключаются в одиночные кавычки ('). Если строка КОИ-8 не заканчивается одиночной кавычкой, все остальные символы в командной строке считаются символами строки в КОИ-8.

При выполнении команды курсор становится невидимым и в строке состояния выводится сообщение:

'Поиск. Нажатие клавиши - конец'

Просматривается все адресное пространство, начиная с указанного адреса, до последней адресуемой ячейки пока строка не найдена или операция не прервана пользователем.

Когда строка обнаружена, значение HS устанавливается равным значению сегмента, а смещение до начала строки используется для отображения области памяти в окне 2.

Если строка не обнаружена, то в строке состояния выводится сообщение об ошибке и курсор устанавливается в командной строке на первый символ указанной строки.

Для повторения операции поиска той же строки достаточно ввести команду '**S**' без указания параметров. Команда поиска продолжается, начиная с байта перед тем, который был найден в последней

операции поиска. Если перед командой '**S**' без параметров не было указано никакого адреса и строки, то в строке состояния выводится сообщение об ошибке.

3.2.6.23. Отображение данных трассировки

Формат команды: **T**
TB

Эта команда используется для отображения данных трассировки. Данные трассировки могут быть отображены только, если данные были записаны во время выполнения прикладной программы. Каждая команда '**G**' сбрасывает указатель буфера трассировки. Если в буфере трассировки нет сохраненных данных, то в строке состояния выводится сообщение об ошибке и курсор устанавливается на команду.

Для отображения данных трассировки можно использовать два варианта. Когда используется команда **TB**, отображается буфер трассировки. Формат буфера трассировки приведен в подразделе 3.2.7.2.

С помощью команды **T** данные трассировки выводятся на основной экран. В командной строке выводится сообщение:

'ОТОБРАЖЕНИЕ ДАННЫХ ТРАССИРОВКИ'

и дисассемблированная область заменяется на записанные инструкции. Верхнее поле области дисассемблера выводится в обратном фоне. Содержимое всех регистров и 4 верхних элемента стека такое же, каким оно было перед выполнением инструкции. Начальная точка останова помечается как ">BRn" (n - равно номеру точки останова). Точка останова, которая прекращает трассировку, помечается с помощью "<BRn". "Top" и "End" (начало и конец) буфера трассировки указываются с соответствующими инструкциями.

Сдвиг записей трассировки позволяет имитировать функцию пошагового выполнения. Сдвиг может быть также выполнен в обратном направлении, то есть для перехода не к следующей, а к предыдущей инструкции.

В окнах памяти отображается содержимое ячеек памяти, действительное в данный момент. Эти значения не сохраняются во время выполнения трассировки.

Сдвиг экрана можно выполнять с помощью клавиш ↑ ('курсор вверх'), ↓('курсор вниз'), PgUp или PgDn. Нажатие клавиши Home отображает начало буфера трассировки. Нажатие End - последние инструкции в буфере трассировки.

Для возврата к основному экрану нужно нажать клавишу Enter или функциональную клавишу F1, как указывается в последней строке экрана.

3.2.6.24. Запись данных в файл

Формат команды: **W спец., адр., длина**
Пример: **W TEST.DAT, ES:SI, 1200**

Эта команда используется для записи данных в дисковый файл. На диск записывается указанное параметром 'длина' число байтов текущего содержимого памяти, начиная с указанного адреса 'адр'.

Адрес может быть задан как сегмент и смещение. И то и другое может быть задано как с помощью арифметических выражений, так и в виде непосредственных значений. Если значение сегмента не введено, то для определения физического начального адреса используется текущее значение сегмента данных.

'Длина' определяет количество байтов, которые нужно записать в файл. Длину можно задавать в десятичном или шестнадцатеричном виде. В шестнадцатеричном виде максимальная длина равна FFFF. Длина может быть любым арифметическим выражением или простым значением, которое можно представить с помощью 16-ти разрядного слова.

В спецификации файла ('спец.') нельзя указывать маршрут. Файл будет создан в текущем каталоге на указанном диске или диске, который используется по умолчанию.

Во время дисковых операций курсор становится невидимым и в строке состояния выводится сообщение:

'Запись'

Примечание. Максимальный размер файла в шестнадцатеричном виде FFFF байтов (64 Кбайтов).

3.2 6.25. Загрузка буфера клавиатуры (загрузка макрокоманды)

Формат команды: **XL спец.**
Пример: **XL TEST.AUT**

Эта команда используется для загрузки содержимого указанного файла, содержащего предварительно записанные последовательности нажатия клавиш, в буфер клавиатуры. Эти клавиатурные последовательности можно рассматривать как макроопределения. С помощью такой макрокоманды, переданной SDT, можно выполнять последовательность обычных команд.

Максимальная длина этого файла 130 байтов. Если файл длиннее или не начинается с идентификатора буфера клавиатуры, то в строке состояния выводится сообщение об ошибке.

Во время дисковых операций курсор исчезает и в строке состояния выводится сообщение:

'Чтение'

Посредством этой команды данные только загружаются (для выполнения макрокоманды используется команда '**XX**').

3.2.6.26. Режим обучения (генерация макроопределения)

Формат команды**:** **XT**

Когда режим обучения запускается с помощью команды "**XT**", все нажатия клавиш записываются в буфер клавиатуры. Может быть сохранено до 128 байтов. Нажатие каждой функциональной клавиши использует два байта. Эта функция может рассматриваться как генерация макроопределений в процессе работы.

Режим обучения завершается, когда буфер заполнен, или когда пользователь одновременно нажал клавиши Ctrl и Break. В случае заполнения буфера в строке состояния выводится сообщение об ошибке.

Записанные нажатия клавиш могут быть выполнены с помощью команды '**XX**' и/или могут быть сохранены в дисковом файле с помощью команды '**XW**'.

3.2.6.27. Запись информации о нажатых клавишах в файл (сохранение макроопределений)

Формат команды: **XW фспец.**

Пример: **XW TEST.AUT**

Эта команда используется для записи в файл данных о нажатых клавишах, которые в данный момент находятся в буфере клавиатуры. Если буфер клавиатуры пуст, то данные не записываются и указанный файл не создается.

Для удаления или добавления команд, файл данных о нажатых клавишах может быть отредактирован с помощью текстового редактора. Максимальный размер файла записей о нажатых клавишах не может превышать 130 байтов, включая идентификатор в первых двух байтах этого файла. Символы табуляции не сохраняются. Каждая команда должна заканчиваться символом "возврат каретки" (0D).

3.2.6.28. Выполнение записанных нажатий клавиш (выполнение макрокоманды)

Формат команды: **XX {фспец.}**

Примеры: **XX**

XX TEST.AUT

Команда '**XX**' используется для выполнения команд, сохраненных в буфере клавиатуры. Если эта команда используется без параметров, то выполняются команды из буфера клавиатуры. Если в качестве параметра добавлено имя файла, содержимое этого файла сначала считывается в буфер клавиатуры, а затем записанные данные выполняются.

Это позволяет выполнять с помощью одной команды несколько предварительно записанных макроопределений. Эти макрокоманды могут создаваться с помощью команд '**XT**' и '**XW**' или с помощью текстового редактора. Последняя выполненная макрокоманда может быть повторена еще раз без указания спецификации файла, так как макроопределение остается в буфере.

Выполнение команд из буфера клавиатуры завершается при возникновении ошибки или в случае, когда одновременно нажаты клавиши Ctrl и Break.

Если во время выполнения возникла ошибка или управление возвращается в SDT после команды '**G**', остальные команды в буфере игнорируются для предоставления доступа к программе или интерпретации результатов выполнения отлаживаемой программы.

Эта команда - наиболее часто используемый способ запуска SDT и установки определенных условий. Ниже приведен пример простого командного файла и файла записанной макрокоманды.

Командный файл:

```
MASM тест;
LINK тест;
SDT "XX DEMO.AUT
```

Содержимое DEMO.AUT:

```
L TEST
BL DEMO.SET
M2 [DI]
G, 44
```

Выполнение командного файла вызывает ассемблирование и компоновку отлаживаемой программы. Когда это завершено, SDT запускается командой, которая загружает и выполняет макрокоманду DEMO.AUT. Выполнение макрокоманды сначала загружает отлаживаемую программу. В

следующем шаге загружается файл определений точек останова DEMO.SET. Этот файл определяет точки останова и условия для запуска трассировки области кода, которую нужно оттестировать.

Окно памяти 2 устанавливается для отображения данных со смещения, которое содержится в регистре DI. Затем отлаживаемая программа запускается со своей первой инструкции и непосредственная точка останова устанавливается на ячейку 44h. Когда отлаживаемая программа достигает эту ячейку, управление передается обратно в SDT и пользователь может просмотреть текущее содержимое регистров или проанализировать с помощью записей трассировки прохождение отлаживаемой программы.

Этот пример демонстрирует способ автоматического выполнения подготовки программы и установки параметров SDT. Последние 6 команд из макрокоманды, выполненной посредством команды '**XX**', остаются доступными для просмотра в буфере сохранения введенных команд.

3.2.7. ЭКСПЛУАТАЦИЯ ПРОГРАММЫ

Эксплуатация программы SDT выполняется в интерактивном режиме. Команды, реализующие основные функции программы, описаны в разд. 6.

3.2.7.1. Описание языка запросов пользователя

Программа выполняет команды, введенные пользователем в командной строке основного экрана. Правила указания параметров приведены в п. 3.2.7.1.1.

Кроме того, выполнение отлаживаемой программы может контролироваться с помощью определения программных точек останова. Правила определения точек останова приведены в п. 3.2.7.1.2.

3.2.7.1.1. Описание параметров команд

Параметры команд показаны строчными буквами. При вводе команды они должны быть заменены действительными значениями.

Необязательные параметры, которые могут быть введены вместе с командой, при описании команд (см. разд. 3.2.6), указаны в скобках "{}".

3.2.7.1.1.1. Спецификация файла (фспец)

Спецификация файла вводится в стандартном формате ДОС и включает до восьми символов, определяющих имя файла, и необязательные три символа, определяющие расширение имени файла. Так как указанное имя должно точно определять единственный файл, символы сканирования (? и *) использовать нельзя. В спецификацию файла может быть включен идентификатор дисковода для того, чтобы обращаться к файлам, расположенным не на текущем дисководе. Указание маршрута не поддерживается.

Если расширение имени файла не указано, то подразумевается расширение ".EXE". Если пользователь хочет обратиться к файлу, не имеющему расширения имени, то имя файла должно оканчиваться точкой "." для того, чтобы избежать использования расширения по умолчанию.

3.2.7.1.1.2. Адрес (адр)

'Адрес' может быть задан как физический адрес двумя параметрами. Первый определяет значение регистра сегмента, а второй - смещение внутри сегмента. Составляющие адреса отделяются друг от друга двоеточием. Если значение сегмента не указано, используется значение по умолчанию, которое зависит от команды. Как значение сегмента, так и смещение могут задаваться как непосредственные значения.

Примеры: **100**
DI+123
DS:DI
23C:100
DS+FS:AX+SI-CX

3.2.7.1.1.3. Длина

Этот параметр определяет диапазон действия команды, например, длину файла для записи. В качестве параметра длина может быть указана как значение, которое не должно превышать 16-разрядное целое.

3.2.7.1.1.4. Значение

'Значение' может быть указано разными способами. Простое значение представляется либо шестнадцатеричным числом длиной до четырех цифр, либо именем любого 8-ми или 16-ти разрядного регистра, либо десятичным числом в диапазоне от 0 до 65535. Перед десятичными числами указывается %.

Простые значения могут объединяться в арифметические выражения с помощью арифметических операторов (+, -, *, /). Все вычисления выполняются строго слева направо без всякого приоритета. Используется 16-разрядная беззнаковая арифметика. Все остатки и переносы игнорируются.

Примеры: **AX**
1C3

%100
%80*%24
BX+DI-12
DI-FS/16
0-10

3.2.7.1.1.5. Регистр (рег)

Параметр 'рег' используется для ссылки на регистры центрального процессора. К регистрам общего назначения (AX, BX, CX и DX) можно обращаться как к 16-ти или 8-ми разрядным регистрам. Для доступа к 8-ми разрядным регистрам после первого символа в имени регистра нужно добавить 'H' или 'L' для указания старшего или младшего байта соответственно (AH, BH, CH, DH, AL, BL, CL, DL).

К регистру флагов можно обратиться как к целому 16-ти разрядному слову по имени 'FL'. К отдельным флаговым разрядам можно также обратиться по их именам (OF, DF, IF, SF, ZF, AF, PF, CF).

Флаг трассировки не указан, так как он используется SDT и не должен изменяться пользователем.

3.2.7.1.1.6. Строка

Строка используется в качестве параметра в командах поиска и заполнения области памяти. Строка может быть списком значений и строкой в КОИ-8. В списке элементы разделяются пробелами или запятыми.

Примеры: **12 23 1020 AX, 12, 'строка в КОИ-8'**
AE2, 3F, SI+CX

3.2.7.1.2. Точки останова по условию

Меню определения точек останова состоит из 5 колонок для каждой из 8 возможных точек останова. В первой колонке указывается номер точки. Во второй колонке определяется адрес точки останова. Следующая колонка - поле условия. Оно позволяет задать условие, которое проверяется каждый раз, когда во время выполнения программы достигается указанный адрес. Если условие - истина, то счетчик проходов увеличивается на единицу. Когда значение счетчика проходов становится равным значению, указанному в поле счетчика, выполняется заданное действие. Счетчик проходов может быть повторно активизирован другой или той же самой точкой останова с помощью действия RST (рестарт). Действие RST сбрасывает счетчик проходов и повторно активизирует точку останова. Если поле счетчика равно 0 или действие не указано, то данная точка останова не активна.

Проверка адреса и условия выполняются последовательно от точки останова 1 к точке останова 8 каждый раз при обнаружении останова.

Временные затраты на обработку каждой точки останова во время выполнения прикладной программы составляют от 1,5 до 2 мсек, в зависимости от числа и типа условий (процессор i80286).

Сообщения о состоянии или о допущенных ошибках выводятся в строке после строки определения точки останова номер 8.

Правильность заполнения каждого поля проверяется в момент попытки выйти из этого поля любым способом. В случае ошибки выход из поля не выполняется. Пользователь должен либо исправить ошибку, либо очистить данное поле нажатием клавиши Esc.

Возможности SDT по обработке точек останова очень широки. Возможно, что при отладке конкретной программы понадобится использовать только действия STOP и TRACE (остановка и трассировка) без указания условий.

3.2.7.1.2.1. Поле адреса

Поле адреса - это первое поле, определяемое для каждой точки останова, и оно может быть указано значением сегмента и смещения. Результирующий физический адрес должен быть адресом начала команды, иначе установка точек останова может привести к непредсказуемым результатам при выполнении прикладной программы.

Пример: **CS:013E**
14D8:1100

Двоеточие ":" не выводится с повышенной яркостью и пропускается при вводе.

Если сегмент пропущен, то по умолчанию он равен текущему значению регистра сегмента кода - CS. Сегмент может быть задан как непосредственное шестнадцатеричное значение длиной до 4-х цифр, или как имя какого-либо регистра сегмента. Если указано имя регистра сегмента, то значение сегмента точки останова будет заменено на текущее значение этого регистра в момент выхода из данного поля. Если такое определение точки останова сохраняется в файле и восстанавливается в сочетании с другим содержимым регистра сегмента, то для указания значения сегмента данной точки останова будет использоваться текущее значение. Это позволяет использовать сохраненные определения точек останова даже, когда прикладная программа размещается в другом месте памяти.

Если значение сегмента было введено в шестнадцатеричном виде, никаких преобразований значения при восстановлении его из файла не выполняется.

Примечание. Код сегмента может иметь другое значение, когда прикладная программа прерывается

с клавиатуры, если вызваны процедуры BIOS или ДОС.

Вообще точки останова устанавливаются по адресу, на который указывает значение заданного регистра сегмента. Если было указано имя регистра сегмента или его непосредственное значение, то должно быть введено и смещение.

Точка останова должна находиться в оперативной памяти (так как SDT использует программные точки останова). При попытке указать адрес точки останова не в оперативной памяти выводится сообщение об ошибке. Для этой проверки используется текущее значение указанного регистра сегмента.

Если адрес точки останова остался пустым, а точка останова разрешена ненулевым счетчиком и, по крайней мере, одним указанным действием, то условие будет проверяться при прохождении каждой точки останова. Это является средством для дополнительных проверок остальных точек останова. Так как точки останова обрабатываются последовательно, от первой до восьмой точки останова, то этот вариант рекомендуется использовать вместо определения точки останова BR8. Это гарантирует, что предварительно будут проверены и обработаны все другие условия.

3.2.7.1.2.2. Поле условия

Каждое поле условия, синтаксис определения которого показан на рис. 3.2, первоначально заполнено точками, чтобы показать возможность ввода в это поле. Для каждой точки останова может быть введено до 8 условий. Для увеличения счетчика проходов все указанные условия должны быть истинными. Это означает, что над всеми условиями в поле условия выполняется операция логического умножения (AND).

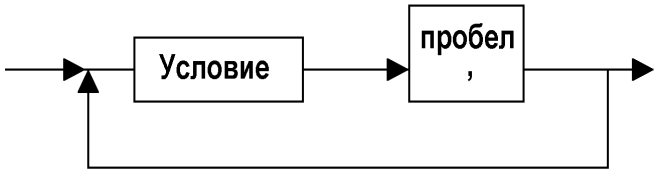


Рис. 3.2. Синтаксис условия точки останова

Условия разделяются пробелом, либо запятой. Могут быть использованы два типа условных элементов, как показано на рис. 3.3. Первый тип - это входная точка останова, а второй - это равенство или отношение.

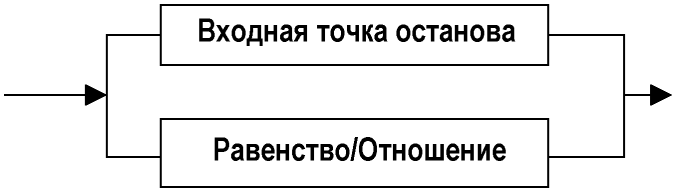


Рис. 3.3. Допустимые условия

Входная точка останова указывается в следующем формате:
BRn, где 0<n<9 и n<>i
(i - номер определяемой точки останова)

В данном случае n - номер входной точки останова. Если указана входная точка останова, условие BRn становится истиной, только тогда, когда счетчик проходов равен заданному для этой точки останова счетчику. Точка останова n должна быть активной, иначе BRn никогда не станет истиной.

Второй тип условия состоит из двух операторов, связанных знаком отношения. Синтаксис условия этого типа показан на рис. 3.4.

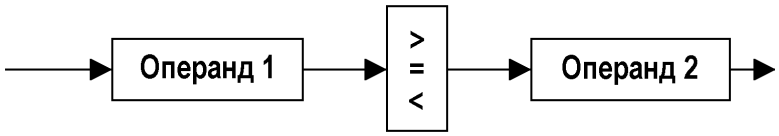


Рис. 3.4. Условие "Равенство/Отношение"

Условие становится истинно, когда равенство или отношение истинно. На рис. 3.5 показан синтаксис определения первого операнда.

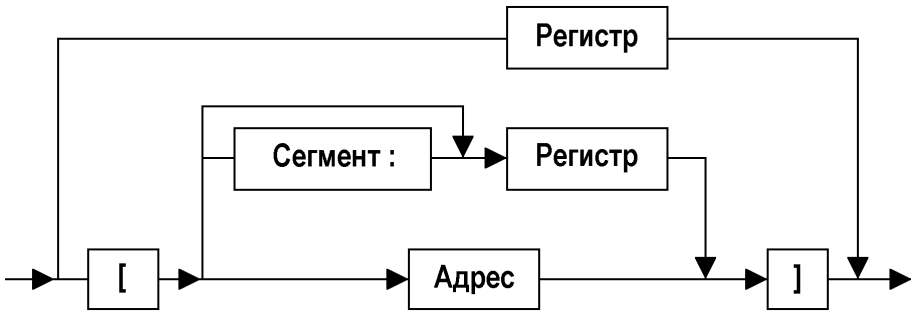


Рис. 3.5. Синтаксис определения первого операнда.

Если первый операнд - это имя регистра, то используется значение указанного регистра. Если операнд начинается с квадратной скобки ' [', то текущее значение ячейки памяти может адресоваться прямо или косвенно. При косвенной адресации указывается какой-либо из 16-ти разрядных регистров, перед которым может быть указан регистр сегмента. Если регистр сегмента не указан, то по умолчанию используется DS. Если указан, 'сегмент' должен быть именем регистра сегмента, и не может быть абсолютным значением сегмента.

Для прямой адресации памяти используется обычная форма записи адреса - 'сегмент:смещение'. Если спецификация сегмента пропущена, то по умолчанию используется DS.

Для второго операнда, синтаксис которого показан на рис. 3.6, нужно рассмотреть некоторые дополнительные варианты. Ко второму операнду можно добавлять маску или смещение до значения, на которое выполняется ссылка. Кроме содержимого регистра или ячейки памяти, в качестве второго операнда, можно указывать непосредственное значение. Значение - это шестнадцатеричное число из 4-х цифр. Внутри шестнадцатеричного числа вместо цифры может стоять 'X' для указания того, что значение этой цифры не играет роли. Такая замена может быть только для непосредственных значений. Значение может быть задано как десятичное, если перед ним указан символ '%'.

Например: **12X5**
A2XX
%1234

Если непосредственное значение не используется, то второй операнд может быть задан вместе со смещением или с маской. Для непосредственного значения в этом способе нет необходимости, так как и смещение, и маска могут быть включены в само непосредственное значение.

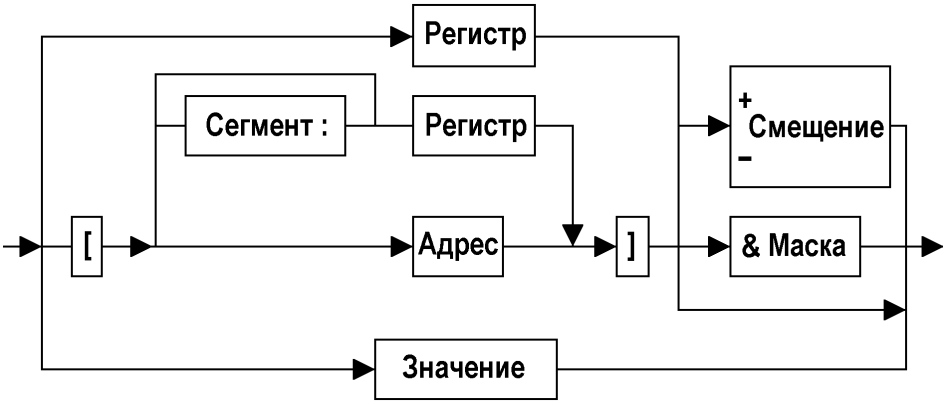


Рис. 3.6. . Синтаксис определения второго операнда.

Синтаксис указания адреса и непосредственного значения показан на рис. 3.7 и 3.8 соответственно.

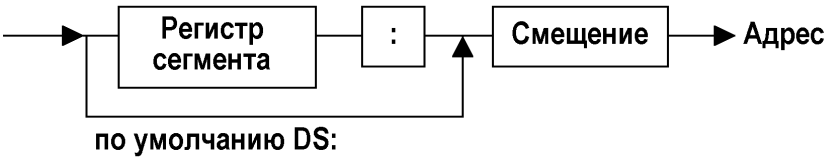


Рис. 3.7. Синтаксис указания адреса

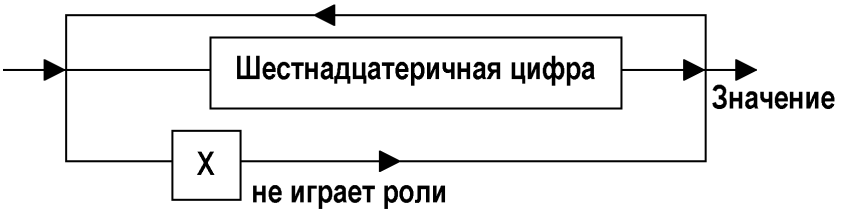


Рис. 3.8. Синтаксис указания непосредственного значения

Когда для операнда задана маска, над каждым из двух операндов выполняется операция логического "И" (AND) с указанной маской перед тем, как выполняется проверка на равенство/отношение. Это позволяет использовать для сравнения только отдельные разряды.

В случае, если дано смещение (со знаком "+" или "-"), то оно прибавляется ко второму операнду перед проверкой равенства/отношения.

Маска и смещение определяются как четырехзначные шестнадцатеричные значения, где задаются все четыре цифры. В спецификации адреса должен быть использован регистр сегмента. По умолчанию - это DS.

3.2.7.1.2.3. Поле счетчика

Поле счетчика определяет, сколько раз должна быть пройдена точка останова с истинным условием перед тем, как будет выполнено действие. Счетчик задается в десятичном виде. Максимальное значение 65535. В этом поле можно вводить только цифры.

Нулевой счетчик запрещает использование связанной с ним точки останова, даже если действие указано. Чтобы точка останова была активной, необходимо, чтобы и поле счетчика и поле действия одновременно содержали допустимую информацию.

После определения значения поля счетчика и выхода из него значение этого поля выравнивается по правому краю поля. Если поле очищается нажатием клавиши Esc, счетчик устанавливается равным нулю.

3.2.7.1.2.4. Поле действия

В поле действия пользователь определяет действие, которое выполняется, если счетчик прохождений через данную точку останова становится равным значению счетчика. В поле действия может быть указано до четырех различных действий, если длина поля достаточна для их определения.

Каждое действие может быть задано своим полным названием или любым его сокращением. При вводе сокращения оно должно быть отделено от параметра или от следующего действия хотя бы одним пробелом или запятой.

Действия могут быть заданы в следующем виде:

Count
Trace ON {NI}
Trace OFF
Rst n,m....
Stop

Действие COUNT используется для того, чтобы сделать точку останова активной, и не выполняет ничего, кроме подсчета количества прохождений точки останова перед тем как она деактивируется при достижении равенства значений поля счетчика и счетчика прохождения. Это действие обычно используется вместе с условиями для входных точек останова в поле другой точки останова. Действие COUNT можно считать фиктивным, так как увеличение счетчика выполняется неявно для всех других действий.

Действие TRACE включает (ON) и выключает (OFF) трассировку для данной точки останова. Если указан параметр NI для действия TRACE ON, то информация о выполнении процедуры, вызываемой прикладной программой с помощью команды прерывания INT, не записывается в буфер трассировки. Это бывает необходимо, когда программа делает много обращений к ДОС или BIOS. Однако, если TRACE ON указана без параметра NI, то возможно, что в буфер трассировки не записывается команда IRET. Конец процедуры прерывания легко определить, так как после возврата счетчик команд (IP) меняет свое значение.

Действие Rst используется для перезапуска точек останова. Список перезапускаемых точек останова определяется с помощью параметров n, m... . Независимо от значения счетчика прохождений через точку останова, счетчик прохождений сбрасывается в ноль и точка становится активной, если она активной не была. Это действие позволяет выполнять очень сложные проверки ошибок и операции трассировки.

Например, точка останова с действием TRACE ON может перезапускаться каждый раз, когда выполнена процедура восстановления прикладной программы после ошибки. Выполнение будет остановлено только, если восстановление после ошибки невозможно. В этом случае структурная схема программы может быть проверена по выведенной информации о трассировке.

Действие Stop выполняет останов прикладной программы по указанному адресу и передает управления обратно в SDT.

3.2.7.2. Входные и выходные данные

Входными данными для программы отладчика является загрузочный программный модуль в формате EXE или COM. А также вспомогательные файлы макрокоманд и определения точек останова, которые могут быть подготовлены с помощью самого SDT. Подробно команды создания этих вспомогательных файлов описаны в разделе 3.2.6.

Выходными данными программы могут быть данные о структуре отлаживаемой программы, информация об обнаруженных ошибках, распечатка дисассемблированного кода, расположенного по указанному адресу памяти, и данных, содержащихся в указанных областях памяти, а также листинг трассировки выполнения отдельных участков программы.

Первая колонка такого листинга показывает смещение записанных инструкций от начала сегмента кода. Следующая колонка показывает дисассемблируемую инструкцию по этому адресу. Остаток строки используется для отображения содержимого регистров и верхних элементов стека. Все значения выводятся такими, какими они были перед выполнением инструкции. Для каждой инструкции используется 4 строки.

На рис. 3.9 показан пример отображения трассировки с помощью команды TB.

Начало буфера трассировки помечается словом "Top" около соответствующей инструкции. "End" указывает последнюю записанную инструкцию. Точки останова, которые запускают и прекращают трассировку, также указываются с соответствующими инструкциями. Если первая инструкция в буфере не помечена точкой останова, которая запустила трассировку, это значит, что произошло циклическое использование буфера трассировки и в буфере остались только последние оттрассированные инструкции.

```
              ОТОБРАЖЕНИЕ  БУФЕРА  ТРАССИРОВКИ
Смещение буфера: 0
***  Начало данных TRACE  ***
1686 RET              AX=0006  SI=FFE2  CS=022B  ZF0  OF0    Стек+0 3974
                      BX=000B  DI=0023  DS=7000  AF0  DF0        +2 000C
                      CX=000C  BP=0000  ES=1ED8  PF1  IF1        +4 58C3
                      DX=0000  SP=0878  SS=022B  CF1  SF1        +6 0ECA
3974 POP    CX        AX=0006  SI=FFE2  CS=022B  ZF0  OF0    Стек+0 000C
                      BX=000B  DI=0023  DS=7000  AF0  DF0        +2 58C3
                      CX=000C  BP=0000  ES=1ED8  PF1  IF1        +4 0ECA
                      DX=0000  SP=087A  SS=022B  CF1  SF1        +6 04ED
3975 LOOP   396D      AX=0006  SI=FFE2  CS=022B  ZF0  OF0    Стек+0 58C3
                      BX=000B  DI=0023  DS=7000  AF0  DF0        +2 0ECA
                      CX=000C  BP=0000  ES=1ED8  PF1  IF1        +4 04ED
                      DX=0000  SP=087C  SS=022B  CF1  SF1        +6 0ECA
396D MOV    BX,CX     AX=0006  SI=FFE2  CS=022B  ZF0  OF0    Стек+0 58C3
                      BX=000B  DI=0023  DS=7000  AF0  DF0        +2 0ECA
                      CX=000B  BP=0000  ES=1ED8  PF1  IF1        +4 04ED
                      DX=0000  SP=087C  SS=022B  CF1  SF1        +6 0ECA
396F PUSH   CX        AX=0006  SI=FFE2  CS=022B  ZF0  OF0    Стек+0 58C3
                      BX=000B  DI=0023  DS=7000  AF0  DF0        +2 0ECA
                      CX=000B  BP=0000  ES=1ED8  PF1  IF1        +4 04ED
                      DX=0000  SP=087C  SS=022B  CF1  SF1        +6 0ECA
```

Рис. 3.9. Пример отображения трассировки с помощью команды TB

3.2.7.3. Описание способа работы с программой

Наиболее часто используемые функции могут быть вызваны нажатием функциональных клавиш. Это упрощает их использование, так как для выполнения операции достаточно нажатия единственной клавиши. В данном подразделе описываются функции всех функциональных клавиш используемых при работе с основным экраном SDT. Меню для определения точек останова также описывается в данном подразделе, так как доступ к этому меню выполняется с помощью функциональной клавиши F5.

Для того чтобы показать, какие функции доступны в текущем состоянии, назначение каждой функциональной клавиши всегда указано в последней строке экрана. Соответствие клавиш командам выбрано так, чтобы клавиши наиболее часто используемых функций можно было нажимать, не глядя на клавиатуру. Поэтому F1, наиболее часто используемая клавиша, предназначена для функции пошагового выполнения программы. Ниже перечислены функциональные клавиши и функции, которые они вызывают при работе с основным экраном:

F1 - один шаг программы;
F2 - шаг, выполняющий процедуру;
F3 - извлечение последней команды из стека команд;
F4 - вывод экрана справочной информации;
F5 - вход в меню определения точек останова;
F6 - переключение на альтернативный экран и обратно;
F7 - курсор на поле вверх;
F8 - курсор на поле вниз;
F9 - курсор на поле влево;
F10 - курсор на поле вправо.

Все функциональные клавиши поддерживаются как клавиши с возможностью автоповтора ввода. Это означает, что, когда клавиша нажата и не отпускается, вызываемая функция будет повторяться в темпе приблизительно 10 раз в секунду.

3.2.7.3.1. Выполнение одного шага программы (F1)

При нажатии F1, независимо от текущего значения указателя инструкции (IP), выполняется инструкция, показанная в дисассемблируемой области в обратном фоне. Все точки останова, определенные пользователем, игнорируются для того, чтобы разрешить неограниченное пошаговое выполнение программ обработки прерываний ДОС и BIOS.

После выполнения одиночного шага все регистры, включая указатель инструкции, корректируются и в строке с обратным фоном выводится следующая инструкция. Строка предыдущей команды показывает инструкцию, выполненную перед этим.

Если выбранный режим поддержки экрана (команда MODE описана в подразд. 3.2.6.14) использует альтернативный экран или микро-ЭВМ поддерживает два монитора, то режим экрана, используемый отлаживаемой программой, восстанавливается перед выполнением каждого шага. Если и отлаживаемая программа, и SDT используют один и тот же экран, то никаких действий для восстановления режима или содержимого экрана не предпринимается.

Выполнение одиночного шага корректирует регистр сегмента кода и указатель текущей инструкции. Это очень удобно в случае, когда нужно пропустить одну или несколько инструкций. Перед тем, как нажать F1, инструкция, которую нужно выполнить, передвигается в поле текущей инструкции с помощью клавиш ↑ ('курсор вверх') или ↓ ('курсор вниз').

Удерживая F1 в нажатом состоянии, одиночные шаги можно повторять в темпе автоповтора клавиатуры.

Примечание. Для сохранения адреса возврата в отлаживаемую программу SDT использует три слова в стеке пользователя. Поэтому в момент запуска стек отлаживаемой программы должен быть допустимым. *Инструкции, которые изменяют сегмент или указатель стека, не должны выполняться в пошаговом режиме. Они должны выполняться с помощью установки непосредственной точки останова после последней инструкции, изменяющей регистр стека.*

3.2.7.3.2. Шаг выполнения процедуры (F2)

Шаг выполнения процедуры используется для выполнения целой подпрограммы как одиночного шага. Эта функция может быть выполнена для любой инструкции и наиболее полезна при выполнении инструкций CALL, INT и LOOP.

Функция работает так же, как и функция выполнения одиночного шага, за исключением того, что так как SDT устанавливает программные точки останова, ее нельзя использовать, когда выполняемый код не находится в оперативной памяти. В этом случае должен использоваться пошаговый режим.

При попытке использовать F2 в области кода, который не находится в оперативной памяти, выводится сообщение:

'Код должен быть в ОП'

Всегда при нажатии F2 выводится сообщение:

'* В Ы П О Л Н Е Н И Е ***'**

Если время выполнения подпрограммы очень мало, это сообщение немедленно исчезает. Выполнение процедуры отлаживаемой программы может быть завершено пользователем с помощью нажатия Ctrl/Esc или клавиши NMI.

3.2.7.3.3. Извлечение последней команды (F3)

Эта функциональная клавиша позволяет извлечь в командную строку последнюю введенную команду из стека хранения команд, имеющего размер, достаточный для хранения шести последних команд. После нажатия F3 команда извлекается из стека, а нажатие Enter вызывает выполнение этой команды. Затем команда снова сохраняется в стеке хранения команд.

3.2.7.3.4. Вывод справочной информации (F4)

Эта функция обеспечивает возможность получения справочной информации непосредственно на

экране, когда выполняется ввод команды и использование документации невозможно.

Пользователю доступны четыре экрана со справочной информацией, объясняющей синтаксис и действие команд, которые можно задавать в командной строке. Вид этих экранов приведен на рис. 3.10 - 3.13 соответственно.

Символы, введенные в командной строке основного экрана, не стираются при нажатии F4 и при возврате к основному экрану курсор устанавливается на то же место в строке, где он располагался перед вызовом справочной информации.

Команды на экране справочной информации расположены не в алфавитном порядке, а по мере их важности. Вторая часть четвертого экрана объясняет общие для различных команд правила задания параметров.

Переход от одного справочного экрана к другому выполняется нажатием клавиши пробела или цифры (1 - 4), определяющей номер экрана. Для возврата к основному экрану нужно нажать клавишу Enter или F4. Нажатие других клавиш недопустимо и вызывает звуковой сигнал.

```
                        КОМАНДЫ SDT M E H Ю  1

L фспец {парам.}{,адр.}  Загрузка файла в память. Задав 'адр.' Можно
                         определить место загрузки. По умолчанию CS:0100.
                              После выполнения функции в BX,CX указано
число                                 загруженных байтов.
W фспец,адр.,длина  Запись данных в файл. Сегмент 'адр.' по умолчанию DS.
                    'длина' указывает число байтов - четыре 16-ные цифры.
{R} рег=знач.       Установка регистра. FL=знач. устанавливает регистр
                    флага как 16-ти битовый регистр. Доступ к отдельным битам
                    по их именам: OF,DF,IF,SF,ZF,AF,PF,CF.
D адр.              Вывод кода на экран. Начальный адрес дисассемблируемой
                    области равен 'адр'. По умолчанию 'Сегмент'- CS.
                    CS:n определяет CS последней выполненной команды.
M n адр.            Вывод окна памяти (n=1 или n=2).По умолчанию
                    'Сегмент' тот же, что в окне. Для вывода фиксированного
                    окна можно использовать FS. Для косвенной адресации можно
                    указать содержимое регистра (например, [SI]).
G  {старт.адр}{,адр.BRK}     Запуск программы с текущей ячейки или со
'старт.адр'.
                    Может быть определена дополнительная точка останова.
                    По умолчанию 'сегмент' для BR - текущее значение CS.
                    Запущенную программу можно прервать по Ctrl/Esc.
QUIT  {R{ESIDENT}}  Конец и возврат в ДОС. Вариант 'R' делает SDT
                    резидентным. В этом случае SDT можно вызвать по Ctrl/Esc.

                    Пробел - следующая страница или номер страницы 1 ... 4
                       F4 или ↵ - продолжение определения BR
```

Рис. 3.10. Первый экран справочной информации

```
                        КОМАНДЫ SDT M E H Ю  2

A {адр.}            Режим ассемблирования. Если адрес не указан, выбирается
                    текущая команда. Ввод Enter - ассемблирование
                    команды. Для перемещения по коду вверх и вниз
                    можно использовать клавиши управления курсором.
P адр.,строка       Замена в памяти. По умолчанию 'сегмент' - CS.
F адр.,повор,строка  Заполнение памяти указанной строкой. По умолчанию
                    сегмент адреса - DS. 'Повтор' определяет сколько
                    раз строку нужно поместить в память.
S {{адр.},строка}  Поиск данных в памяти. Если 'адр.' не указан, поиск
                    начинается с CS:0. Сегмент по умолчанию - CS. Когда
                    данные найдены,M2 отображает эту область, используя HS.
                    Команда S без параметров запускает повторный поиск.
C адр.,адр.,длина  Сравнение двух областей памяти. При несовпадении M1
                    отображает область, заданную первым параметром, M2 -
```

```
                область, заданную вторым ('Сегмент' по умолчанию - DS:)
CO ис.адр,пр.адр,длина  Копирование данных из области, заданной 1-м
парамет-
                ром в область 2-го ('Сегмент' по умолчанию - DS:)
I адр.          Ввод и отображение данных из порта В/В. 'Адр.'- 8 или
                16 битовый адр. или содержимое регистра.
O адр.,знач.    Вывод значения в порт В/В. Если 'знач.' слово, то
                выполняется операция с 16-ти разрядным словом

                Пробел - следующая страница или номер страницы 1 ... 4
                   F4 или ↵ - продолжение определения BR
```

Рис. 3.11. Второй экран справочной информации

```
                      КОМАНДЫ SDT М Е Н Ю  3

T{B}            Вывод на экран буфера трассировки. При пропуске 'B'
                вывод идет на основной экран. В противном случае
                выбирается отдельный формат вывода.
BW фспец        Запись точек останова в файл.
BL фспец        Загрузка точек останова из файла.
PH адр.,длина{,фспец}  Печать данных в КОИ-8 и 16-м виде. По умолчанию:
вывод
                на принтер, 'сегмент' - DS. 'Длина' определяет число
байт.
PD адр.,длина{,фспец}  Печать дисассемблированного кода. По умолчанию
сегмент
                - CS. 'Длина' - число команд.
PT {старт,длина{,фспец}} Печать содержимого буфера трассировки. 'Старт' -
                смещение до 1-ой распечатываемой команды. Количество
                выводимых команд определяется по 'длине' или по числу
                действительно записанных команд. По умолчанию - все.
286 ON          Включение/выключение режима для процессора 286.
    OFF         Первоначально установлен режим для процессора типа 86.
MO{DE} M{ONO}   Установка режима экрана. 'M' - монохромный адаптор,
       C{OLOR}  'C'- цветной; 'A ON' - включение возможности использо-
       A{LTERN} ON  вания при выполнении дополнительного экрана. F6 перекл
            OFF     чает основной и дополнительный экраны. Без параметров
                показывает текущие характеристики прикладного экрана.
BE{EP} ON       Вкл/Выкл звукового сигнала
       OFF
                Пробел - следующая страница или номер страницы 1 ... 4
                F4 или ↵ - продолжение определения BR
```

Рис. 3.12. Третий экран справочной информации

```
                      КОМАНДЫ SDT М Е Н Ю  4

XT              Запуск режима обучения. Все введенные коды сохраняются
                в буфере и могут быть вывердены в файл или выполнены.
                Конец режима - заполнение буфера или по Ctrl/Break.
XX {фспец}      Выполнение сохраненных кодов. Если указана 'фспец', то
                данные читаются из файла и затем выполняются.
XW фспец        Запись сохраненных кодов в файл.
XL фспец        Загрузка данных файла в буфер клавиатуры.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- -
            фспец  Спецификация файла в ДОС. Для команды 'L' по
                 умолчанию - расширение'.EXE'
            адр.  Задается как {сег:}смещен. Если 'сег' не указан, исполь-
                 зуются умолчания.'Смещен' м.б. любым значением.
                 Например,DS:SI+BX-123 или * (*=адрес следующей команды)
            рег  Любой 8 или 16 битовый регистр с основного экрана
```

```
          Например, AX или BL или SI  ...
     знач.  Содержимое регистра или байт или слово. '*' за-
          меняется содержимым IP. Десятичные значения с % (%123)
          Арифметические выражения допустимы (например,AX+BX*3/2)
          и вычисляются строго слева направо!
     строка  Список значений или строк КОИ-8 (в кавычках), разде-
          ленных пробелами или запятыми, напр,1234 BX,'ASCII' FF.

          Пробел - следующая страница или номер страницы 1 ... 4
             F4 или ↵ - продолжение определения BR
```

Рис. 3.13. Четвертый экран справочной информации

3.2.7.3.5 Вход в меню определения точек останова (F5)

При нажатии F5 на экране отображается меню определения точек останова. Вид этого меню показан на рис. 3.14. С его помощью пользователь может задать достаточно сложные определения точек останова. Правила задания отдельных полей даны в п.п. 3.2.7.1.2.1 - 3.2.7.1.2.4.

```
           МЕНЮ  ДЛЯ    ОПРЕДЕЛЕНИЯ  ТОЧЕК  ОСТАНОВА

BR#  Адрес BRK    Условие                          Счетч  СчПрх  Действие
  1  37D6:0200    ................................1       0     TRACE ON NI
  2  37D6:023E    ................................1       0     TRACE OFF
  3  37D6:0244    SI>100..........................10      0     T ON
  4  37D6:0245    SI=600 BR3......................1       0     T OFF
  5  37D6:0245    SI=1000.........................1       0     RST 2345
  6  37D6:0247    ................................2       0     STOP
  7  37D6:0000    ................................0       0
  8    CS:0       ................................0       0
================================================================================
==
Адрес окна дисассемблера                                    -
--------------------------------------------------------------------------------
---

   37D6:0198 2E88260601       MOV    CS:[0106],AH                  -
        019D EBE8             JMP    0187                          -
        019F 0AC0             OR     AL,AL                         -
        01A1 CB               RET    Far                           -
        01A2 803ECA0302       CMP    [03CA],02                     -
        01A7 750B             JNZ    01B4                          -
        01A9 803ECB031B       CMP    [03CB],1B                     -
        01AE 7504             JNZ    01B4                          -
        01B0 381ECC03         CMP    [03CC],BL                     -
================================================================================
===
1Просмотр 3ЧтнУмолчан 4Справка 5ОснвМеню 7ХрнУмолчан 8ОкноДисассм 9Сброс-
```

Рис. 3.14. Меню определения точек останова

Верхняя половина экрана содержит поля для определения точек останова, а нижняя показывает дисассемблированный код, начиная с текущей команды. Для восьми точек останова можно определить их адрес, условие, значение счетчика прохождений и выполняемое действие. Как и на основном экране, все поля, выведенные с повышенной яркостью, могут быть заполнены или изменены. На цветном мониторе введенные символы отображаются желтым цветом до тех пор, пока они не будут проверены и приняты отладчиком.

В последней строке экрана показано, какие функциональные клавиши поддерживаются и указаны назначенные им функции.

3.2.7.3.5.1. Ввод точек останова и редактирование

При первоначальном входе в меню определения точек останова курсор указывает на первый символ (отличный от пробела) в поле адреса первой точки останова. Каждый раз при перемещении курсора на следующее или предыдущее поле, курсор будет указывать на первый непустой символ нового поля. Введенные символы преобразуются в символы верхнего регистра.

В этом режиме можно использовать все клавиши редактирования и управления курсором, кроме PgUp и PgDn. Курсор можно свободно передвигать между полями определения точек останова. Когда курсор достигает конца строки, он переходит на начало следующей. Если эта строка последняя, то - на начало первой.

Переход от одного поля к другому может быть выполнен с помощью нажатия клавиш Enter или Tab. С помощью Tab возможен переход на следующее поле, а с помощью Shift/Tab на предыдущее.

Для стирания всего поля нужно нажать клавишу Esc. Каждое поле (условия, действия и др.) обрабатывается отдельно

Для стирания символа в текущей позиции курсора нажмите клавишу Del. Все символы справа от курсора сдвигаются влево каждый раз при нажатии клавиши Del. Для того чтобы удалить только что введенный символ или символ слева от курсора, используйте клавишу BackSpace.

Клавиша Ins используется для переключения между режимами вставки и замены. В режиме вставки курсор имеет вид яркого прямоугольника размером в половину знакоместа. При вводе символов в режиме вставки, все символы справа от курсора сдвигаются вправо. Последний символ строки при сдвиге вправо теряется.

Для перемещения курсора на позицию после последнего символа в строке используется клавиша End. Если поле полностью заполнено, курсор будет указывать на последний символ поля. Нажатие клавиши Home устанавливает курсор на первую позицию текущего поля. При одновременном нажатии Ctrl и Home курсор перемещается на поле адреса первой точки останова.

Клавиши PgUp и PgDn используются для перемещения вверх и вниз окна дисассемблированного кода в нижней половине экрана.

При работе с меню определения точек входа поддерживаются следующие функциональные клавиши:

```
     Enter           - переход на следующее поле;
     Home            - курсор в начало поля;
     End             - курсор после последнего непустого символа поля;
     Ins             - переключение режима ввода;
     Del             - удаление символа над курсором;
     Esc             - очистка текущего поля;
     BackSpace       - удаление символа слева от курсора;
     Ctrl/F6         - стирание до конца поля;
     Ctrl/Home       - курсор в поле адреса BR1;
     Курсор вправо    - на место следующего символа или на следующее поле,
если                 - символ последний;
     Курсор влево    - на   место предыдущего символа или на предыдущее
поле,
                      - если символ первый;
     Курсор вверх    - вверх на строку или переход на
                       последнюю, если строка первая;
     Курсор вниз     - вниз на строку или переход на
                       верхнюю строку в поле адреса;
     Tab             - следующее поле;
     Shift/Tab       - предыдущее поле;
     PgUp            - сдвиг дисассемблируемой области вверх на 8 строк;
     PgDn            - сдвиг дисассемблируемой области вниз на 8 строк;
     F1              - вывод записей трассировки
     F3              - чтение с диска файла SDT.SET;
     F4              - вывод справочной информации;
     F5              - возврат к основному экрану;
     F7              - сохранение текущих определений точек останова в
файле
                      - на диске;
     F8              - курсор в окно дисассемблера;
     F9              - сброс всех определений точек останова;
     PrtSc           - печать текущего дисассемблируемого экрана.
```

Данные проверяются и интерпретируются в момент выхода из заполненного поля после нажатия Enter, какой-либо клавиши перемещения курсора, либо клавиш F1, F5 или F7. При ошибке курсор устанавливается на первый неверный символ и выводится сообщение о чтении ошибки. Покинуть поле, в определении которого обнаружена ошибка, нельзя. Если нужно выйти из этого поля, следует либо исправить ошибку, либо очистить поле, нажав Esc перед тем, как курсор будет сдвинут на другое поле экрана.

Текущее назначение команд функциональным клавишам всегда указано в последней строке экрана. При работе в меню определения точек останова функции назначены клавишам F1, F3, F4, F5, F7 и F9. При нажатии других функциональных клавиш выводится сообщение об ошибке:

'Пустая клавиша'

3.2.7.3.5.2. Просмотр записей трассировки (F1)

Нажатие клавиши F1 вызывает отображение содержимого буфера трассировки. Если записей нет, выводится сообщение об ошибке. Формат информации в буфере трассировки подробно описан в подразделе 3.2.7.2.

3.2.7.3.5.3. Чтение определений точек останова из файла (F3)

С помощью функциональной клавиши F3 можно восстановить предварительно сохраненные определения точек останова. SDT пытается прочитать файл SDT.SET из текущего каталога. Это позволяет в каждом каталоге иметь свои определения точек останова, используемые по умолчанию.

В случае неудачи чтения выводится сообщение о том, что файл не найден.

Другой файл определения точек останова можно загрузить с помощью команды BL, допускающей указание имени файла из текущего каталога.

3.2.7.3.5.4. Вывод справочной информации (F4)

Справочная информация об определении точек останова перекрывает нижнюю половину экрана. Выход из текущего определяемого поля не происходит и, следовательно, оно не обязательно должно быть определено без ошибок.

Для продолжения определения точек останова после вывода справочной информации достаточно нажать любую клавишу. На рис. 3.15 приведен вид экрана справочной информации этого типа.

```
            МЕНЮ  ДЛЯ    ОПРЕДЕЛЕНИЯ  ТОЧЕК  ОСТАНОВА

BR#  Адрес BRK    Условие                          Счетч  СчПрх  Действие
  1  37D6:0200    ..............................    1       0  TRACE ON NI
  2  37D6:023E    ..............................    1       0  TRACE OFF
  3  37D6:0244    SI>100........................   10       0  T ON
  4  37D6:0245    SI=600 BR3....................    1       0  T OFF
  5  37D6:0245    SI=1000.......................    1       0  RST 2345
  6  37D6:0247    ..............................    2       0  STOP
  7  37D6:0000    ..............................    0       0
  8    CS:0       ..............................    0       0

Адрес BRK Адрес точки останова. Должен быть установлен на начало команды.
       {Сегмент:}Смещение. 'Сегмент' - регистр сегмента или значение.
       Если пропуск - проверка условия в каждой точке останова
Условие    Условия для проверки в точках останова. Объединяются логическим
             И(&).РЕГ=знач., [РЕГ]=знач., СМЕЩЕН=знач., BRn. Перед 'адр.'
можно
       указать регистр сегмента. BRn = истина, в случае равенства
СчПрх=Счетч.
       Значение - число до 4х 16-х цифр, 'x' - символ маскирования.
Счетч  Десятичное число проходов перед выполнением действия. '0' запрещает
BR
СчПрх  Десятичное число проходов, сделанных после команды 'G'.
Действ  Действие, выполняемое, если условие - истина. T{RACE} ON {NI} или
OFF
       - трассировка программы. NI запрещает трассировку процедур INT.
       C{OUNT}, S{TOP}, R{ST}n,m...'RST' сбрасывает СчПрх для точки BRn в 0

     Любая клавиша - продолжение определения BR
```

Рис. 3.15. Справочная информация для определения точек останова

3.2.7.3.5.5. Возврат к основному экрану (F5)

Определение точек останова может быть закончено нажатием функциональной клавиши F5 и возврата к основному экрану. Все введенные поля не должны содержать ошибок. Если введенное поле содержит ошибку, то функция клавиши F5 не будет отрабатываться до тех пор, пока не будет исправлена ошибка или поле не будет очищено нажатием Esc.

Текущее положение курсора в последнем введенном поле меню сохраняется, что позволяет быстрое переключение с одного экрана на другой.

3.2.7.3.5.6. Сохранение в файле определений точек останова (F7)

При нажатии F7 отображаемые в данный момент определения точек останова записываются в файл SDT.SET в текущем каталоге. При выполнении дисковых операций курсор исчезает и выводится информационное сообщение.

Для выполнения этой функции необходима правильность определения всех точек останова. Если в текущем вводимом поле имеется ошибка, она должна быть исправлена или поле должно быть очищено перед сохранением определений в файле.

3.2.7.3.5.7. Установка курсора на окно дисассемблера (F8)

Нажатие клавиши F8 перемещает курсор на окно дисассемблера. Он устанавливается на поле адреса первой дисассемблированной команды. Для определения нового положения окна может быть указан любой адрес.

Для перехода со строки на строку можно использовать клавиши 'курсор вверх' и 'курсор вниз'. Клавиши PgUp и PgDn работают так же, как в основном меню определения точек останова.

3.2.7.3.5.8. Сброс всех точек останова (F9)

Все определения точек останова сбрасываются при нажатии клавиши F9. Это наиболее быстрый способ запретить использование всех точек останова.

3.2.7.3.6. Переключение экрана (F6)

После разрешения использования дополнительного экрана для отлаживаемой программы (команда MO A ON) с помощью клавиши F6 можно выполнять переключение между основным экраном SDT и альтернативным экраном. Если клавиша F6 нажата, а использование дополнительного экрана не разрешено командой MODE, то выводится сообщение об ошибке:

'Эта клавиша без функции'

Признаком разрешения использования этой клавиши служит подсказка 'СмЭкр', указывающая назначение клавиши F6 в последней строке экрана.

При переключении с помощью клавиши F6 на дополнительный экран выводятся все данные этого экрана. Курсор помещается туда, где он был установлен отлаживаемой программой. Если SDT использовал точки останова в режиме трассировки, то информация, которая выводилась во время выполнения трассировки, добавляется к данным альтернативного экрана.

Последняя строка дополнительного экрана не используется или перекрывается справочной информацией SDT для того, чтобы избежать смешения информации, выводимой SDT, с выводом данных на дополнительный экран.

Для возврата к основному экрану SDT достаточно нажать любую клавишу.

3.2.7.3.7. Перемещение курсора на одно поле вверх (F7)

Курсор может быть перемещен на область вверх по экрану нажатием клавиши F7. Из командной строки с помощью F7 курсор может быть перемещен на область регистров. После перехода из командной строки и последующего возврата положение курсора в командной строке сохраняется. Введенные символы в командной строке остаются без изменений.

В области регистров курсор устанавливается на том же месте, где он находился в момент выхода из этой области. Первоначально - это верхнее левое поле регистра AX. Когда курсор находится в области регистров, нажатие клавиши F7 передвигает курсор в адресное поле окна 2. Когда курсор находится в шестнадцатеричной области окна 2, нажатие F7 перемещает курсор обратно на командную строку. Если он находится в области данных в коде КОИ-8 (правая нижняя часть экрана) нажатие клавиши F7 перемещает курсор на предыдущее поле - окно 1. Из окна 1 нажатие F7 перемещает курсор в область регистров.

3.2.7.3.8. Перемещение курсора на одно поле вниз (F8)

Нажатие клавиши F8 курсор перемещается на одно поле вниз относительно текущего. Если курсор находится в командной строке, он перемещается на адресное поле окна 2. Если текущее положение курсора - нижнее поле, то F8 помещает курсор на область регистров. Это означает, что перемещение курсора выполняется циклически - с нижнего поля экрана на верхнее и т.д.

Если курсор находится в шестнадцатеричной области окна 2, нажатие F8 помещает курсор на область данных в коде КОИ-8 окна 2.

3.2.7.3.9. Перемещение курсора на одно поле влево (F9)

Эта функциональная клавиша перемещает курсор на окно, расположенное слева от того, где в данный момент находится курсор. Также реализовано циклическое перемещение курсора, что позволяет перемещать курсор с крайнего левого поля на правое нажатием клавиши F9.

3.2.7.3.10. Перемещение курсора на одно поле вправо (F10)

Эта клавиша выполняет те же функции, что и F9, только в обратном направлении. Когда курсор находится в области регистров, клавиша F10 может быть использована для перехода к полю флагов.

3.2.7.3.11. Перемещение курсора и редактирование данных основного экрана

Курсор может быть перемещен с одной области экрана на другую нажатием клавиш F7-F10. Все символы с повышенной яркостью (или ярко зеленого цвета) могут быть изменены, если курсор указывает на них. При переходе в режим ассемблирования единственной строкой с повышенной яркостью, которую можно изменять, является дисассемблированная строка, выводимая на обратном фоне.

3.2.7.3.11.1. Область регистров

Когда курсор перемещается на область регистров, содержимое регистров может быть изменено заменой отображаемого значения. Курсор сдвигается на одну позицию вправо после ввода каждого символа или с помощью клавиши 'курсор вправо'. В случае изменения содержимого регистров допустимы только шестнадцатеричные цифры, а для изменения значения отдельных флаговых разрядов регистра флагов допустимы только цифры 0 или 1. Символы нижнего регистра при вводе преобразуются в символы верхнего регистра.

Чтобы покинуть область регистров, можно нажать какую-либо из десяти функциональных клавиш (F1-F10) или клавишу Enter для перехода на командную строку. Текущее положение курсора сохраняется и при следующем переходе в область регистров (курсор помещается в данное положение).

При нажатии клавиши End курсор помещается на последний символ данного поля, при нажатии Home - на первый. Для перехода на следующее поле можно использовать клавишу Tab или клавишу 'курсор вправо', если курсор указывает на последний символ поля. В конце каждой строки положение курсора меняется, он циклически переходит на начало следующей строки области регистров. Перемещение курсора на предыдущее левое поле может выполняться с помощью одновременного нажатия клавиши Shift и Tab или с помощью клавиши 'курсор влево', если курсор указывает на первый символ поля. Циклическое перемещение курсора выполняется так же, в случае Tab, только в обратном направлении. Клавиши 'курсор вверх', и 'курсор вниз' могут использоваться для перехода с одной строки на другую в соответствующем направлении.

С помощью нажатия Ctrl/Home курсор перемещается на начало первого поля в левом верхнем углу области регистров.

Другие функциональные и управляющие клавиши не поддерживаются и их нажатие вызывает печать сообщения об ошибке.

При работе в области регистров можно использовать следующие клавиши:

Enter - возврат в командную строку;
0-9, A-F - замена содержимого регистров (для флаговых разрядов только 0 или 1);
Home - на первый символ поля;
End - на последний символ поля;
Курсор вправо - на следующий символ;
Курсор влево - на предыдущий символ;
Курсор вверх - на предыдущую строку;
Курсор вниз - на следующую строку;
Tab - на следующее поле;
Shift/Tab - на предыдущее поле;
Ctrl/Home - на поле в левом верхнем углу (AX);
F1-F10 - выход из области регистров и выполнение соответствующей функции;
PrtSc - печать текущего экрана.

3.2.7.3.11.2. Окна памяти

Когда с помощью одной из функциональных клавиш F7-F10 курсор перемещается на область окна памяти, он позиционируется на сегментный регистр адреса. Чтобы выйти из области окна, нужно использовать функциональные клавиши или клавишу Enter. Нажатие Enter перемещает курсор в командную строку. При нажатии функциональной клавиши выполняется соответствующая функция, а затем курсор помещается в командную строку.

Первый символ в поле адреса окна имеет повышенную яркость и может быть заменен на первый символ любого допустимого адресного регистра (C, D, E, S, F, H). Символы нижнего регистра автоматически преобразуются в символы верхнего. После замены имени сегментного регистра или нажатия клавиши 'курсор вправо' курсор перемещается на начало поля смещения, пропуская символы, не имеющие повышенной яркости. Попытка переместить курсор влево с имени сегментного регистра вызывает вывод в строке состояния сообщения об ошибке.

Поле смещения может быть изменено на любое шестнадцатеричное значение. Изменение любой цифры вызывает изменение адресов в последующих строках и корректировку содержимого отображаемого окна памяти.

Так как все поля данных в окне повышенной яркости, они могут быть заменены на

шестнадцатеричные цифры. Их замена приводит к немедленному изменению содержимого соответствующих ячеек памяти. После ввода каждого символа курсор сдвигается вправо на следующий символ и строка предыдущей команды в окне дисассемблера очищается, так как это изменение может изменить код программы. В случае, если данная ячейка памяти не может быть изменена, так как она не находится в оперативной памяти, выводится сообщение об ошибке.

С помощью клавиши Home курсор перемещается в левый верхний угол данной области.

Правая половина окна 2, область КОИ-8 показывает содержимое того же диапазона адресов, что и левая. В этой части все байты памяти могут быть изменены на любой символ. Однако, символы, шестнадцатеричные коды которых не находятся в диапазоне 20 - BF, будут отображаться в виде точек.

Примечание. Это поле отделено от левой части окна 2 и на него можно перейти только с помощью функциональных клавиш F7-F10.

Для перехода к следующему символу в окне могут быть использованы клавиши 'курсор вправо' и 'курсор влево'. Пробелы при перемещении курсора пропускаются. Для перемещения курсора с поля на поле можно использовать Tab и Shift/Tab. В конце строки при перемещении вправо выполняется циклическое перемещение курсора. В конце правого нижнего поля каждого окна дальнейший сдвиг вправо не выполняется. В случае такой попытки выводится сообщение об ошибке:

'Конец входного поля'

С помощью клавиш 'курсор вверх' и 'курсор вниз' курсор перемещается в указанном направлении, если он находится в области, выделенной повышенной яркостью.

3.2.7.3.11.3. Сдвиг окна по памяти вверх и вниз

Если курсор находится в верхней строке и нажата клавиша 'курсор вверх', то курсор остается в верхней строке, но адрес уменьшается на количество байтов в строке и содержимое окна корректируется, чтобы показать новую область. То же самое выполняется в противоположном направлении, когда курсор находится в нижней строке или в поле адреса окна, выделенном повышенной яркостью, и нажата клавиша 'курсор вниз'. Это позволяет построчно перемещать окно по памяти.

Клавиши PgUp и PgDn могут быть использованы для сдвига окна на одну страницу, соответственно вверх или вниз. Длина страницы определяется числом строк в соответствующем окне. Эти клавиши могут быть использованы всегда, когда курсор находится в одном из окон памяти.

Если курсор находится в области окна, можно использовать следующие клавиши:

```
Enter        - возврат в командную строку;
0-9, A-F     - замена адресов или данных в окне;
Home        - курсор на начало поля адреса окна;
Курсор вправо  - на следующий символ;
Курсор влево   - на предыдущий символ;
Курсор вверх   - на предыдущую строку или сдвиг окна,
                 если курсор на верхней строке или в поле адреса;
Курсор вниз    - на следующую строку или сдвиг окна,
                 если курсор на последней строке или в поле адреса;
Tab         - вперед на следующее поле;
Shift/Tab   - назад на предыдущее поле;
F1-F10      - выход из области окна и запуск соответствующей функции;
PrtSc       - печать текущего содержимого экрана.
```

3.2.7.3.12. Ввод командной строки и ее редактирование

Команды SDT могут вводиться, когда курсор установлен в командной строке. Большинство команд имеют длину 1 или 2 символа. При интерпретации команд ведущие пробелы игнорируются. Для ввода команд и параметров можно использовать как прописные, так и строчные буквы.

При вводе лишних параметров выводится сообщение об ошибке, указывающее на наличие в строке лишних символов. Эти символы должны быть удалены перед выполнением команды. Они могут быть удалены нажатием клавиш Ctrl/F6, которое удаляет все символы от текущей позиции курсора до конца строки.

Когда курсор находится в командной строке, клавиши 'курсор вверх' и 'курсор вниз' используются для сдвига дисассемблируемой области. Клавиши PgUP и PgDn используются для "листания страниц" дисассемблируемой области. При попытке "листания" в обратном направлении (с помощью PgUp или 'курсор вверх') в области, которая не может быть верно дисассемблирована, выдается сообщение об ошибке и сдвиг области не выполняется.

3.2.7.3.12.1. Редактирование команд

Когда курсор находится в командной строке, он может перемещаться для редактирования команды только в горизонтальном направлении. При попытке выйти из командной строки в горизонтальном направлении выводится сообщение об ошибке:

'Конец входного поля'

Клавиши 'курсор вверх' и 'курсор вниз' используются для сдвига дисассемблированной области и не изменяют положения курсора.

Команды могут вводиться символами как верхнего, так и нижнего регистра. Для очистки всей командной строки и помещения курсора на начало вводимого поля достаточно нажать клавишу Esc. Клавиша Del используется для удаления символа в текущей позиции курсора. С помощью клавиши Backspace удаляется символ слева от текущей позиции курсора. Одновременное нажатие Ctrl и F6 удаляет символы от текущего положения курсора до конца строки.

Для вставки символов SDT должен находиться в режиме вставки. Переключение между режимами вставки и замены выполняется нажатием клавиши Ins. Когда SDT находится в режиме вставки, курсор имеет вид половины закрашенного знакоместа. В режиме замены в качестве курсора используется полностью окрашенное знакоместо. Когда команда введена без ошибок или когда курсор перемещается на другой экран или другую область экрана, восстанавливается режим замены - режим редактирования, используемый по умолчанию.

Клавиша End используется для установки курсора на последний символ строки, отличный от пробела. Если командная строка заполнена, нажатие клавиши End устанавливает курсор на последний в строке символ. Клавиша Home устанавливает курсор на начало командной строки.

После нажатия Enter команда передается SDT для интерпретации и выполнения.

Для редактирования командной строки можно использовать следующие нефункциональные клавиши:

Enter - выполнить команду;
Home - курсор в начало поля;
End - курсор после последнего символа в строке, отличного от пробела;
Ins - переключение режима ввода;
Del - удаление символа в текущей позиции курсора;
Esc - очистка командной строки;
Backspace - удаление символа слева от курсора;
Ctrl/F6 - очистка до конца строки;
Курсор вправо - позиция следующего символа;
Курсор влево - позиция предыдущего символа;
Курсор вверх - сдвиг дисассемблируемой области вниз на одну строку;
Курсор вниз - сдвиг дисассемблируемой области вверх на одну строку;
PgUp - сдвиг дисассемблируемой области вверх на 8 строк;
PgDn - сдвиг дисассемблируемой области вниз на 8 строк;
PrtSc - печать содержимого текущего экрана.

3.2.7.3.12.2. Обработка ошибок при вводе команд

Если во время ввода команды, обнаруживается ошибка, то в верхней строке дисассемблируемой области выводится сообщение об ошибке или указание пользователю. В дальнейшем эту строку будем называть строкой состояния. Она временно перекрывает строку предыдущей команды в окне дисассемблера. Дисассемблированная команда, которую замещает строка состояния, выводится вновь после нажатия любой клавиши.

Вывод сообщений об ошибке сопровождается звуковым сигналом, который может быть разрешен или запрещен с помощью команд BEEP ON и OFF. По умолчанию при запуске SDT звуковой сигнал разрешен.

При обнаружении ошибки ввода курсор указывает на первый ошибочный символ. Если пропущен параметр или ограничитель, то курсор устанавливается там, где он должен быть введен. Соответствующее сообщение указывает пользователю, что нужно ввести. В случае затруднений с помощью клавиши F4 на экран может быть выведена справочная информация.

При запуске дисковой операции, которая не может быть успешно завершена из-за каких-либо ошибок, курсор помещается на имя файла. Это позволяет после изменения имени повторить данную команду для другого файла.

3.3. Описание команд отладчика DEBUG

Отладчик DEBUG предназначен для решения широкого круга задач. К ним относятся, например, следующие задачи:

- Изучение текущего содержимого оперативной памяти;
- Редактирование отдельных секторов на флоппи дисках и на винчестере;
- Дизассемблирование COM и EXE-файлов;
- Разработка и отладка собственных программ на языке ассемблера (точнее на мнемокодах);
- Изучение работы программ и их модификация;
- Тестирование периферийного оборудования, для работы с портами ввода/вывода напрямую (в диалоговом режиме);
- Изучение системы команд процессора, прерываний BIOS и MS-DOS.

При изучении схемотехники PC/AT необходим инструмент, позволяющий напрямую работать с периферийными портами и оперативной памятью, а также создавать короткие тестовые программы для генерации нужных сигналов на системной плате и периферийных адаптерах. Отладчик наилучшим образом подходит для этих целей.

В версии MS-DOS 3.3 размер отладчика составляет 15897 байтов. Существует два способа запуска отладчика: debug ENTER и debug filename ENTER. После запуска отладчик загружается в оперативную память, а содержимое сегментных регистров CS, DS, ES, SS - на первый свободный параграф сразу после самого отладчика. Регистр IP устанавливается равным 100.

Отладчик DEBUG имеет специальный указатель адреса данных, который используется по умолчанию во многих командах отладчика.

После запуска отладчика слева на экране появляется черта [-], которая указывает на то, что отладчик ждет команду. Все числа интерпретируются отладчиком в шестнадцатеричной системе исчисления.

Рассмотрим непосредственно команды отладчика DEBUG (далее символ ↵ будет обозначать нажатие клавиши возврата каретки **(Enter)**) :

3.3.1. Команда A - установка режима ассемблирования. Эта команда позволяет вводить программы с использованием мнемокода команд процессора в оперативную память. Ввод команды отладчика

A <число>↵

заставляет его перейти в режим приема команд с клавиатуры и последовательного размещения их, начиная с адреса, равному указанному в команде. При вводе можно использовать две популярные инструкции ассемблера DB и DW. Например:

```
DB      1,2,3,"EXAMPLE"
DW      1000,2000,"FFFF"
```

Отладчик поддерживает мнемоники всех команд процессора, а также и сопроцессора 80287. При ассемблировании команд JMP и CALL по умолчанию, если это возможно, используется SHORT-вариант этих команд. Но можно указывать перед адресом перехода NEAR и FAR, что приведет к генерации соответствующих команд.

Мнемоникой оператора RET, соответствующего дальнейшему вызову CALL FAR, является RETF. Возможно, а в сомнительных случаях необходимо, использовать указатели WORD PTR или BYTE PTR. При вводе программы допускается печать префиксов CS:, ES:, SS: впереди команды в той же строке.

Пример 1.

Сначала наберите **A 100**. Затем введите программу

```
SUB     CX,CX ↵
LOOP    102 ↵
LOOP    104 ↵
INT     20 ↵
↵       ;Выход из режима A
```

Выход из режима ввода программы осуществляется нажатием клавиши ↵ после перехода к пустой строке вслед за последней командой программы. В данном примере показан метод организации задержки путем «прокрутки» двух пустых циклов. А запустить эту программу можно, набрав команду

G=100 ↵

Пример 2.

Ниже приведена программа забивки экрана символом «!»

Командой **A 200 ↵** устанавливаем ассемблерный режим с адреса CS:0200,

затем вводим программу

```
MOV     CX,1000 ↵
```

```
MOV     AX,0E21 ↵
INT     10 ↵
LOOP    208 ↵
INT     20 ↵
↵
```

В конце программ в этих примерах стоит команда **INT 20**, обеспечивающая возврат управления обратно на монитор команд отладчика. Запуск этой программы осуществляется по команде

G=200 ↵

3.3.2. Команда C - сравнение. Эта команда сравнивает побайтно две области памяти и печатает все различия между ними в форме

<адрес> <содержимое> <содержимое> <адрес>

В данной записи слева приведена информация о первой области памяти, а справа - о второй.

Пример 1.

Сравнить два блока памяти длиной 256 байт. Первый начинается с адреса 100, второй с адреса 300. Для этого надо набрать

C 100,1FF 300 ↵

Другой вариант той же команды

C 100L100 300 ↵

Пример 2.

Сравнить первые 100_{16} байт оперативной памяти с последними:

C 0:0L100 F000:FF00 ↵

Эта команда может быть полезна, например, при проверке на идентичность двух дискет. Для этого в соседние области памяти загружаются интересующие нас соответствующие блоки этих дискет, которые затем анализируются командой **C**.

3.3.3. Команда D - дамп оперативной памяти. Эта команда выводит на экран монитора адреса и содержимое указанной области оперативной памяти в шестнадцатеричной системе исчисления, а справа дает их ASCII-эквиваленты. Причем, кодовые комбинации, не имеющие символьного представления в стандарте ASCII, изображаются точками.

В строке отображается шестнадцать байт. При этом слева указывается полный адрес самого левого байта. Таким образом, в одной строке приводится шестнадцатеричный дамп, а также ASCII-дамп шестнадцати байт оперативной памяти.

Если команда D дана без параметров, то всего на экране отображается 128 байт в восьми строках. В каждой строке имеется знак «-», разделяющий 16 байт пополам: между восьмым и девятым байтами.

Пример 1.

Для того , чтобы просмотреть указатели-вектора первых тридцати двух прерываний (20h), надо ввести команду

D 0:0,7F ↵

Другой вариант без указания диапазона:

D 0:0 ↵

Первые четыре байта дают вектор прерывания INT 0, следующие четыре INT 1 и т.д.

Пример 2.

Просмотр последних шестнадцати байт из ПЗУ:

D FFF0:0 L10 ↵

Пример 3.

Просмотр области оперативной памяти, используемой BIOS:

D 40:0 L100 ↵

Если периодически несколько раз повторять эту команду, то при сравнение исходного и последующих распечаток имеются изменения в некоторых местах, в частности происходит увеличение содержимого четырех байтов, начинающейся с адреса 40:006C. Дело в том, что именно по этому адресу хранится четырехбайтный счетчик системных часов.

3.3.4. Команда E - изменение содержимого байтов. Эта команда позволяет побайтно просматривать содержимое памяти вперед и назад и, в случае необходимости, изменять содержимое просматриваемых байтов.

Ввод команды:

E <адрес> ↵

вызывает переход отладчика в режим редактирования отдельных байтов. При этом печатается содержимое текущего байта в шестнадцатеричной форме, за которым следует точка. После этого можно набрать величину нового содержимого байта, но можно и оставить содержимое байта прежним, не набирая ничего.

Затем необходимо набрать один из трех управляющих символов:

а) «пробел», что означает переход к редактированию следующего байта;

б) «ENTER». Это приведет к выходу из режима побайтного редактирования на командный уровень отладчика;

в) «-». Нажатие этого знака приведет к переходу на редактирование предыдущего байта,

Пример 1.

Изменим значение счетчика системных часов. Для этого введем команду **E 40:6C** ↵ и наберем четыре числа 70 70 70 70, разделенных пробелами, затем символ возврата каретки (↵). Далее необходимо выйти из отладчика и выполнить команду **time** операционной системы.

3.3.5. Команда F - заполнение. Эта команда позволяет заполнить содержимое указанного диапазона оперативной памяти повторяющейся цепочкой байтов заданными значениями. В частности, когда цепочка состоит из одного байта, эта команда позволяет обнулить нужную область памяти, когда байт равен 00, а также занести во все биты единицу, заполняя байтом равным FF.

Команда имеет следующий синтаксис:

F <диапазон_памяти> <цепочка_байтов> ↵

Пример 1.

Команда

F 100 L4000 00 ↵

заполнит область памяти, начинающуюся с адреса DS:0100 нулями общим числом 16384 байта (так как имеется указатель L4000). Это можно проверить командой

D 100 L4000 ↵

Пример 2.

Команда

F 5000 L1000 00 FF ↵

заполнит отрезок памяти длиной 4096 байт (1000h) повторяющейся парой байтов 00 FF, начиная с адреса DS:5000.

Пример 3.

Следующая команда F заполняет область памяти, являющуюся текстовым дисплейным буфером. Результат будет немедленно виден на экране:

F 800:0000 L1000 41 05 41 15 41 85 ↵

3.3.6. Команда G - запуск программы. Эта команда предназначена для запуска программы на исполнение. При этом, если той области памяти, где хранятся данные, передается управление как программе, компьютер «зависает», и требуется его повторный запуск. В случае, когда в программе имеются серьезные ошибки (например, отсутствует оператор INT 20 возврата на командный уровень отладчика), попытка исполнить ее с помощью оператора G также приводит к зависанию компьютера.

Этот оператор обычно используется в одной из следующих четырех форм:

а) G ↵

Исполнение этого оператора сводится к передаче управления адресу -CS:IP. Значения CS и IP можно узнать, набрав команду R(Enter) дампа всех регистров процессора;

б) G=<адрес> ↵

При этом производится запуск по указанному адресу. Например, команда **G=FFFF:0** ↵ приводит к запуску процедуры **POST**;

в) G <адрес> ↵

В этом случае программа запускается с адреса CS:IP и при достижении команды с указанным адресом осуществляется BREAK-остановка исполнения программы. Распространенной ошибкой начинающих является пропуск знака = при использовании команды G. В этом случае указанный адрес воспринимается как адрес останова, и если CS:IP указывал, например, на область данных, то компьютер «зависнет».

г) G=<адрес> <другой_адрес> ↵

В этом случае производится запуск программы с указанного после знака = адреса и в случае достижения программой команды с величиной адреса, указанного вторым (другой _адрес), происходит остановка.

Такой способ остановки программы называется введением контрольной точки останова. При указании контрольной точки останова соответствующий байт команды по этому адресу заменяется командой INT 3, имеющей размер 1 байт. При достижении контрольной точки исходное значение байта восстанавливается.

Пример.

Сначала введем программу (с помощью команды **A 100**):

```
MOV   BX,10 ↵
MOV   WO[BX+110],1234 ↵
DEC   BX      ;здесь поставить контрольную точку ↵
JNE   103 ↵
INT   20 ↵
↵
```

Если запустить программу командой

G=100 109 ↵

Программа, дойдя до адреса 109, остановится и выдаст полный дамп регистров.

3.3.7. Команда H - шестнадцатеричная (гекс) арифметика. Эта команда позволяет получить сумму и разность двух указанных в команде шестнадцатеричных чисел.

Пример 1.

H 19F 10A ↵

Получаем на экране в ответ 02A9 (сумму) и 0095 (разность) .

Пример 2.

H 0 1 ↵

В ответ получим 1 (сумма) и FFFF (разность).

3.3.8. Команда I - ввод из порта. Эта команда позволяет прочесть содержимое порта ввода/вывода с указанным адресом, распечатав его на экране в шестнадцатеричном виде.

Синтаксис команды:

I <адрес_порта> ↵

Пример 1.

I 61 ↵

При этом мы получим содержимое системного порта B.

Пример 2.

I 40 ↵

Эта команда даст состояние младшего байта счетчика системных часов.

Исполнение команды **I 42 ↵** с ее повторением даст нам младший и старший байты счетчика, связанного с динамиком.

3.3.9. Команда L - загрузка с диска. Эта команда позволяет загружать как логические сектора с флоппи-дисков и винчестеров, так и отдельные файлы.

Загрузка секторов с диска производится командой:

L <адрес> <номер_диска> <начальный_сектор> <число_секторов> ↵

Здесь <адрес> означает начальный адрес в оперативной памяти, начиная с которого будет последовательно размещаться содержимое блоков-секторов. Переменная <номер диска> указывает с какого диска будет производиться загрузка. Число 0 означает диск A, число 1 -диск B, число 2 - диск C и т. д. Следующие две переменные соответственно указывают с какого сектора начинается чтение и общее число прочитанных секторов.

Пример 1.

Загрузить BOOT-блок с флоппи-диска A: в оперативную память, начиная с адреса DS:1000, для чего необходимо выполнить команду

L 1000 0 0 1 ↵

Затем содержимое BOOT-блока можно распечатать командой

D 1000 L200 ↵

и

U 1000 L200 ↵

Пример 2.

Загрузка BOOT-блока винчестера с диска C: в оперативную память, начиная с адреса DS:2000 осуществляется командой

L 2000 2 0 1 ↵

Затем этот блок можно просмотреть так же, как в примере 1 командами:

D 2000 L200 ↵ и **U 2000 L200** ↵

Пример 3.

Загрузка таблицы размещения файлов FAT с диска A: (на 360K) производится командой

L 1000 0 1 2 ↵

Затем просмотреть FAT можно командой

D 1000 L400 ↵

Для сравнения эти команды можно выполнить сначала для пустого отформатированного флоппи-диска, а потом с записанными на нем двумя-тремя файлами.

Пример 4.

Загрузка корневого каталога флоппи-диска на 360K, находящегося на диске A: и его просмотр

L 1000 0 5 7 ↵

D 1000 LE00 ↵

Загрузка файлов в оперативную память производится двумя командами **N** и **L**. Сначала надо командой **N** указать отладчику имя необходимого файла:

N <имя_файла> ↵

Далее необходимо исполнить команду **L** без аргументов. В результате этих действий файл (за исключением файлов с расширением EXE), будет загружен в оперативную память, начиная с адреса CS:100. Если загружаемый файл имеет расширение EXE, то отладчик расшифрует PSP и в соответствии с полученной при этом информацией загрузит файл, начиная с адреса CS:0. Число прочитанных байтов хранится в регистровой паре BX:CX после выполнения команды **L**.

Пример 5.

Загрузка файла COMMAND.COM в оперативную память для его возможного изучения:

N C:\COMMAND.COM ↵

L ↵

3.3.10. Команда M - копирование

содержимого части оперативной памяти в другую область оперативной памяти.

Синтаксис команды следующий:

M <диапазон> <начальный_адрес> ↵

Здесь «диапазон» указывает на копируемую область памяти, а «начальный адрес» - на адрес первого байта, начиная с которого размещаются скопированные данные. Эту команду можно использовать для временного сохранения содержимого оперативной памяти с ее последующим восстановлением.

Пример 1.

В начале просмотрим содержимое памяти:

D 100 L100 ↵

Затем сохраним часть памяти, начиная с адреса

M 100 L100 1000 ↵

Затрем исходное содержимое

F 100 L100 0 ↵

Проверим обнуление

D 100 L100 ↵

И, наконец, восстановим исходное содержимое

M 1000 L100 100 ↵

Эту команду можно также использовать для редактирования программ, отлаживаемых или разрабатываемых в среде отладчика. Особенно она полезна при вставке новых операторов в тело программы.

Пример 2.

Войдите в режим ассемблирования командой

A 100 ↵

и наберите программу следующего вида:

```
MOV     CX,10
MOV     AX,0E21
LOOP    103
INT     20
```

Эта программа должна в цикле 10_{16} раз напечатать восклицательный знак !. Однако, команда вызова прерывания **INT 10** была пропущена, хотя должна находиться перед командой цикла LOOP. Для того, чтобы исправить ошибку можно заново набрать конец программы, начиная с адреса CS:0106. В случае больших программ это весьма затруднительно, поэтому вставку можно произвести, сдвинув конец программы в часть

верхних адресов памяти командой M. Затем вставить необходимую цепочку команд, и второй командой M присоединить хвостовую часть программы в конец добавленных операторов. В данном примере соответственно сделаем следующее:

M 106 L10 200 ↵

тем самым мы произведем сдвиг конца программы.

LOOP 103
INT 20.

Далее исполним команду

A 106 ↵

и перейдем в режим ассемблирования. Введем пропущенную инструкцию **INT 10** и выйдем обратно на монитор отладчика - на черту.

После этого выполним команду

M 200 L10 108 ↵

и тем самым мы присоединим «хвост» программы.

3.3.11. Команда N - указание имени. Эта команда, прежде всего, определяет имя файла, который далее либо будет считываться с диска командой **L**, либо записываться на диск командой **W**.

Синтаксис этой команды следующий:

N <имя_файла> ↵

Пример 1.

Прочтем с диска в оперативную память файл AUTOEXEC.BAT. Для этого выполним

N C:\AUTOEXEC.BAT ↵

L ↵

В регистровую пару BX:CX, будет занесена длина загруженного файла в байтах.

Пример 2.

Создадим на диске файл TEST.DAT, состоящий из нулей длины 1000_{16} байт. Выполним следующую цепочку команд:

F 100 L1000 0 ↵

N TEST.DAT ↵

R BX ↵

0 ↵

R CX ↵

1000 ↵

W ↵

Эту же команду **N** можно использовать для ввода дополнительных параметров командной строки.

Пример 3.

Предположим, что мы решили изучить программу форматирования FORMAT.COM. Загрузим ее в оперативную память, являющуюся рабочей областью отладчика. Прежде чем начинать трассировку работы программы, необходимо указать имя диска, который собираемся форматировать. В этом случае мы должны использовать следующие команды:

N C:\DOS\FORMAT.COM ↵

L ↵

N A: ↵

T=100 ↵

Имеются три области памяти, которые могут быть изменены командой **N**. Ими являются:

- область размером 16 байт с адреса CS:5C. Это FCB для файла 1;
- область размером 16 байт с адреса CS:6C. Это FCB для файла 2;
- область размером 128 байт, начинающаяся с адреса CS:80. Здесь хранится число символов в строке после команды **N**, а начиная с байта CS:81 хранится сама строка символов после команды **N**.

Проверьте это, используя команду распечатки **D**.

3.3.12. Команда O - вывод данных в порт. Эта команда позволяет вывести указанный байт в порт с заданным адресом. Она полезна для тестирования работы периферийных устройств и периферийных БИС на системной плате в диалоговом режиме.

Синтаксис команды:

O <адрес_порта> <величина> ↵

Здесь вместо «адреса порта» подставляется адрес необходимого порта в диапазоне 0-FFFF, а вместо

«величина» - значение, которое затем загружается в порт. Если порт шестнадцатибитный, то «величина» может быть четырехразрядным шестнадцатеричным числом, которое загружается в порт.

Пример 1.

Мотор дисковода A: для флоппи-дисков можно включить следующей командой

O 3F2 10 ↵

Пример 2.

Для компьютеров с CGA-контроллером можно включить окаймление дисплея командой

O 3D9 <цвет_окаймления> ↵

где <цвет_окаймления> может быть числом из интервала 0 - F.

Пример 3.

Покажем, как можно включить звуковой сигнал путем обращения к порту 061h. Для этого сначала прочитаем содержимое порта командой

I 61 ↵

запомним полученное число. Далее добавим к этому числу константу 3 и по команде **O** выведем в порт 61, полученное в результате сложения число.

3.3.13. Команда P - высокоуровневая трассировка. Эта команда также, как и команда **T**, предназначена для трассировки COM и EXE-программ. Однако, в отличие от T-трассировки, эта трассировка менее детальна. Она не отслеживает досконально подпрограммы и программные прерывания, а также циклы LOOP и строковые инструкции с повторением. Но, как и в случае T-трассировки после каждой инструкции печатается содержимое всех регистров и следующей исполняемой инструкции.

Синтаксис команды:

P=<адрес> <число_инструкций> ↵

Можно опустить любой из двух параметров командной строки: «адрес» и/или «число инструкций». Параметр «адрес» задает начальный адрес, начиная с которого будет производиться трассировка, а параметр «число инструкций» будет указывать общее число инструкций, которые будут исполнены после нажатия клавиши (Enter) . Обязателен ввод символа = при указании адреса для того, чтобы не спутать адрес с числом исполняемых инструкций.

Этот тип P-трассировки крайне полезен при изучении общей логики выполнения программы, избавляя от утомительного отслеживания деталей. При начальном изучении программы необходимо начинать именно с P-трассировки.

Пример 1

Введем, начиная с адреса 100, следующую программу

```
MOV     AX,B800 ↵
MOV     DS,AX ↵
MOV     BX,0 ↵
MOV     CX,0800 ↵
MOV     AX,0321 ↵
MOV     [BX],AX ↵
INC     BX ↵
INC     BX ↵
INT     20 ↵
↵
```

Затем проведем P-трассировку. Для этого сначала введем команду

P=100 ↵

и далее последовательные

P ↵

После этого проведем обычную детальную T-трассировку

T=100 ↵

T ↵

Пример 2.

Введем программу печати символа «A», начинающегося с адреса 100:

```
MOV     AX,0E41 ↵
INT     10 ↵
INT     20 ↵
```

↵

и проведем сначала P-трассировку:

P=100 ↵

P ↵

P ↵

Заметим, что оттрассировались только эти три инструкции «целиком». Теперь дадим команду

T=100 ↵

и затем многократные

T ↵

Увидим, что при исполнении команды INT 10 мы перешли в другую область памяти, а именно в область BIOS.

3.3.14. Команда Q - выход из отладчика. Команда **Q** приводит к выходу из отладчика на следующий верхний программный уровень. При этом рабочие файлы не сохраняются.

3.3.15. Команда R - дамп/коррекция регистров. Эта команда позволяет просматривать содержимое всех регистров сразу, а также флаги, или просматривать значения отдельных регистров и регистра флагов с возможностью их изменения.

Команда

R ↵

дает распечатку всех регистров, а также команды, на которую указывает CS:IP. Команда

R <имя-регистра> ↵

дает распечатку содержимого указанного регистра, и затем печатает двоеточие на следующей строке. После этого пользователь может нажать клавишу ↵ и выйти обратно на монитор команд отладчика, либо задать новое значение регистра перед нажатием клавиши ↵. В последнем случае старое значение регистра заменится на новое.

Перечислим возможные имена регистров:

AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP, F (регистр флагов).

При использовании команды

R F ↵

надо учитывать, что кодирование значений флагов производится весьма специфическим образом. Ниже приводится таблица кодировки, где символьная пара слева соответствует значению 0 соответствующего флага, а справа - значению, равному 1 (флаг поднят):

флаг дополнительного переноса	NA	AC
флаг нуля-результата	NZ	ZR
флаг знака	PL	NG
флаг маскирования прерывания	DI	EI
флаг направления	UP	DN
флаг переполнения	NV	OV

При исполнении команды

R F ↵

значения флагов печатаются в строку в порядке, обратном к порядку в таблице. Сразу же за этим в строке печатается знак черты «-», отладчик переходит в состояние ожидания ввода с клавиатуры. Если пользователь нажмет только клавишу ↵ то значения флагов не изменятся и произойдет возврат на монитор команд отладчика. Однако, перед этим пользователь может набрать новые значения некоторых флагов согласно кодировочной таблицы. Порядок следования при этом не существенен.

3.3.16. Команда S - поиск упорядоченного набора байтов. Эта команда позволяет провести поиск указанной цепочки байт в заданном диапазоне оперативной памяти и имеет синтаксис:

S <диапазон памяти> <цепочка байтов>↵

Ответ выдается в виде списка адресов начиная с которых располагается указанная цепочка байтов.

Пример.

Предположим, мы хотим определить встречается ли цепочка «DOS» в первых 32К оперативной памяти. Для этого необходимо ввести команду

S 0:0 L8000 44 4F 53 ↵

3.3.17. Команда T - трассировка. Эта команда позволяет исполнить одну или несколько инструкций в режиме трассировки с печатью содержимого всех регистров и мнемоники следующей декодированной исполняемой инструкции. После трассировки текущей инструкции указатель **IP** сдвигается

так, что он указывает на следующую инструкцию.

T ↵ производит трассировку одной текущей инструкции, на которую указывает CS:IP, с соответствующим изменением IP

Команда **T <число>** ↵ производит последовательную трассировку нескольких инструкций, общее число которых задастся в команде.

Команда **T <адрес>** ↵ трассирует одну инструкцию по указанному адресу. Общий вид команды **T** следующий:

T =<адрес> <число_инструкций> ↵

Пример 1.

Команда

T=FFFF:0 T < число > ↵

производит трассировку первой исполняемой инструкции при первоначальном запуске компьютера.

Пример 2.

Введем программу, начиная с адреса CS:100.

```
MOV     AX, 123 ↵
MOV     BX, 456 ↵
MUL     BX ↵
INT     20 ↵
↵
```

Затем, исполнив команду отладчика

T =100 3 ↵

мы получим представление о выполнении этой программы умножения двух чисел.

3.3.18. Команда U - дизассемблирование. Эта команда позволяет дизассемблировать .COM и .EXE -программы, загруженные в рабочую область отладчика.

Общий синтаксис:

U <начальный_адрес> <длина> ↵

или

U <начальный_адрес>, <конечный_адрес> ↵

Команда

U ↵

вызывает дизассемблирование 32 байт, начиная с байта, на который указывает CS:IP с выводом результирующего листинга на дисплей.

Если при запуске отладчика переадресовать вывод в файл, то можно получить листинг дизассемблированной программы. При этом команду U и команду Q (выход из отладчика) придется набирать «вслепую» без эхопечати на дисплее.

Пример.

Дизассемблирование BIOS компьютера PC/AT можно (в первом приближении) произвести по команде

U F000:E000 L2000 ↵ Однако, часть полученного кода будет бессодержательной, так как отладчик дизассемблировал также программу и области данных, хранимых в BIOS.

Эта проблема не снимается другими отладчиками, которые также нуждаются в знании того, какая часть файла программы отведена под данные.

Опыт практической работы подсказывает следующее решение данной задачи. Во-первых, прежде всего необходимо провести символьный дамп всей программы и выписать адреса тех ее частей, которые содержат осмысленный текст. Далее эти части будут исключены из процесса дизассемблирования. Во-вторых, надо дизассемблировать командой U оставшиеся «куски» программы. При этом надо внимательно изучать появляющийся листинг. Присутствие псевдокоманд DB, обильное использование косвенной адресации, последовательные команды с занесением в один и тот же регистр различных значений, условные переходы без предварительных команд с установкой флагов позволяет сделать вывод, что отладчик дизассемблирует область данных.

Начало такой области данных можно определить, найдя команду JMP, RET, IRET, после которой и начинается программная «бессмыслица». Для уточнения конца таких областей данных полезна таблица адресов условных и безусловных переходов, а также адресов входа в подпрограммы, пометив которые, можно определить конец областей данных.

В результате такой работы мы сможем найти дополнительные области данных.

В третьих, делаем, можно сказать, окончательный листинг программы, чередуя команды U и D для кодовых областей и областей с данными.

Обычно требуется совсем незначительное уточнение к листингу, полученному таким образом.

3.3.19. Команда W - запись на диск. Эта команда позволяет записывать на диск (флоппи-диски или винчестер) в последовательные блоки указанный диапазон оперативной памяти, а также записывать в файлы.

Запись в секторы на диск производится командой

W <адрес> <номер_диска> <начальный_сектор> <число_секторов> ↵

Эта команда записывает на диск не более 80h.

Параметр «адрес» означает адрес оперативной памяти, начиная с которого содержимое памяти копируется на диск.

Параметр «номер диска» указывает накопитель, на который производится запись (0-диск A:, 1-диск B:,..., 80 - винчестер C:,).

Параметр «начальный сектор» указывает номер логического сектора, начиная с которого будет производиться загрузка образа оперативной памяти.

Параметр «число секторов» указывает общее количество записываемых на диске секторов.

Пример 1.

Предположим, необходимо заменить диагностические сообщения в BOOT-секторе на свои собственные. Для этого выполним последовательность действий:

загрузка BOOT-сектора:

L 100 80 0 1 ↵

Затем, используя команду **E**, проведем редактирование диагностических сообщений (при этом ни в коем случае нельзя испортить текст программы). И, наконец, запомним отредактированный загрузчик на диске:

W 100 80 0 1 ↵

Запись в файл на диске производится командой **W** без аргументов. Однако, предварительно необходимо командой **N** указать имя файла, куда будет производиться запись, а также в пару регистров BX:CX занести длину записываемого файла. Начало области памяти, откуда будет производиться запись, всегда по умолчанию является байтом с адресом CS:0100. Если же используется команда

W <адрес> ↵

для записи в файл, то параметр <адрес> указывает начало области памяти, копируемой в файл.

Пример 2.

Скопируем в файл на диске содержимое BIOS-области оперативной памяти. Предполагая наличие PC/AT со стандартным BIOS размером 64K, обнулим сначала регистр BX, а затем занесем в регистр CX 2000H. Далее зададим имя файла, куда будем копировать старшую половину BIOS, командой

N BIOS.COM ↵

И, наконец, команда .

W F000:E000 ↵

занесет в файл BIOS.COM содержимое BIOS.

Пример 3.

Запомним загрузчик BOOT как файл BOOT.COM на диске. Для этого в регистровую пару BX:CX занесем 0000:0200. Затем исполним команды:

L 100 80 0 1 ↵

N BOOT.COM ↵

W ↵

В результате на диске появится файл BOOT.COM, исполнение которого приведет к инициализации системы.

В данном разделе использованы материалы из [14].

3.4.Система команд микропроцессора i8086

3.4.1. Форматы команды

Микропроцессор i8086 относится к классу однокристальных с фиксированной системой команд. При рассмотрение системы команд удобно обращаться к программной модели МП, содержащей его функциональные узлы (регистры), доступные программисту (рис. 3.16). Общие регистры разбиты на две группы: 1) группа HL, состоящая из регистров AX, BX, CX, DX, которые предназначены прежде всего для хранения данных и допускают раздельную адресацию их старших (H) и младших (L) половин; 2) группа PI, содержащая указательные регистры BP, SP и индексные регистры SI, DI, в которых обычно хранится адресная информация. Для общих и сегментных регистров указаны коды, используемые в форматах команд для их адресации. Регистр флагов F и указатель команд IP адресуются в командах неявно.

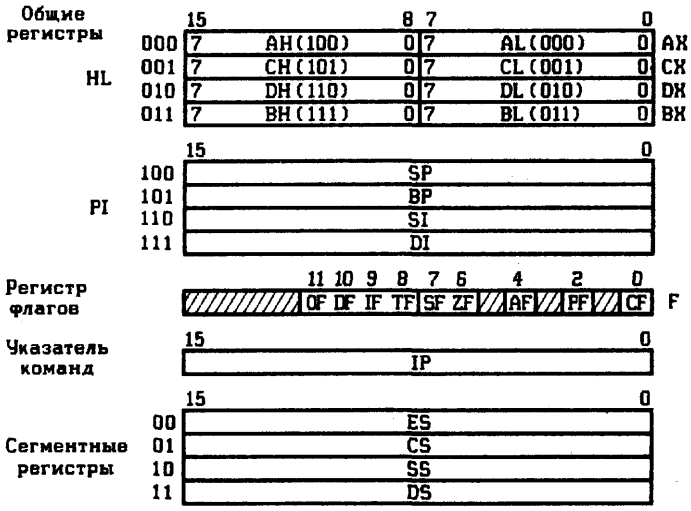


Рис. 3.16. Программно-доступные регистры МП ВМ86

Команды МП ВМ86 могут адресовать один или два операнда, причем двухоперандные команды являются, как правило, симметричными, так как результат операции может быть направлен на место любого из операндов. Однако в таких командах один из операндов должен обязательно располагаться в регистре, поскольку имеются команды типа регистр — регистр, регистр — память и память — регистр, но команды типа память — память отсутствуют (за исключением команды пересылки цепочки байт или слов).

В общем виде формат двухоперандной команды приведен на рис. 3.17, *а,* где штриховыми линиями обозначены необязательные байты команды. Первый байт команды содержит код операции COP и два однобитовых поля: направления d и слова w. При d = 1 осуществляется передача операнда или результата операции в регистр, который определяется полем reg второго байта команды; при d = 0 — передача из указанного регистра. Поле w идентифицирует тип (разрядность) операндов: при w = 1 команда оперирует словом, при w = 0 — байтом.

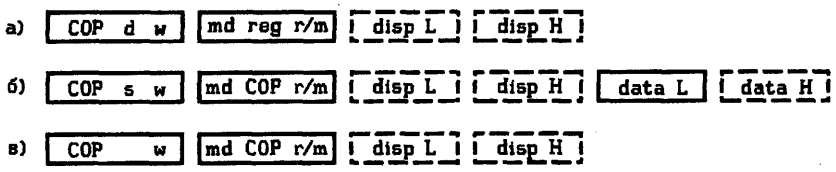


Рис. 3.17. Форматы команд

Второй байт, называемый постбайтом, определяет участвующие в операции регистры или регистр и ячейку памяти. Постбайт состоит из трех полей: md — режим, reg — регистр, r/m — регистр/память. Поле reg определяет операнд, который обязательно находится в регистре МП и условно считается вторым операндом. Поле r/m определяет операнд, который может находиться в регистре или памяти и условно считается первым. Способ кодирования внутренних регистров МП в полях reg и r/m представлен в табл. 3.1.

Таблица 3.1

reg,	Регистр		reg,	Регистр	
r/m	w=0	w=l	r/m	w=0	w=l
000	AL	AX	100	AH	SP
001	CL	CX	101	CH	BP
010	DL	DX	110	DH	SI
011	BL	BX	111	BH	DI

Отметим, что поле reg используется для указания регистра только в двухоперандных командах. Если в

команде один операнд, то он идентифицируется полем r/m, а поле reg используется для расширения кода операции.

Поле md показывает, как интерпретируется поле r/m для нахождения первого операнда: если md = 11, то операнд содержится в регистре, в остальных случаях — в памяти. Когда адресуется память, поле md определяет вариант использования смещения disp, находящегося в третьем и четвертом байтах команды:

md = { 00, disp = 0—смещение отсутствует;
{ 01, disp = disp L—команда содержит 8-битовое смещение, которое расширяется со знаком до 16 бит;
{ 10, disp = disp H, disp L—команда содержит 16-битовое смещение.

При md = 11 реализуется косвенная адресация памяти и поле r/m определяет правила формирования эффективного адреса EA операнда в соответствии с табл. 3.2, где disp означает смещение, заданное в формате команды.

Таблица 3.2

Поле r/m	Эффективный адрес EA	Адресация
000	BX+SI+disp	Базово-индексная
001	BX+DI+disp	
010	BP+SI+disp	
011	BX+DI+disp	
100	SI+disp	Индексная
101	DI+disp	
110	BP+disp	Базовая
111	BX+disp	

Приведенные в табл. 3.2 правила имеют одно исключение, позволяющее реализовать прямую (абсолютную) адресацию: если md = 00 и r/m = 110, то EA = disp H, disp L. Таким образом, имеется три варианта интерпретации поля md и восемь вариантов интерпретации поля r/m, что дает 24 варианта вычисления эффективного адреса EA. Суммарные сведения о постбайтовых режимах адресации ЦП BM86 приведены в табл. 3.3 и на рис. 3.18.

Таблица 3.3

Поле r/m	Поле md				
	00	01	10	11	
				W = 1	W = 0
000	BX+SI	BX+SI+D8	BX+SI+D16	AL	AX
001	BX+DI	BX+DI+D8	BX+DI+D16	CL	CX
010	BP+SI	BP+SI+D8	BP+SI+D16	DL	DX
011	BP+DI	BP+DI+D8	BP+DI+D16	BL	BX
100	SI	SI+D8	SI+D16	AH	SP
101	DI	DI+D8	DI+D16	CH	BP
110	D16	BP+D8	BP+D16	DH	SI
111	BX	BX+D8	BX+D16	BH	DI

Примечание. D8 = disp L(однобайтовое смещение);
D16 = diap H, disp L (двухбайтовое смещение).

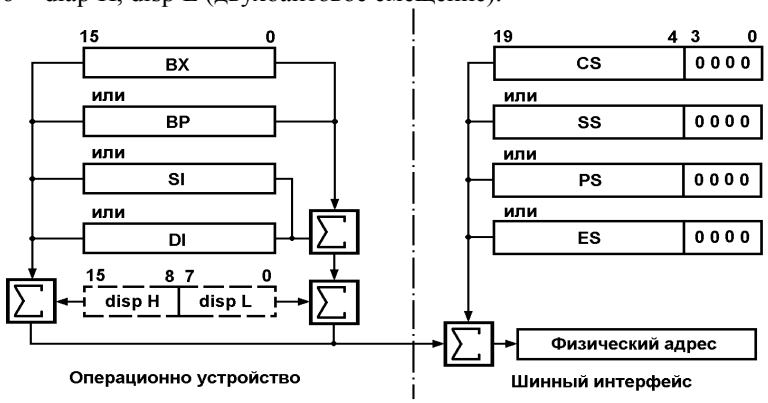


Рис. 3.18. Вычисление физического адреса памяти

Рассмотренная интерпретация полей md и r/m справедлива для всех форматов команд с постбайтовой адресацией.

Подчеркнем смысловое различие двух случаев употребления термина «смещение». Смещение disp,

содержащееся в команде, интерпретируется как знаковое целое, которое участвует в вычислении эффективного адреса EA. С другой стороны, из-за сегментной организации памяти весь эффективный адрес EA является смещением (offset) относительно базового адреса сегмента seg и интерпретируется как беззнаковое целое при вычислении физического адреса. В необходимых случаях во избежание ошибок будем называть offset смещением в сегменте.

Наиболее общий формат двухоперандной команды с непосредственным операндом приведен на рис. 3.17,б. В нем необходимость адресации второго операнда отсутствует, и поле reg используется для расширения кода операции. Отсутствует также бит направления d, так как результат операции можно поместить только на место первого операнда. Место этого бита занимает бит s, который является признаком использования одного байта для задания непосредственного операнда при работе со словами. Поля s и w интерпретируются следующим образом:

SW= { x0, один байт данных data L;
      01, два байта данных data H, data L;
      11, один байт данных, который автоматически расширяется со знаком до 16 бит.

Типичный формат одноопе рандной команды (рис. 3.17,*в*) содержит поля, назначение которых уже рассмотрено.

Следует отметить, что постбайтовая адресация является весьма универсальной и позволяет адресовать как общие регистры, так и ячейки памяти с указанием любого варианта вычисления эффективного адреса EA. Однако при адресации только регистров или аккумулятора постбайт оказывается излишним, если трехбитовое поле reg разместить в первом байте команды или использовать неявную адресацию. Эта возможность реализуется в специальных (укороченных) форматах, которые содержат минимально необходимое число байтов. Специальные форматы предусмотрены для команд, выполняющих следующие часто используемые операции: включение содержимого регистра в стек, извлечение из стека в регистр, передача непосредственных данных в регистр, инкремент и декремент содержимого регистра, обмен содержимым аккумулятора AX и регистра, пересылка между аккумулятором и ячейкой памяти с прямой адресацией, вычитание с участием содержимого аккумулятора и непосредственного операнда. В качестве примера на рис. 3.19 приведены стандартный и специальный форматы команды INC r. Программа-ассемблер выбирает более короткий формат команды автоматически.

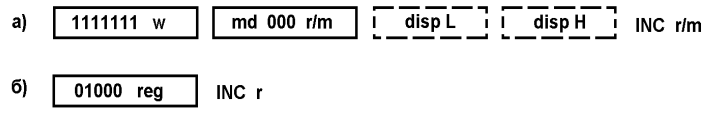


Рис. 3.19. Пример стандартного (а) и специального (б) форматов команды INC r

Всего можно выделить девять существенно различных форматов команд. В табл. 3.4 приведены форматы команд, не обязательные байты заключены в круглые скобки, а также указаны варианты форматов, различающихся назначением битов b_i первого байта B1 формата команды, в котором содержится код операции COP. Звездочкой помечены специальные форматы. (В данную таблицу не включены редко используемые двухбайтовые команды AAM и AAD, оба байта которых заняты кодом операции.)

Таблица 3.4

Формат	Вариант формата	Полный список команд
COP		CLC; CMC; STC; CLD; STD; CLI; STI; HLT; WAIT; NOP; RET; IRET; INTO; INT3; POSHF; POPF; LAHF; SAHF; XLAT; CBW; CWD; AAA; DAA; AAS; DAS; LOCK; REP; REPNZ
	bo=w	CMPS; LODS; MOVS; STOS; SCAB; IN ac.DX; OUT DX.ac
	b2-bo=reg	INC r; DEC r; PUSH r; POP r; XCHG AX,r
	b2-bo=COP b4-b3=sr	PUSH sr; POP sr
COP		LEA; LDS; LES
md reg r/m	bo=w	TEST r.r; TEST r,m; XCHG r,r; XCHG r,m
(disp L)	bo=w,	MOV r,r; MOV r.m; MOV m,r; ADD r.r;
(disp H)	b1=d	ADD r,m; ADD m.r; ADC r.r; ADC r.m; ADC m,r; SUB r.r; SUB r.m; SUB m.r; SBB r.r; SBB r.m; SBB m.r; CMP r.r;

```
                                CMP r,m ; CMP m.r; AND r.r; AND r.m;
                                AND m,r; OR r,r; OR r,m; OR m,r; XOR r,r;
                                XOR r,m ; XOR m.r;

COP                             PUSH r; PUSH m; POP r; POP m; JMP [r];
md COP r/m                      CALL [r]; JMP [m]; CALL [m]
(disp  L)    bo=w               INC m; DEC m; NEG; NOT; MUL; IMUL; DIV;
(disp  H)                       IDIV
             bo=w,              RCL; RCR; ROL; ROR; SAL; SAR; SHL; SHR
             b1=s

COP          bo=w               MOV m,d; MOV r,d; AND m,d; AND r,d;
md COP r/m                      OR m.d; OR r.d; XOR m,d; XOR r,d;
(disp  L)                       TEST m.d; TEST r.d; CMP m.d; CMP r.d
(diap  H)    bo=w               ADD m.d; ADD r.d; ADC m.d; ADC r.d;
data   L                        SUB m.d; SUB r.d; SBB m.d; SBB r.d
(data  H)

COP                             RET d
data   L     bo=w               ADD ac.d; ADC ac.d; SUB ac.d; SBB ac.d;
(data  H)                       CMP ac.d; AND ac.d; OR ac.d; XOR ac.d;
                                TEST ac.d
             b2-bo=reg          MOV r.d
             b3=w

COP                             JMP displ6; JMP disp8; Jcond disp8;
(disp  L)                       CALL displ6;
(disp  H)    bo=w               MOV ac,m; MOV m.ac (при diap L, disp H)

COP                             MOV sr.m; MOV sr,r; MOV m.sr;
md Osr r/m                      MOV r.sr;
(disp  L)
(disp  H)

COP                             IN addr8; OUT addr8
addr8

COP                             JMP addr; CALL addr (тип FAR)
offset L
offset H
seg L
seg H
```

Примечание: поле sr - определяет один из трех сегментных регистров: 11 - регистр DS, 10 - регистр SS, 00 - регистр ES; сокращение ac определяет регистр-аккумулятор: для байтовых операций - AL, для операций со словами - AX.

3.4.2. Способы адресации

Команды ЦП ВМ86 реализуют весьма разнообразные способы адресации, что упрощает организацию и использование сложных структур данных, а также расширяет возможности отдельных команд и повышает гибкость их применения.

Регистровая адресация. Операнд находится в одном из общих регистров МП или в одном из сегментных регистров. Регистр может быть определен в байте кода операции или в постбайте, в котором выделены 3-битовые поля reg и r/m (при md = 11). Команды, оперирующие содержимым регистров, являются наиболее короткими и выполняются за наименьшее время, так как не требуют вычисления ЕА и выполнения цикла шины для обращения к памяти. Для многих команд с регистровой адресацией предусмотрены специальные укороченные форматы.

Непосредственная адресация. Непосредственными операндами являются константы длиной 8 или 16 бит, которые размещаются в последних байтах команды (младший байт следует первым). Доступ к таким операндам в ВМ86 осуществляется очень быстро, поскольку во время выполнения команды они находятся во внутренней очереди команд. Имеются команды, позволяющие манипулировать непосредственными

операндами и содержимым общих регистров или ячеек памяти. Однако отсутствуют команды непосредственной загрузки сегментных регистров и включения константы в стек. Поэтому эти операции осуществляются с помощью промежуточной загрузки общего регистра или ячейки памяти.

Прямая адресация является простейшим способом адресации ячейки ЗУ, при котором эффективным адресом EA является содержимое байтов смещения disp команды. В командах преобразования данных этот способ реализуется при использовании постбайта с полями md = 00 и r/m = 110. Для пересылок между аккумулятором и памятью предусмотрен укороченный формат.

Разновидностью этого способа является длинная прямая адресация, при которой в формате команды содержатся четыре байта, указывающие базовый адрес сегмента и сегментное смещение offset. В этом случае обеспечивается доступ к ячейке с любым логическим адресом, т. е. к произвольной ячейке во всем пространстве адресов 1 Мбайт, но длинная прямая адресация используется только в командах межсегментных переходов и вызовов подпрограмм. Другая разновидность прямой адресации применяется для указания портов ввода — вывода в двухбайтовых командах IN и OUT, второй байт которых содержит адрес (номер) порта.

Косвенная регистровая адресация. В командах преобразования данных эффективный адрес EA равен содержимому одного из регистров SI, DI, BX и BP при соответствующем кодировании полей md и reg постбайта: md = 00; r/m = 100, 101, 111 и md = 01, r/m = 110; dispL = 0. В командах безусловного перехода и вызова подпрограммы с регистровой косвенной адресацией допускается указание любого 16-битового общего регистра (при md = 11; r/m = 000,..., 111).

Данный способ адресации позволяет вычислять адреса во время выполнения программ, что часто требуется, например, для обращения к различным элементам таблиц данных. При модификации содержимого регистра одна и та же команда оперирует различными ячейками памяти, что позволяет организовать вычислительные циклы. Изменение содержимого регистра обычно осуществляется с помощью команд инкрементирования и декрементирования, а также других арифметических команд и команды загрузки эффективного адреса LEA. Разновидностью этого способа является косвенная адресация портов ввода — вывода с помощью содержимого регистра DX в однобайтовых командах IN и OUT.

Базовая адресация. Эффективный адрес операнда EA вычисляется путем суммирования содержимого базовых регистров BX или BP и смещения disp (при md = 01, 10 и r/m = 111, 110). При использовании BX происходит обращение к операнду в текущем сегменте данных, а при использовании BP — в текущем сегменте стека. Смещения, содержащиеся в команде, могут иметь длину 8 или 16 бит и интерпретируются как знаковые целые, представленные в дополнительном коде.

Базовая адресация обычно используется для подступа к элементам структур данных, когда смещение (номер) элемента известно на стадии разработки программы (при ее ассемблировании), а базовый (начальный) адрес структуры должен вычисляться при выполнении программы. Модификация содержимого базового регистра позволяет обратиться к одноименному элементу различных структур данных.

Индексная адресация. Значение EA вычисляется как сумма смещения disp, находящегося в команде, и содержимого индексного регистра SI или DI (при md = 01, 10 и r/m = 100, 101). Данный способ обычно применяется для обращения к различным элементам одномерного массива (таблицы) данных, когда смещение определяет известный при ассемблировании начальный адрес массива, а индексный регистр, содержимое которого может модифицироваться при выполнении программы, определяет элемент массива. По существу индексная адресация в BM86 аналогична базовой.

Базовая индексная адресация. Эффективный адрес EA равен сумме содержимого базового регистра BX или BP, индексного регистра SI или DI и смещения disp, находящегося в команде (в частном случае смещение может отсутствовать) . Этот способ реализуется при следующем кодировании полей постбайта: md ≠ 11; r/m = 000, 001, 010, 011 и обеспечивает наибольшую гибкость адресации, так как два компонента адреса можно определить и варьировать при выполнении программы. Это удобно при обращении к элементам матриц, т. е. к двумерным массивам, представляемым в памяти как совокупность одномерных массивов.

Относительная адресация ЦП BM86 реализуется только по отношению к указателю команд IP, так что сегментное смещение вычисляется как сумма смещения disp, находящегося в команде, и текущего значения IP. При этом значение IP равно адресу байта, следующего за рассматриваемой командой, которая в это время выполняется микропроцессором. В BM86 относительная адресация не используется в командах, оперирующих данными, а применяется только в командах условных и безусловных переходов, вызовов подпрограмм и управления циклами. Смещение по отношению к содержимому IP не зависит от размещения программ в адресном пространстве памяти, что обеспечивает позиционную независимость команд. При автоматизированном ассемблировании программы указывается метка команды, которой передается управление, а необходимое смещение вычисляется программой-ассемблером.

Неявная адресация. Объект, содержимым которого манипулирует команда, указывается с помощью первого байта команды вместе с кодом операции без выделения специального поля для этой цели. Чаще всего этот специфический способ адресации встречается в однобайтовых командах, где адресуемым объектом являются аккумулятор, регистр флагов или отдельные флаги. В частности, в командах обработки

цепочек неявно используются индексные регистры, причем регистр SI адресует первый элемент цепочки-источника, а регистр DI — первый элемент цепочки-получателя.

3.4.3. Описание системы команд

Система команд ВМ86 содержит 91 мнемокод и позволяет совершать операции над байтами, двухбайтовыми словами, отдельными битами, а также цепочками байтов и слов. Имеется широкий набор арифметических команд, включающий умножение и деление, который ориентирован на обработку как беззнаковых, так и знаковых чисел. Весьма разнообразны команды пересылки данных, логических операций, переходов в программе и вызовов подпрограмм, а также управления микропроцессором. Число вариантов команд, т.е. число различных машинных кодов, превышает 3800 благодаря использованию восьми способов адресации в их различных модификациях. Тем самым обеспечивается гибкость применения большого числа команд, что позволяет выбрать наиболее рациональные способы адресации в конкретных случаях. Это особенно важно при обработке сложных структур данных.

По функциональному признаку система команд ВМ86 разбивается на шесть групп: пересылка данных, арифметические операции, логические операции и сдвиги, передача управления, обработка цепочек и управление микропроцессором. В наиболее компактной форме система команд представлена в табл. 2.5, в которой арифметические и логические команды объединены исходя из общности параметров этих команд и одинаковой последовательности действия МП при их выполнении. В табл. 3.5 введены следующие обозначения:

r — общий регистр; sr — сегментный регистр; m — адрес ячейки памяти, который указывается в мнемокоде в соответствии с используемым способом адресации; d — непосредственный операнд; ac — аккумулятор AX или AL; p — адрес 8-разрядного порта ввода — вывода; disp — смещение при адресации относительно IP; addr — указатель адреса при межсегментных переходах и вызовах; type — тип (вектор) прерывания; cond — условия в команде условных переходов.

В табл. 3.5 представлены все мнемонические обозначения команд и все допустимые варианты представления операндов. Для каждого варианта указано число байтов в формате n_B и число тактов синхронизации n_T, требуемое для выполнения команды. Значение E, равное числу тактов, которое требуется для вычисления эффективного адреса EA, следует брать из табл. 3.6 в соответствии с указанным в команде способом адресации. Если имеется два варианта команды, определяющие одно и то же действие, то в табл. 2.5 приводится более короткий вариант. Команды, которые имеют общие форматы и одинаковые варианты представления операндов при одинаковом времени выполнения (т.е. различаются только операцией, выполняемой над операндами), приводятся в одной строке.

Влияние команд на флаги иллюстрируют табл. 3.7 и 3.8, в которые включены только те команды, которые это влияние оказывают. Знак вопроса соответствует случаям, когда состояние флага после выполнения команды произвольно (например, зависит от конкретных значений операндов).

Полный набор команд, упорядоченный по мнемокоду, машинные коды команд и описание выполняемых операций приведены в табл. 3.9.

Для удобства пользования в таблице представлены все варианты мнемонических обозначений команд условных переходов и команд управления циклами. В дополнение к введенным ранее обозначениям в данной таблице используются следующие: disp 8/16—смещение в формате команды, состоящее из одного или двух байтов; d 8/16 — одно- или двухбайтовая константа; sbr — имя подпрограммы; diff— разница между адресом перехода и содержимым указателя команд IP (при адресации относительно IP); label — метка, к которой осуществляется переход. Для каждой цепочечной команды приведено только общее мнемоническое обозначение, которое требует дополнительного указания разрядности элементов цепочки, например MOVSB — пересылка байтов, MOVSW — пересылка слов.

Таблица 3.5.

Мнемокод	Операнд	n_B	n_T	Мнемокод	Операнд	n_B	n_T	Мнемокод	Операнд	n_B	n_T
MOV	r, r	2	2					LOOPZ	disp	2	18/6
	r, m	2-4	8+E	Corr[3]	-	1	4	LOOPNZ			
	m, r	2-4	9+E					JCXZ			
	ac, m	3	11	Flag[4]	-	1	2				
	m, ac	3	11					INT	-	1	52
	r, d	2-3	4	MUL[5]	r/m	2-4	133		type	2	51
	m, d	3-6	10+E								
	r, sr	2	2	IMUL[5]	r/m	2-4	154	INTO	-	1	53/4
	sr, r	2	2								
	m, sr	2-4	9+E	DIV[5]	r/m	2-4	162	IRET	-	1	24
	sr, m	2-4	8+E								
				IDIV[5]	r/m	2-4	184	REP	-	1	6
PUSH	r	1	10								

	m	2-4	16+E	AAM	-	2	83	MOVS	-	1	18
	sr	1	10	AAD	-	2	60	CMPS[8]	-	2	9+13
PUSHF	-	1	10	SBW		1	5	SCAS[8]	-	1	15
				CWD	-	1	5			2	9+15
POP	r	1	8								
	m	2-4	17+E	SHL	r	2	2	LODS[8]	-	1	12
	sr	1	8	SAL	m	2-4	15+E			2	9+13
				SHL	r, CL	2	8+4N				
POPF	-	1	8	SAR	m, CL	2-4	20+E +4N	STOS[8]	-	1	11
										2	9+10
XCHG	r, r	2	4	ROL							
				ROR				WAIT	-	1	3
XLAT	-	1	11	RCL							
				RCR				NOP	-	1	3
IN	ac, p	2	10								
	ac,[DX]	1	8	JMP[6]	disp	2-3	15	HLT	-	1	2
					r	2	11				
OUT	p, ac	2	10		m	2-4	18+E	ESC	r	2	2
	[DX],ac	1	8						m	2-4	8+E
				JMP[7]	addr	5	15				
LEA	r, m	3-4	2+E		m	2-4	24+E	LOCK	-	1	2
LDS	r, m	3-4	16+E								
LES	r, m	3-4	16+E	Jcond	disp	2	8/4				
INC	r	1	2	CALL[6]	disp	3	19				
DEC	m	2-4	15+E		r	2	16				
					m	2-4	21+E				
NEG	r	2	3								
NOT	m	2-4	16+E	CALL[7]	addr	5	28				
					m	4	37+E				
ADD	r, r	2	3								
ADC	r, m	2-4	9+E	RET[6]	-	1	8				
SUB	m, r[2]	2-4	16+E		d	3	12				
SBB	r, d	3-4	4								
CMP	ac, d	2-3	4	RET[7]	-	1	12				
AND[1]	m, d	3-6	17+E		d	3	18				
OR[1]											
XOR[1]				LOOP	disp	2	17/5				
TEST[2]											

[1] Логические команды (в отличие от приведенных в этой же строке арифметических команд) не имеют формата, в котором при работе со словами (т.е. при w = 1) указывается однобайтовый непосредственный операнд d.

[2] Команда TEST не имеет варианта m, r со временем выполнения 16+E, так как он был бы эквивалентен более короткому варианту r, m.

[3] Corr - команды коррекции при сложении и вычитании: DAA, DAS, AAA, AAS.

[4] Flag используется для обозначения команд управления флагами CLC, STC, CMC, CLI, STI, CLD, STD.

[5] Для команд умножения и деления указано время выполнения при работе с операндами максимально допустимой длины, размещенными в регистрах.

[6] Команда осуществляет внутрисегментный переход, вызов или возврат.

[7] Команда осуществляет межсегментный переход, вызов или возврат.

[8] Дано время выполнения одного цикла цепочечной команды, имеющей префикс повторения.

Данные табл. 3.9 позволяют осуществить переход от мнемокодов к машинным кодам команд. При необходимости выполнить обратный переход целесообразно воспользоваться системой команд, упорядоченной по машинному коду операции. Естественно, что наиболее удобным является автоматическое преобразование кодов с помощью соответствующих программ ассемблера и ре-ассемблера (дисассемблера).

Таблица 3.6

Адресация	Способ вычисления EA	Обозначение в мнемокоде	Число тактов E (EA)
Прямая	disp	disp	6
Косвенная	[BX], [BP], [SI], [DI]	[r]	5
Базовая или индексная	[BX, BP, SI, DI]+disp	[r]disp	9
Базовая индексная:			
без смещения	[BP+DI], [BX+SI]	[r][r]	7
	[BP+SI], [BX+DI]		8
со смещением	[BP+DI]+disp	[r]disp[r]	11
	[BX+SI]+dispsp		11
	[BP+SI]+disp		12
	[BX+DI]+disp		12

Таблица 3.7

Операции	Команды	Флаги состояний					
		OF	CF	AF	SF	ZF	PF
Сложение,	ADD, ADC, SUB, SBC	+	+	+	+	+	+
вычитание	CMP, NEG, CMPB, SCAS	+	+	+	+	+	+
	INC, DEC	+	-	+	+	+	+
Умножение	MUL, IMUL	+	+	?	?	?	?
Деление	DIV, ID IV	?	?	?	?	?	?
Десятичная	DAA, DAS	?	+	+	+	+	+
коррекция	AAA, AAS	?	+	+	?	?	?
	AAM, AAD	?	?	?	+	+	+
Логические	AND, OR, XOR, TEST	0	0	?	+	+	+
Сдвиги	SHL, SHR[1)]	+	+	?	+	+	+
	SHL, SHR[2)]	?	+	?	+	+	+
	SAR	0	+	?	+	+	+
	ROL,.ROR,.RCL,.RCR[1)]	+	+	-	-	-	-
	ROL, ROR, RCL, RCR[2)]	?	+	-	-	-	-
Восстановле	POPF, IRET	+	+	+	+	+	+
ние флагов	SAHF	-	+	+	+	+	+
Управление	STC	-	1	-	-	-	-
флагом	CLC	-	0	-	-	-	-
переноса	CMC	-	r	-	-	-	-

<u>Примечание.</u> "+" - результат операции влияет на флаг;
"-" - не влияет; 1 - устанавливает в "1"; 0 - устанавливает в "0";
r - инвертирует; ? – не определен.

[1)] Одиночный сдвиг.

[2)] Многоразрядный сдвиг.

Таблица 3.8

Операции	Команды	Флаги управления		
		DF	IF	TF
Восстановле	POPF IRET	+	+	+
ние флагов				
Прерывания	INT INTO	-	0	0
Управление	STD	1	-	-
флагами	CLD	0	-	-
	STI	-	1	-
	CLI	-	0	-

Примечание. "+" - результат операции влияет на флаг;
"-" - не влияет; 1 - устанавливает в "I";
0 - устанавливает в "0".

Таблица 3.9

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
1	AAA	00110111			ASCII - коррекция для сложения
2	AAD	11010101	00001010		ASCII - коррекция для деления
3	AAM	11010100	00001010		ASCII - коррекция для умножения
4	AAS	00111111			ASCII - коррекция для вычитания
5	ADC r,r/m	0001001W	md reg r/m	(disp8/16)	r←r+r/m+CF; сложение с переносом
	r/m,r	0001000W	md reg r/m	(disp8/16)	r/m←r+r/m+CF
	r/m,d	1000000W	md 010 r/m	(disp8/16)d8/16	r/m←r/m+d+CF
	r16/m16,d8	10000011	md 010 r/m	data L	r/m←r/m+d+CF
	ac,d	0001010W	data L	(data H)	ac←ac+d+CF
6	ADD r,r/m	0000001W	md reg r/m	(disp8/16)	r←r+r/m; сложение
	r/m,r	0000000W	md reg r/m	(disp8/16)	r/m←r+r/m
	r/m,d	1000000W	md 000 r/m	(disp8/16)d8/16	r/m←r/m+d
	r16/m16,d8	10000011	md 000 r/m	data L	r/m←r/m+d
	ac,d	0000010W	data L	(data H)	ac←ac+d
7	AND r,r/m	0010001W	md reg r/m	(disp8/16)	r←r∧r/m; конъюнкция, И
	r/m,r	0010000W	md reg r/m	(disp8/16)	r/m←r∧r/m
	r/m,d	1000000W	md 100 r/m	data L data H	r/m←r/m∧d
	ac,d	0010010W	data L	(data H)	ac←ac∧d
8	CALL				
	NEAR sbr	11101000	diff L	diff H	IP←IP+diff; вызов прямой
	NEAR r16/m16	11111111	md 010 r/m	(disp8/16)	IP←IP+r/m; вызов косвенный
	FAR sbr	10011010	IP-L	IP-H CS-L CS-H	IP←IP-L,IP-H; CS←CS-L,CS-H; вызов прямой
	FAR m32	11111111	md 011 r/m	(disp8/16)	IP←m16;CS←m16; вызов косвенный
9	CBW	10011000			Преобразование байта в слово
10	CLC	11111000			CF←0; сброс переноса
11	CLD	11111100			DF←0; сброс триггера направления
12	CLI	11111010			IF←0; запрет прерывания
13	CMC	11110101			CF←CF; инверсия переноса
14	CMP r,r/m	0011101W	md reg r/m	(disp8/16)	r-r/m; сравнение
	r/m,r	0011100W	md reg r/m	(disp8/16)	r/m-r
	r/m,d	1000000W	md 111 r/m	(disp8/16)d8/16	r/m-d
	r16/m16,d8	10000011	md 111 r/m	data L	r/m-d
	ac,d	0011110W	data L	(data H)	ac-d
15	CMPS	1010011W			Сравнение элементов цепочек
16	CWD	10011001			Преобразование слова в двойное слово
17	DAA	00100111			Десятичная коррекция для сложения
18	DAS	00101111			Десятичная коррекция для вычитания
19	DEC r/m	1111111W	md 001 r/m	(disp8/16)	r/m←r/m-1; декремент
	r	01001reg			
20	DIV r/m	1111011W	md 110 r/m	(disp8/16)	Деление чисел без знака
21	ESC OPCODE,r/m	11011XXX	md YYY r/m	(disp8/16)	Переключение на сопроцессор
22	HLT	11110100			Останов
23	IDIV r/m	1111011W	md 111 r/m	(disp8/16)	Деление чисел со знаком
24	IMUL r/m	1111011W	md 101 r/m	(disp8/16)	Умножение чисел со знаком
25	IN ac,port	1110010W	port8		Ввод из фиксированного порта
	ac,DX	1110110W			Ввод из порта с косвенной адресацией
26	INC r/m	1111111W	md 000 r/m	(disp8/16)	r/m←r/m+1; инкремент
	r	01000reg			r←r+1;

Таблица 3.9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
27	INT type	11001101			Прерывание заданного типа
	INTO	11001110			Прерывание по переполнению
	INT3	11001100			Прерывание типа 3
28	IRET	11001111			Возврат из прерывания
29	JA label	01110111	diff L		IP←IP+diff L; перейти, если выше
30	JAE label	01110011	diff L		IP←IP+diff L; если выше или равно
31	JB label	01110010	diff L		IP←IP+diff L; если ниже
32	JBE label	01110110	diff L		IP←IP+diff L; если ниже или равно
33	JC label	01110010	diff L		IP←IP+diff L; если есть перенос
34	JCXZ label	11100011	diff L		IP←IP+diff L; если CX равен нулю
35	JE label	01110100	diff L		IP←IP+diff L; если равно
36	JG label	01111111	diff L		IP←IP+diff L; если больше
37	JGE label	01111101	diff L		IP←IP+diff L; если больше или равно
38	JL label	01111100	diff L		IP←IP+diff L; если меньше
39	JLE label	01111110	diff L		IP←IP+diff L; если меньше или равно
40	JMP				
	SHORT label	11101011	diff L		IP←IP+diff L; прямой короткий переход
	NEAR label	11101001	diff L	diff H	IP←IP+diff L; прямой переход
	NEAR r16/m16	11111111	md 100 r/m	(disp8/16)	IP←IP+r/m; косвенный переход
	FAR label	11101010	IP-L	IP-H CS-L CS-H	IP←IP-L,IP-H; CS←CS-L,CS-H; прямой переход
	FAR m32	11111111	md 101 r/m	(disp8/16)	IP←m16;CS←m16; косвенный переход
41	JNA label	01110110	diff L		IP←IP+diff L; если не выше
42	JNAE label	01110011	diff L		IP←IP+diff L; если не выше или равно
43	JNB label	01110111	diff L		IP←IP+diff L; если не ниже
44	JNC label	01110011	diff L		IP←IP+diff L; если нет переноса
45	JNE label	01110101	diff L		IP←IP+diff L; если не равно
46	JNG label	01111110	diff L		IP←IP+diff L; если не больше
47	JNGE label	01111100	diff L		IP←IP+diff L; если не больше или равно
48	JNL label	01111101	diff L		IP←IP+diff L; если не меньше
49	JNLE label	01111111	diff L		IP←IP+diff L; если не меньше или равно
50	JNO label	01110001	diff L		IP←IP+diff L; если нет переполнения
51	JNP label	01111011	diff L		IP←IP+diff L; если нет паритета
52	JNS label	01111001	diff L		IP←IP+diff L; если результат положительный
53	JNZ label	01110101	diff L		IP←IP+diff L; если не нуль
54	JO label	01110000	diff L		IP←IP+diff L; если есть переполнение
55	JP label	01111010	diff L		IP←IP+diff L; если есть паритет

Таблица 3.9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
56	JPE label	01111010	diff L		IP←IP+diff L; если паритет четный
57	JPO label	01111011	diff L		IP←IP+diff L; если паритет нечетный
58	JS label	01111000	diff L		IP←IP+diff L; если результат отрицательный
59	JZ label	01110100	diff L		IP←IP+diff L; если нуль
60	LAHF	10011111			AH←FL; пересылка младшего байта F в AH
61	LDS r16,m32	11000101	md reg r/m	(disp8/16)	r,DS←m32; загрузка указателя адреса
62	LEA r16,m	10001101	md reg r/m	(disp8/16)	r←EA; загрузка эффективного адреса
63	LES r16,m32	11000100	md reg r/m	(disp8/16)	r,ES←m32; загрузка указателя адреса
64	LODS	1010110W			Загрузка элемента цепочки
65	LOOP label	11100010	diff L		IP←IP+diff L; зациклить
66	LOOPE/LOOPZ label	11100001	diff L		IP←IP+diff L; зациклить, если равно
67	LOOPNE/LOOPNZ label	11100000	diff L		IP←IP+diff L; зациклить, если не равно
68	MOV r,r/m	1000101W	md reg r/m	(disp8/16)	r←r/m; пересылка
	r/m,r	1000100W	md reg r/m	(disp8/16)	r/m←r;
	r/m,d	1100011W	md 000 r/m	(disp8/16)d8/16	r/m←d;
	r,d	1011Wreg	data L	(data H)	r←d
	ac,m	1010000W	disp L	disp H	ac←m; при прямой адресации
	m,ac	1010001W	disp L	disp H	m←ac; при прямой адресации
	sr,r/m	10001110	md 0sr r/m	(disp8/16)	sr←r/m; запись в сегментный регистр
	r/m,sr	10001100	md 0sr r/m	(disp8/16)	r/m←sr; чтение из сегментного регистра
69	MOVS	1010010W			Пересылка элемента цепочки
70	MUL r/m	1111011W	md 100 r/m	(disp8/16)	Умножение чисел без знака
71	NEG r/m	1111011W	md 011 r/m	(disp8/16)	Смена знака числа
72	NOP	10010000			Пустая операция
73	OR r,r/m	0000101W	md reg r/m	(disp8/16)	r←rvr/m; дизъюнкция, ИЛИ
	r/m,r	0000100W	md reg r/m	(disp8/16)	r/m←r/mvr
	r/m,d	1000000W	md 001 r/m	(disp8/16)d8/16	r/m←r/mvd
	ac,d	0000110W	data L	(data H)	ac←acvd
74	OUT ac,port	1110011W	port8		Вывод в фиксированный порт
	ac,DX	1110111W			Вывод в порт при косвенной адресации
75	POP r/m	10001111	md 000 r/m	(disp8/16)	Чтение из стека
	r	01011reg			
	sr	000sr111			
76	POPF	10011101			Загрузка регистра флагов из стека
77	PUSH r/m	11111111	md 110 r/m	(disp8/16)	Запись в стек
	r	01010reg			
	sr	000sr110			
78	PUSHF	10011100			Запись регистра флагов в стек
79	RCL r/m,1	1101000W	md 010 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 010 r/m	(disp8/16)	влево через CF
80	RCR r/m,1	1101000W	md 011 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 011 r/m	(disp8/16)	вправо через CF

Таблица 3.9 (продолжение)

№ п/п	Команда	Байты кода команды			Символическое описание и/или содержание операции
		B1	B2	B3-B6	
81	RET (NEAR)	11000011			Возврат из подпрограммы
	d(NEAR)	11000010	data L	data H	Возврат с прибавлением константы к SP
	(FAR)	11001011			Возврат из подпрограммы
	d(FAR)	11001010	data L	data H	Возврат с прибавлением константы к SP
82	ROL r/m,1	1101000W	md 000 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 000 r/m	(disp8/16)	влево
83	ROR r/m,1	1101000W	md 001 r/m	(disp8/16)	Циклический сдвиг
	r/m,CL	1101001W	md 001 r/m	(disp8/16)	вправо
84	SAHF	10011110			Пересылка AH в регистр F
85	SAL r/m,1	1101000W	md 100 r/m	(disp8/16)	Арифметический сдвиг
	r/m,CL	1101001W	md 100 r/m	(disp8/16)	влево
86	SAR r/m,1	1101000W	md 111 r/m	(disp8/16)	Арифметический сдвиг
	r/m,CL	1101001W	md 111 r/m	(disp8/16)	вправо
87	SBB r,r/m	0001101W	md reg r/m	(disp8/16)	r←r-r/m-CF; вычитание с заемом
	r/m,r	0001100W	md reg r/m	(disp8/16)	r/m←r-r/m-CF
	r/m,d	1000000W	md 011 r/m	(disp8/16)d8/16	r/m←r/m-d-CF
	r16/m16,d8	10000011	md 011 r/m	data L	r/m←r/m-d-CF
	ac,d	0001110W	data L	(data H)	ac←ac-d-CF
88	SCAS	1010111W			Сканировать элемент цепочки
89	SHL r/m,1	1101000W	md 100 r/m	(disp8/16)	Логический сдвиг
	r/m,CL	1101001W	md 100 r/m	(disp8/16)	влево
90	SHR r/m,1	1101000W	md 101 r/m	(disp8/16)	Логический сдвиг
	r/m,CL	1101001W	md 101 r/m	(disp8/16)	вправо
91	STC	11111001			CF←1; установка флага переноса
92	STD	11111101			DF←1; установка триггера направления
93	STI	11111011			IF←1; разрешение прерывания
94	STOS	1010101W			[DI]←ac; запоминание ac в цепочке
95	SUB r,r/m	0010101W	md reg r/m	(disp8/16)	r←r-r/m; вычитание
	r/m,r	0010100W	md reg r/m	(disp8/16)	r/m←r-r/m
	r/m,d	1000000W	md 101 r/m	(disp8/16)d8/16	r/m←r/m-d
	r16/m16,d8	10000011	md 101 r/m	data L	r/m←r/m-d
	ac,d	0010110W	data L	(data H)	ac←ac-d;
96	TEST r,r/m	1000010W	md reg r/m	(disp8/16)	r∧r/m; проверка, неразрушающее И
	m,r	1000010W	md reg r/m	(disp8/16)	r∧r/m
	r/m,d	1111011W	md 000 r/m	(disp8/16)d8/16	r/m d
	ac,d	1010100W	data L	(data H)	ac∧d
97	WAIT	00111011			Ожидание
98	XCHG AX,r	10010reg			AX↔r; обмен
	r,AX	10010reg			r↔AX
	r,r/m	1000011W	md reg r/m	(disp8/16)	r↔r/m
	m,r	1000011W	md reg r/m	(disp8/16)	m↔r
99	XLAT	11010111			Преобразование байта из AL
100	XOR r,r/m	0011001W	md reg r/m	(disp8/16)	r←r⊕r/m; суммирование по модулю 2 (исключающее ИЛИ)
	r/m,r	0011000W	md reg r/m	(disp8/16)	r/m←r⊕r/m;
	r/m,d	1000000W	md 110 r/m	(disp8/16)d8/16	r/m←r/m⊕d
	ac,d	0011010W	data L	(data H)	ac←ac⊕d
	Префиксы:				
	LOCK	11110000			Префикс блокировки шины
	REP/REPE/REPZ	11110011			Повторять цепочечную операцию
	REPNE/REPNZ	11110010			То же
	SEG	001sr110			Префикс замены сегмента

3.4.4. Особенности выполнения отдельных команд

Приведем здесь описание лишь некоторых команд, выбранных по принципу частоты их использования или трудности освоения. Описание остальных команд можно найти в [11 - 13].

Команды пересылки данных составляют четыре подгруппы: общие, обращения к стеку, ввода — вывода и пересылки цепочек (последние рассмотрены вместе с другими цепочечными командами). Эти команды, за исключением POPF и SAHF, не влияют на флаги.

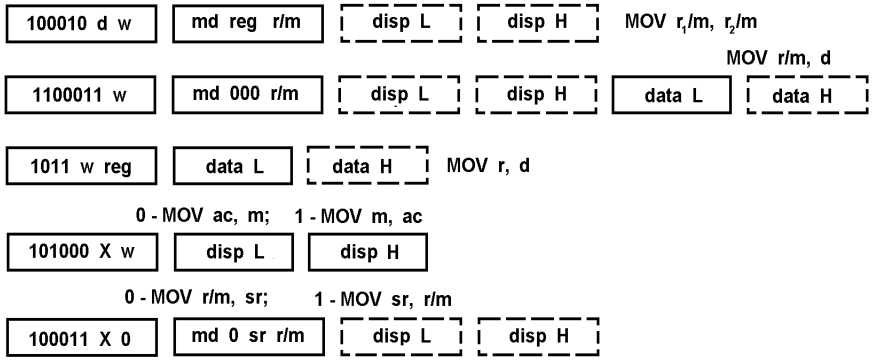


Рис. 3.20. Форматы команды MOV

Команда MOV (рис. 3.20) осуществляет пересылку содержимого источника src в получатель dst и имеет обобщенное представление:

MOV dst, src; dst←src

Команда MOV r_1/m, r_2/m содержит постбайт и обеспечивает пересылки регистр — регистр/память и память — регистр при использовании любого общего регистра и любого способа адресации памяти. Бит w определяет передачу байта или слова, а бит d — направление передачи. Команда MOV r/m, d позволяет передать непосредственные данные в общий регистр или ячейку памяти. Команда MOV r, d представляет более короткий вариант (специальный формат) предыдущей команды и осуществляет загрузку констант в общие регистры.

Команды MOV ac, m и MOV m, ac предназначены для загрузки и запоминания содержимого аккумуляторов AL и AX при использовании прямой адресации. Обращение производится к текущему сегменту данных, и адрес, указанный в команде, представляет смещение в этом сегменте. Если пересылаются два байта, то младший располагается по указанному адресу, а старший — по следующему адресу.

Команды MOV sr, r/m и MOV r/m, sr осуществляют пересылки между сегментным регистром и регистром или памятью. При этом передаются только слова, а ячейка памяти может быть определена с помощью любого допустимого способа адресации. Следует учитывать, что в команде MOV sr, r/m нельзя указывать сегментный регистр кода CS, так как при этом результат операции неопределен (выполнение этой команды было бы равносильно специальному безусловному переходу в программе).

Так как не существует команды непосредственной загрузки сегментных регистров, то команда MOV sr, r/m используется для инициализации регистров SS, DS и ES, т. е. для определения соответствующих сегментов памяти. Если, например, в регистр DS необходимо загрузить число 8000, то потребуется две команды:

MOV AX, 8000H; Инициализация регистра DS на 8000
MOV DS, AX;

При этом в качестве промежуточного регистра обычно используется AX, так как команда MOV ac, d короче более общей команды MOV m/r, d.

Возможна также инициализация сегментных регистров из программной памяти при использовании префикса замены сегмента для замены DS на CS при вычислении адреса EA в следующих командах с прямой адресацией:

MOV DS, CS: ADS; Инициализация DS
MOV ES, CS: AES; Инициализация ES
MOV SS, CS: ASS; Инициализация SS

Напомним, что МП BM86 обеспечивает защиту процесса инициализации регистров SS и SP**,** состоящего из двух команд:

MOV SS, CS: ASS
MOV SP, CS: ASP

от прерываний, чтобы гарантировать правильную работу стека. (Операнды ADS, AES, ASS, ASP обозначают адреса ячеек памяти, в которых хранятся базовые адреса соответствующих сегментов.)

Команда XCHG (рис. 3.21) осуществляет обмен данными между источником и получателем:

XCHG dst, src; dst↔src
и имеет два формата. Общий формат позволяет произвести обмен содержимым любой пары общих регистров, а также обмен между общим регистром и ячейкой памяти при любом допустимом способе адресации. Укороченный формат осуществляет обмен любого общего регистра и аккумулятора AX. Команда XCHG AX, AX, 16-ричный код которой равен 90, используется как команда пустой операции NOP, обеспечивающая задержку 3T.

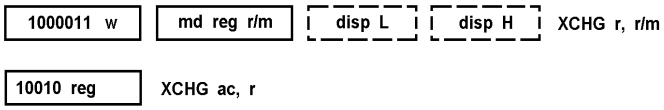
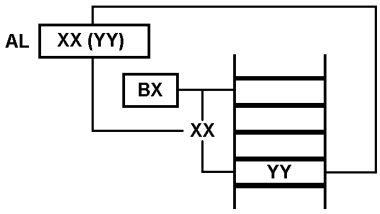


Рис. 3.21. Команда XCHG

XLAT — однобайтная команда с кодом операции D7 предназначена для быстрого преобразования кодов и заменяет содержимое AL на байт из 256-байтовой таблицы, начальный (базовый) адрес которой содержится в регистре BX (рис. 3.22). Другими словами, содержимое AL используется как индекс таблицы, находящейся в сегменте данных и адресуемой регистром BX. При выполнении этой команды к содержимому BX прибавляется содержимое AL, а полученный результат используется как смещение относительно DS. Адресуемый таким образом байт из памяти пересылается в AL.



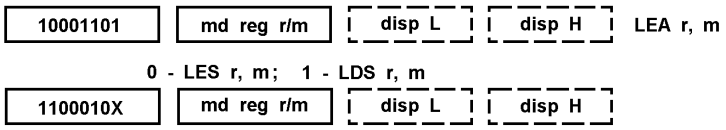


Рис. 3.22. Действие команды XLAT **Рис. 3.23.** Команды LEA, LDS, LES

Команды LEA, LDS, LES (рис. 3.23) отличаются от других команд пересылки тем, что при их выполнении в адресуемый регистр (регистры) передаются не собственно данные из памяти, а их адреса. Основное назначение этих команд — инициализация регистров перед выполнением цепочечных команд или перед вызовом подпрограммы.

Команда LEA r, m обеспечивает вычисление эффективного адреса EA ячейки памяти в соответствии с указанным способом адресации и загрузку EA (а не содержимого адресуемой ячейки памяти!) в указанный общий регистр. Такая операция может потребоваться, например, для загрузки начального адреса таблицы в регистр BX перед выполнением команды XLAT.

Команды LDS *r/m* и LES r, m загружают адресную информацию из памяти в адресуемый общий регистр и в соответствующий сегментный регистр. Сначала вычисляется адрес EA памяти, который, как обычно, суммируется с содержимым регистра DS, затем слово из памяти по вычисленному адресу загружается в общий регистр, а следующее слово — в регистр DS (команда LDS) или ES (команда LES). При этом кодирование поля md=11 (регистровая адресация) не должно использоваться, так как действия команд в этом случае не определены.

Команды LDS и LES упрощают коммутацию сегментов данных, поскольку одновременно загружают базовый или индексный регистр и сегментный регистр. Если эти команды подготавливают обращение к цепочке, то в команде LDS указывается регистр SI (цепочка-источник расположена в сегменте данных, определяемым содержимым регистра DS, и адресуется регистром SI), а в команде LES—регистр DI (цепочка-получатель обязательно находится в дополнительном сегменте, определяемым содержимым регистра ES, и адресуется регистром DI).

Арифметические команды выполняются над целыми числами четырех типов: беззнаковыми и знаковыми двоичными, упакованными и неупакованными десятичными.

Беззнаковые двоичные числа могут иметь длину 8 или 16 бит, каждый из которых является значащим, т. е. учитывается при определении значения числа. Это обеспечивает диапазон представления чисел 0 — 255 и 0 — 65 535 соответственно. Для таких чисел имеются команды сложения, вычитания, умножения и деления.

Знаковые двоичные числа также могут содержать 8 или 16 бит, но значащими являются все биты, кроме старшего, который определяет знак числа: 0 — положительное число, 1 — отрицательное число. Соответственно диапазоны значений чисел: от -128 до +128 и от -32 768 до +32 767. Число нуль содержит нули во всех разрядах и считается положительным и четным. Число, имеющее нули во всех разрядах, кроме знакового, равно -2^7 для 8-битовых чисел и -2^{15} для 16-битовых.

Числа представляются в стандартном дополнительном коде, благодаря применению которого сложение и вычитание как знаковых, так и беззнаковых чисел выполняется с помощью одних и тех же команд (рис. 3.24). Для умножения и деления знаковых чисел предусмотрены специальные команды.

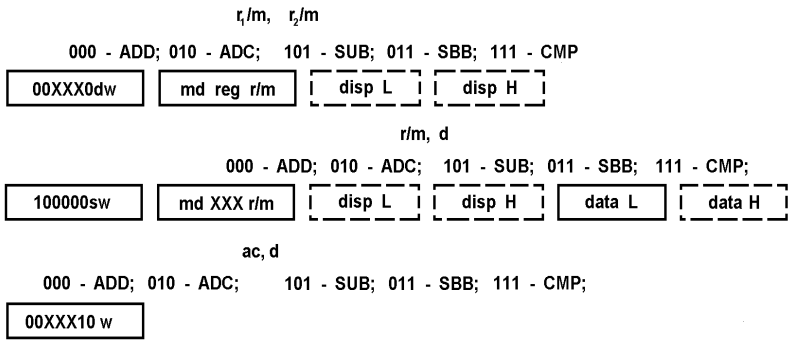


Рис. 3.24. Команды сложения и вычитания

Команда NEG изменяет знак числа, находящегося в общем регистре или ячейке памяти, т. е. формирует его дополнительный код:

NEG dst; dst←0 - dst

Например, если однобайтовый операнд равен -1 (11111111), то команда NEG изменит его на +1 (00000001). Если операнд равен нулю, его значение не изменяется. Попытка изменить знак числа -128 (10000000) или -2^{16} не модифицирует операнд, но устанавливает флаг переполнения OF. Отметим, что команда NEG отсутствует в ВМ80 (i8080), поскольку в нем не предусмотрено специальных средств по обработке знаковых чисел.

Команды умножения и деления (рис. 3.25). В МП ВМ86 (i8086) имеется по две команды умножения **(MUL и IMUL)** и деления **(DIV и IDIV),** выполняющие операции с беззнаковыми и знаковыми числами (в дополнительном коде) соответственно. Работа с десятичными числами требует использования специальных команд коррекции AAM и AAD.

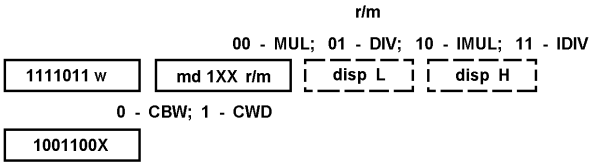


Рис. 3.25. Команды умножения и деления

Команды умножения

MUL (IMUL) src; ext:ac←ac×src

выполняют умножение адресуемого операнда (общего регистра или ячейку памяти) и содержимого аккумулятора ac. При работе с байтами функции аккумулятора ac выполняет регистр AL, а функции его расширения (ext) — регистр AH, так что 16-битовое произведение формируется в регистре AX. Если перемножаются слова, то множимое располагается в регистре AX, функции расширения которого выполняет регистр DX, так что 32-битовый результат образуется в регистрах DX и AX.

Команды умножения беззнаковых и знаковых чисел осуществляют практически одни и те же действия и несколько различаются только влиянием на флаги OF и CF (значения остальных арифметических флагов после выполнения этих команд не определены, т. е. эти флаги могут принимать произвольные состояния). При выполнении команды MUL флаги OF и CF устанавливаются в единицу, если старшая половина произведения, находящаяся в регистрах AH или DX, отличается от нулевой, т. е. разрядность результата действительно превышает разрядность операндов. В противном случае эти флаги принимают нулевые значения. При выполнении команды IMUL флаги OF и CF устанавливаются в единицу, если старшая половина разрядов произведения не является расширением знака младшей половины, т. е. не содержит 00 (0000) или FF (FFFF) при умножении байтов (слов). Это означает, что в старшей половине находятся значащие цифры результата. В противном случае OF=CF=0 и старшую половину разрядов произведения можно не сохранять. Отсюда следует, что флаги OF и CF несут однотипную информацию о результатах выполнения команд MUL и IMUL.

Команды деления

DIV (IDIV) src; ac←quot (ext:ac/src)
ext←rem (ext:ac/src)

производят деление содержимого аккумулятора и его расширения (AH:AL для 8-битового и DX:AX для 16-битового делителя) на содержимое регистра или ячейки памяти src. Частное quot формируется в регистре AL или AX, а остаток rem — в регистре AH или DX. Дробное частное округляется до целого путем отбрасывания дробной части результата. Состояния всех флагов не определены.

Если частное выходит за диапазоны представления чисел в байте (слове) или делитель является нулем, то автоматически генерируется прерывание по ошибке деления (тип 0), а частное и остаток не определены. В результате этого МП переходит к подпрограмме обработки прерывания, полный адрес которой CS:IP берется

из ячеек 0000 и 0002.

В ряде случаев, например для предотвращения прерывания, может появиться необходимость до выполнения операции деления проверить возможность возникновения прерывания типа 0. Такая проверка осуществляется с помощью команд:

```
CMP ext, src;        сравнение и переход
JNB OVERFLOW;        по переполнению
```

осуществляющих сравнение старшей половины разрядов делимого с делителем и передающих управление по метке OVERFLOW, если левый операнд в команде CMP больше или равен правому (что является условием возникновения переполнения при делении).

Частное и остаток после выполнения команды IDIV всегда имеют одинаковые знаки. Например, при делении числа -47 на +3 из двух возможных результатов: -15 с остатком -2 и -16 с остатком +1 будет сформирован первый результат.

Однобайтовые команды преобразования разрядности операнда CBW и CWD примыкают к командам деления и осуществляют расширение со знаком операнда, который будет использоваться в качестве делимого. Обе команды не влияют на флаги и не изменяют значения операнда. Команда CBW (код операции 98) реализует преобразование байта в слово путем расширения (копирования) знака содержимого регистра AL в регистр AH. Команда CWD (код операции 99) осуществляет аналогичное преобразование слова в двойное слово путем передачи знака содержимого AX во все биты регистра DX.

Команды CBW и CWD удобно использовать для превращения делимого одинарной длины в делимое двойной длины, что может потребоваться для корректного выполнения команды IDIV. Выполнение команды CBW перед командой IDIV позволяет осуществлять деление 8-битовых чисел, а выполнение команды CWD—деление 16-битовых чисел.

Алгоритмы умножения и деления в МП ВМ86 реализованы не аппаратно, а в виде микропрограмм. Поэтому длительность команд MUL, IMUL, DIV, IDIV включает большое число тактов, причем длительность каждой команды зависит не только от разрядности операндов и расположения операнда src (в регистре или в памяти), но и от конкретных значений операндов в пределах, указанных в табл. 3.10 диапазонов длительности команд. В первом столбце в скобках указана длина сомножителей команд умножения и длина делителя (частного) для команд деления.

Таблица 3.10

Тип операции (длина операнда)		MUL	IMUL	DIV	IDIV
r-r	(байт)	70-77	80-98	80-90	101-112
r-r	(слово)	118-133	128-154	144-162	165-184
r-m	(байт)	(76-83)+EA	(84-104)+EA	(134-160)+EA	(86-96)+EA	(107-118)+EA
r-m	(слово)	(124-139)+EA		(150-168)+EA	(171-190)+EA

К логическим командам относятся следующие (рис. 3. 26): AND (конъюнкция, И), **OR** (дизъюнкция, ИЛИ), **XOR** (исключающее ИЛИ, сумма по модулю два), **TEST** (неразрушающая проверка, которая выполняет конъюнкцию операндов без изменения их значений, но с влиянием на флаги) и **NOT** (инверсия, НЕ). Все логические операции выполняются поразрядно, т. е. каждый бит операндов обрабатывается независимо от других. Для этого в АЛУ микропроцессора включены наборы не связанных друг с другом логических элементов, содержащие по 16 двухвходовых элементов, И, ИЛИ и сумматоров по модулю два, а также набор из 16 инверторов.

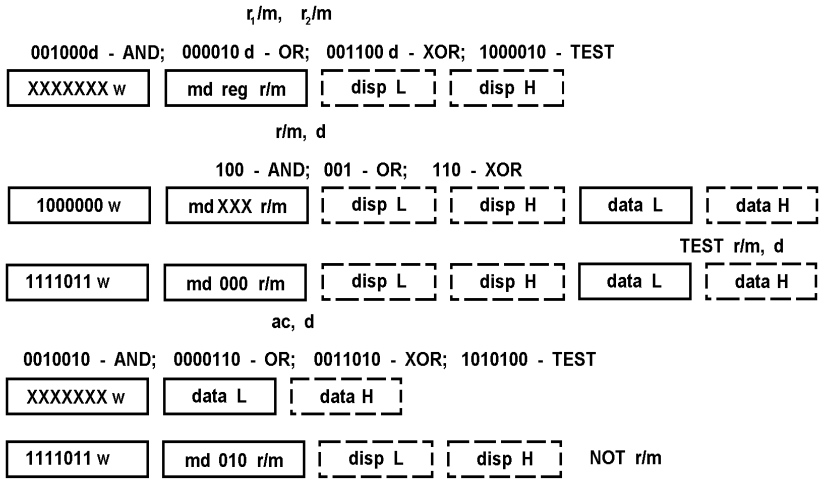


Рис. 3.26. Команды логических операций

Команды передачи управления используются в разветвляющихся и циклических программах, а также при вызове подпрограмм и возврате из них.

Сегментная организация программной памяти определяет два основных типа команд передачи управления: внутрисегментные **NEAR** (близкие) и межсегментные **FAR** (далекие). При выполнении команд типа NEAR модифицируется только регистр IP, и адрес переходов представляется одним словом или даже байтом, если используется короткий вариант перехода с ограниченным диапазоном адресов. При выполнении команд типа FAR изменяется содержимое регистров IP и CS и адрес перехода представляется двумя словами (сегмент:смещение), что позволяет перейти в любую точку адресного пространства памяти.

В рассматриваемую группу команд входят команды безусловных переходов, вызовов, возвратов, условных переходов, управления циклами и прерываний.

Команды безусловных переходов JMP (рис. 3.27) производят модификацию регистра IP или регистров IP и CS без сохранения прежних, значений этих регистров. Имеется три формата (рис. 3.28) команды JMP типа NEAR, осуществляющих переход в пределах текущего кодового сегмента (внутрисегментный, близкий переход), и два формата команды JMP типа FAR, осуществляющих переход в любую точку адресного пространства (межсегментный, дальний переход)

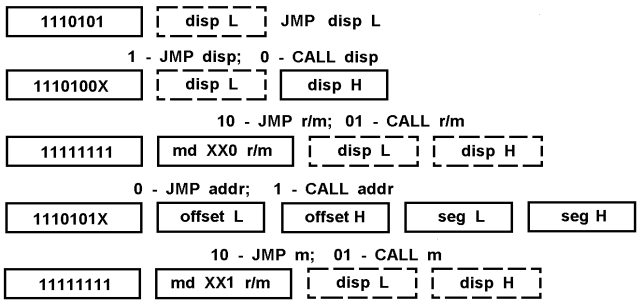


Рис. 3.27. Форматы команд JMP и CALL

Двухбайтовая команда JMP disp L во втором байте содержит смещение которое интерпретируется как знаковое целое. Это смещение добавляется (с предварительным расширением знака до 16 бит) к содержимому IP, которое соответствует адресу команды, находящейся после данной команды JMP. Диапазон значений disp L составляет от -128 до +127, причем при положительном смещении осуществляется переход вперед, а при отрицательном — переход назад. Данная команда реализует так называемый короткий переход который в мнемоническом обозначении отмечается указателем SHORT (напри мер, JMP SHORT COUNT — короткий переход к метке COUNT). Она находит широкое применение, поскольку большинство переходов в прикладных программах осуществляется на небольшие расстояния.

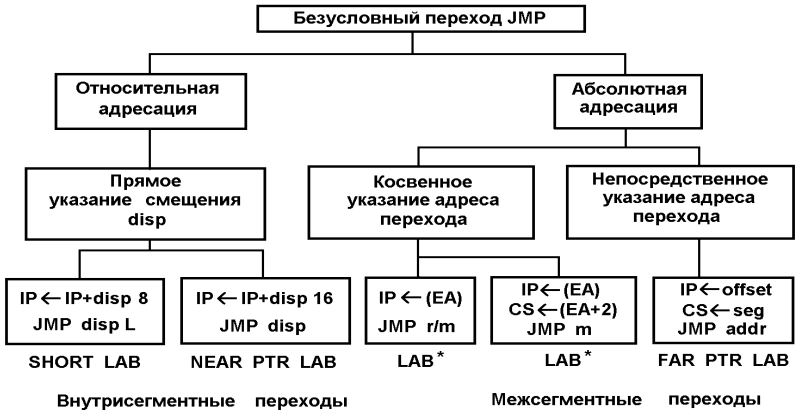


Рис. 3.28. Варианты формирования адресов переходов командой JMP

Трехбайтовая команда JMP disp производит такое же действие, как и предыдущая команда, но содержит 16-битовое смещение disp (ассемблерный указатель NEAR PTR). Это смещение также интерпретируется как знаковое целое, принимающее значения от -32 768 до +32 767, что обеспечивает переход в любую точку текущего кодового сегмента. Напомним, что благодаря игнорированию переноса из бита 16, возникающего при сложении IP и disp, текущий сегмент рассматривается как кольцо. Поэтому диапазон переходов составляет от -IP до 2^{16}-IP, а не от -2^{15} до 2^{15}-1. Если, например, IP = 8000 и смещение равно 7FFF или меньше, оно вызовет переход вперед, а смещение, равное 8000 или больше, вызовет переход назад.

Первые два рассмотренных формата команды JMP реализуют относительный способ адресации, который обеспечивает позиционную независимость программ. Следующие три формата определяют абсолютные адреса переходов и загружают регистр IP (а при межсегментных переходах и регистр CS) новыми значениями, не зависящими от прежних.

Команда **JMP** r/m осуществляет косвенный внутрисегментный переход (ассемблерный указатель WORD PTR), при котором в регистр IP загружается содержимое регистра или ячейки памяти в соответствии с постбайтовым режимом адресации. Необязательные байты disp L, disp H в команде стандартно относятся к

режиму адресации памяти.

Команда **JMP m** реализует косвенный межсегментный переход (ассемблерный указатель DWORD PTR) через содержимое двух ячеек памяти: слово из адресуемой с помощью постбайта ячейки загружается в IP, а слово из следующей ячейки — в регистр CS. Если в постбайте указывается регистр (rnd=11), то операция не определена.

Команда **JMP addr** реализует прямой межсегментный переход (ассемблерный указатель FAR PTR) и содержит 4 байта адреса перехода: два байта сегментного смещения offset загружаются в регистр IP, а два байта sr — в регистр CS.

Команды вызова подпрограммы CALL (см. рис. 3.27) имеют такие же форматы, как и команды JMP (кроме команды короткого перехода) и выполняются аналогичным образом, за исключением того, что автоматически запоминается адрес возврата (т. е. адрес команды, следующей за командой CALL). С этой целью при внутрисегментных вызовах в стеке запоминается содержимое IP, а при межсегментных вызовах— сначала содержимое IP, а затем CS. Напомним, что включение в стек каждого слова сопровождается уменьшением содержимого SP на два и что SP адресует последнюю заполненную 16-битовую ячейку стека. Использование стека для временного хранения адресов возврата позволяет легко реализовать правильный порядок возвратов из вложенных подпрограмм.

Перечислим действия, выполняемые 5-байтовой командой прямого межсегментного вызова CALL addr:

содержимое регистра SP уменьшается на два;

в адресуемую регистрами SP и SS ячейку памяти пересылается содержимое CS;

содержимое SP уменьшается на два;

в адресуемую регистрами SP и SS ячейку памяти засылается содержимое регистра IP;

в IP загружается смещение offset;

в CS загружается сегментный адрес seg.

В микропроцессоре BM86 отсутствуют команды условных вызовов подпрограмм (в отличие от BM80) и поэтому при необходимости условный вызов реализуется двумя командами. Например, если требуется вызвать подпрограмму SUBR при ненулевом результате операции, т. е. реализовать отсутствующую команду CNZ, то выполняется команда условного перехода, проверяющая противоположное условие, и осуществляется «перескок» через команду вызова CALL:

```
        JZ      SKIP
        CALL    SUBR
SKIP: ...
```

В зависимости от того, как определен тип подпрограммы SUBR, ассемблер сформирует необходимую команду внутрисегментного вызова.

Команды возвратов (из подпрограмм) RET (рис. 3.29). Каждая подпрограмма должна содержать хотя бы одну команду возврата RET, которая возвращает управление программе, осуществившей вызов. Такая передача управления осуществляется путем извлечения из стека адреса возврата, включенного в него командой вызова подпрограммы. Поэтому команды возврата не содержат никакой адресной информации и неявно адресуют вершину стека. Тип команды возврата (внутрисегментный NEAR или межсегментный FAR) выбирается в соответствии с типом команды CALL, осуществившей вызов данной подпрограммы.

Однобайтовая команда внутрисегментного возврата RET (код операции C3) выполняет следующие действия: слово из вершины стека загружается в IP, а содержимое SP увеличивается на два.

Однобайтовая команда межсегментного возврата RET (код операции CB) осуществляет следующие действия: слово из вершины стека передается в IP; производится инкремент SP на два; слово из следующей ячейки стека передается в CS; производится инкремент SP на два. В результате в регистрах IP и CS оказывается полный адрес возврата, а SP адресует новую вершину стека.

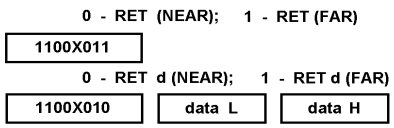


Рис. 3.29. Команды возврата из подпрограмм

Две трехбайтовые команды RET (внутрисегментный возврат с кодом операции C2 и межсегментный возврат с кодом CA) содержат два байта данных, интерпретируемых как беззнаковое целое. Они производят такие же действия, как и соответствующие однобайтовые команды возврата, но дополнительно прибавляют содержащиеся в них данные к указателю стека SP (после извлечения из стека адреса возврата). Эти команды упрощают возврат из тех подпрограмм, параметры которых передаются в стеке. Прибавление к SP числа, равного числу байтов в блоке параметров, эквивалентно удалению этого блока из стека.

В микропроцессоре BM86 нет команд условного возврата, и поэтому условный возврат из подпрограмм обеспечивается командой условного перехода противоположного смысла и безусловного возврата, как это делалось при эмуляции команд условного вызова:

```
    JZ SKIP
    RET
SKIP: .....
```

Команды условных переходов (рис. 3.30). Имеется 18 команд условных переходов, которые представлены единым двухбайтовым форматом, позволяющим осуществлять короткие (в пределах от -128 до +127) переходы относительно указателя команд IP. При выполнении этих команд анализируется некоторое условие, соответствующее текущим состояниям отдельных флагов (кроме флага AF) или некоторым комбинациям флагов (табл. 3.11). Если условие выполнено, то осуществляется переход и однобайтовое смещение, расширенное со знаком до 16 бит, добавляется к содержимому IP. Если условие не выполнено, выполняется следующая по порядку команда. Время выполнения команд составляет восемь тактов в первом случае и четыре такта во втором.

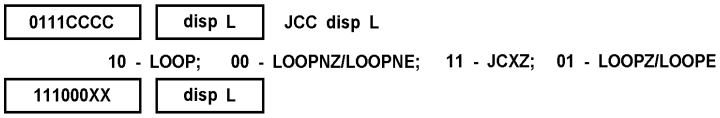


Рис. 3.30. Форматы команд условных переходов и управления циклами

Команды условных переходов обеспечивают ограниченный диапазон переходов, для расширения которого необходимо сочетать их с командами безусловных переходов. Большинство команд условных переходов имеет по два мнемонических обозначения, которые подчеркивают содержательный смысл проверяемого условия и введены для удобства программирования. Обобщенное мнемоническое обозначение команд имеет вид **Jcond label,** где cond соответствует проверяемому условию (первая графа табл. 3.11), двоичный код которого CCCC (вторая графа табл. 3.11), помещается в соответствующее поле первого байта формата команды. Операнд label обозначает команду, к которой осуществляется переход. В качестве label может использоваться метка, а также метка плюс/минус константа или выражение, вычисление которого дает константу.

Команды условных переходов позволяют проверить все отношения между знаковыми и беззнаковыми числами. В мнемокодах этих команд при сравнении знаковых чисел для обозначения условия «больше» используется буква G (Greater), а для обозначения «меньше» — буква L (Less). Для аналогичных условий при сравнении беззнаковых чисел используются соответственно буквы A (Above) и B (Below). Условие равенства обозначается буквой E (Equal), а невыполнение некоторого условия — буквой N (Not).

Таблица 3.11

Мнемокод условия	Код CCCC	Условие перехода	
		Логика	Признак
		Для чисел со знаком	
G/NLE	1111	Больше/не меньше и не равно	(SF⊕OF) ∨ ZF = 0
NG/LE	1110	Не больше/меньше или равно	(SF⊕OF) ∨ ZF =1
L/NGE	1100	Меньше/не больше и не равно	SA⊕OF = 1
NL/GE	1101	Не меньше/больше или равно	SF⊕OF = 0
		Для чисел без знака	
A/NBE	0111	Больше/не меньше и не равно	CF ∨ ZF = 0
NA/BE	0110	Не больше/меньше или равно	CF ∨ ZF = 1
B/NAE/C	0010	Меньше/не больше и не равно	CF = 1
NB/AE/NC	0011	Не меньше/больше или равно	CF = 0
		Для прочих данных	
E/Z	0100	Равно/по нулю	ZF = 1
NE/NZ	0101	Не равно/по не нулю	ZF = 0
S	1000	Отрицательный результат	SF = 1
NS	1001	Положительный результат	SF = 0
O	0000	Есть переполнение	OF = 1
NO	0001	Нет переполнения	OF = 0
P/PE	1010	Четно	PF = 1
NP/PO	1011	Нечетно	PF = 0

Первые восемь команд в табл. 3.11 предназначены в основном для использования после команд сравнения CMP, в которых второй операнд вычитается из первого. В результате сравнения двух операндов и последующего перехода в зависимости от состояния флагов реализуются решения, соответствующие операторам отношений **<,** >, ≤, ≥**,** ≠, =. Последние восемь команд можно использовать в любой ситуации, когда в зависимости от значения указанного в табл. 2.11 флага следует предпринять одно из двух действий.

Команды условных переходов реализуют относительную адресацию, как и первые два формата команды безусловного перехода JMP. При этом встает задача определения конкретного значения смещения dispL,

которое должно быть добавлено к регистру IP для осуществления перехода к указанной метке. Решение этой задачи рассмотрим на примере следующего фрагмента программы:

```
0050 AGAIN: INC  CX
0052        ADD  AX, [BX]
0054        JNS  AGAIN
0056        MOV RESULT, CX
```

В левом столбце приведены эффективные адреса команд, т. е. содержимое IP при выборке каждой команды.

```
-0050
 0056
   -6
```

Вычисление показывает, что при ассемблировании команды JNS следует установить смещение dispL=FA, соответствующее дополнительному коду числа -6.

Команды управления циклами (см. рис. 3.30) используются для удобства реализации вычислительных циклов (итераций). Они осуществляют условный переход в зависимости от состояния регистра CX, который выполняет функции счетчика циклов. Команды LOOP, LOOPE и LOOPNE, кроме того, декрементируют регистр CX перед выполнением условного перехода, что исключает использование соответствующей команды декремента. Как и в командах условных переходов, в данном случае при выполнении условия управление передается команде, расположенной в диапазоне адресов от -128 до +127, задаваемых смещением в байте 2 команды.

Команда LOOP, которая обычно ставится в конце цикла, осуществляет декремент CX - 1 и если CX≠0, то добавляет смещение dispL к регистру IP; в противном случае (CX=0 и цикл окончен) выполняется следующая по порядку команда. Таким образом, смысл этой команды состоит в повторении цикла CX раз, т. е. в выполнении переходов до тех пор, пока CX≠0.

Команда JCXZ имеет противоположный смысл и осуществляет переход только при CX=0. Ее удобно использовать в начале цикла, особенно в ситуации, когда цикл может не выполняться ни разу. Если команда LOOP выполняет постпроверку содержимого CX, то команда JCXZ выполняет его предпроверку.

Фактически команда LOOP label эквивалентна двум командам

```
DEC  CX
JNZ  label
```

и ее использование кроме удобства программирования позволяет экономить один байт объектного кода и один такт Т. При большом числе повторений цикла экономия времени может оказаться значительной.

Команда LOOPE/LOOPZ по сравнению с командой LOOP вводит дополнительное условие повторения цикла: ZF=1. Это позволяет выходить из вычислительного цикла как по окончании заданного числа циклов, так и при получении ненулевого результата (или неравенстве сравниваемых операндов). Команда LOOPNE/LOOPNZ вводит дополнительное условие противоположного смысла: ZF=0, которое позволяет прекращать цикл при получении нулевого результата (или по равенству сравниваемых операндов).

Команды обработки цепочек. Имеется пять однобайтовых команд (рис. 3.31), предназначенных для обработки одного элемента цепочки. Напомним, что под цепочкой (строкой) понимается последовательность любых контекстно связанных байтов или слов, находящихся в смежных ячейках памяти. Цепочечной команде может предшествовать однобайтовый префикс повторения REP, который вызывает повторение действия команды над следующим элементом. Это позволяет обрабатывать цепочки значительно быстрее, чем при организации программного цикла. Повторение рассчитано на максимальную длину цепочки 64 Кбайт и может заканчиваться по одному или двум условиям. Требуемое число повторений указывается в регистре CX, содержимое которого автоматически декрементируется при каждом повторении операции. Когда содержимое CX становится равным нулю, МП выполняет следующую команду. При выполнении повторяющейся цепочечной команды МП воспринимает внешние прерывания.

В качестве операндов команды могут иметь цепочку-источник, цепочку-получатель или то и другое одновременно. Цепочка-источник по умолчанию размещается в текущем сегменте данных, и все элементы адресуются регистром SI. Имеется возможность использовать другой сегмент, указав его в префиксе смены сегмента. Цепочка-получатель всегда размещается в дополнительном сегменте, и ее элементы адресуются регистром DI. Для начальной загрузки адресов в регистры можно использовать команды LEA, LDS и LES.

При выполнении цепочечной команды содержимое SI и DI автоматически изменяется на ±1 (при обработке байта) или на ±2 (при обработке слова), чтобы адресовать следующие элементы цепочек. Флаг DF определяет направление изменения адресов: автоинкремент (DF=0) или автодекремент (DF=1).

Каждая цепочечная команда способна оперировать как с байтами, так и со словами, причем тип элементов цепочек ассемблер может определить по атрибутам операндов. Однако многие версии ассемблера допускают указание типа элементов путем добавления букв B (байт) или W (слово) к мнемоническим обозначениям цепочечных команд: MOSB или MOVSW, CMPSB или SMPSW и т. д. При этом операнды в команде становятся необязательными (ассемблер их игнорирует, так как адресация цепочек известна заранее) и могут указываться только для удобства программирования.

Префикс повторения (см. рис.3.31) обеспечивает повторяющееся (циклическое) выполнение цепочечной команды. При отсутствии префикса команда оперирует только одним элементом цепочки.

Префикс повторения имеет пять мнемокодов: REP/REPE/REPZ и REPNE/REPNZ, которые определяют только два объектных кода, различающихся однобитным полем z, и введены для лучшего отражения содержательного смысла команды. Префикс повторения не влияет на флаги. Он обеспечивает автоматическое декрементирование регистра CX на каждом повторении команды и проверку содержимого CX на нуль.

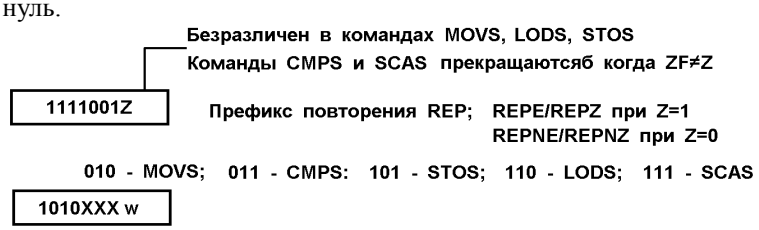


Рис. 3.31. Команды обработки цепочек

Префикс REP используется с командами пересылки MOVS и STOS и инициирует действие «повторять, пока не будет достигнут конец цепочки», т. е. пока CX≠0. Префиксы REPE и REPZ действуют аналогично и имеют такой же машинный код, как и префикс REP. Они используются с командами сравнения CMPS и SCAS и оперируют с флагом ZF, состояние которого определяется результатом исполнения этих команд, причем появление ZF=1 останавливает процесс повторения команд. Префиксы REPNE и REPNZ также применяются командами CMPS и SCAS, но используют противоположное условие ZE=0. Наличие дополнительных проверяемых условий позволяет окончить выполнение этих команд по результату вычислений до завершения перебора всех элементов цепочки.

Таблица 3.12.

Команда	Время выполнения команды	
	без префикса повторения	с префиксом повторения
MOVS	18	9 + 17n
CMPS	18	9 + 17n
SCAS	15	9 + 15n
LODS	12	9 + l3n
STOS	11	9 + 10n

Цепочечные команды пересылки MOVS, LODS и STOS на флаги не влияют, а команды неразрушающего сравнения CMPS и SCAS устанавливают значения флагов в соответствии с разностью операндов, сохраняя значения самих операндов. Время выполнения цепочечных команд (в тактах синхронизации) указано в табл. 3.12, где n - число повторений, которое предварительно заносится в регистр CX.

Команды с префиксом повторения при обработке длинных цепочек могут выполняться в течение значительного времени. Поэтому для сокращения времени реакции на внешние прерывания МП воспринимает внешние сигналы прерывания после обработки каждого элемента цепочки. (Сигнал INTR=1 должен установиться не более чем за один такт до окончания цикла обработки).

После возврата из прерывания операция возобновляется, как обычно, с точки прерывания. Однако во время обработки прерывания МП помнит действие только одного префикса, непосредственно предшествующего команде. Поэтому, если с цепочечной командой используется несколько префиксов (префикс замены сегмента, префикс блокировки), необходимо запрещать прерывания на время ее выполнения.

Обобщенные обозначения цепочечных команд и выполняемые ими действия при различных префиксах приведены в табл. 3.13.

Таблица 3.13

Команда, её действие	Префикс повторения	Описание действия
MOVS dst, src; dst ← src	-	Пересылка из цепочки src, адресуемой регистром SI (сегмент данных) в цепочку dst, адресуемую регистром DI (дополнительный сегмент)
	REP	Блоковая пересылка память-память
CMPS dst, src; src - dst	-	Вычитание элемента цепочки dst из элемента цепочки src (неразрушающее сравнение)
	REPE/REPZ	Сравнение элементов до обнаружения одинаковых элементов или до окончания цепочек

	REPNE/REPNZ	Сравнение элементов до обнаружения различающихся элементов или до окончания цепочек
SCAS dat; ac - dst	-	Вычитание элемента цепочки dst, адресуемой регистром DI, из содержимого AL или AX (неразрушающее сравнение)
	REPE/REPZ	Поиск элемента цепочки со значением, отличающимся от ac
	REPNE/REPNZ	Поиск элемента цепочки со значением, совпадающим с ac
LODS src; ac ← src	-	Загрузка аккумулятора элементом цепочки, адресуемым регистром SI (префикс повторения обычно не используется)
STOS dst; dst ← ac	-	Запоминание содержимого аккумулятора AL или AX в элементе цепочки, адресуемом регистром DI
	REP	Инициализация цепочки на фиксированное значение, содержащееся в аккумуляторе

Команду CMPS удобно применять для нахождения одинаковых (REPE CMPS) или различающихся (REPNE CMPS) элементов цепочек. Достижение конца цепочки при выполнении этих команд соответственно означает, что цепочки полностью различны (не содержат одинаковых элементов) или полностью совпадают (не содержат различающихся элементов). Команда SCAS производит сравнение элемента цепочки с некоторым эталоном, содержащимся в аккумуляторе. Вариант REPE SCAS осуществляет просмотр цепочки до тех пор, пока значение элемента не отличается от эталона или не достигается конец цепочки, а вариант REPNE SCAS — просмотр цепочки до обнаружения элемента, равного эталону, или до достижения конца цепочки.

Команду LODS удобно использовать в программных циклах вместо двух команд: MOV, ac, src и INC SI (или DEC SI в зависимости от направления продвижения по цепочке). Отметим, что команда LODS оперирует элементами цепочки, расположенной в сегменте данных, причем имеется возможность замены сегмента, а команда STOS оперирует элементами цепочки, которая всегда находится в дополнительном сегменте. В последнем случае префикс замены сегмента не указывается, а при его наличии — игнорируется.

В данном разделе использованы материалы из [10].

СПИСОК ЛИТЕРАТУРЫ

1. Павлов В. А. Концепция преподавания дисциплин цикла "Система ввода-вывода ПК". //Вестник СарФТИ. 2004. №6. с.19-26.
2. Павлов В.А. Учебные материалы по курсу «Периферийные устройства ЭВМ»: Часть 1. СарФТИ, Саров, 2001. - 232с.: ил.
3. Гук М. Аппаратные интерфейсы ПК. Энциклопедия. -СПб.: Питер, 2002. -528 с.; ил.
4. Гук М. Аппаратные средства IBM PC. Энциклопедия, 2-е изд. -СПб.: Питер, 2001. - 928 с.: ил.
5. Кулаков В. Программирование на аппаратном уровне: Специальный справочник. 2-е изд. СПб.: Питер, 2003. - 848 с.: ил.
6. Ан П. Сопряжение ПК с внешними устройствами / Пей Ан; Перю с англ. Мерещука П.В. - 2-е изд., стер. - М.: ДМК Пресс; СПб,: Питер, 2004. - 320с.: ил.
7. Вегнер В.А. и др. Аппаратура персональных компьютеров и ее программирование. IBM PC/XT/AT и PS/2. М: Радио и связь, 1995. 224с. - (Библиотека системного программиста).
8. Фролов А.В., Фролов Г.В. Аппаратное обеспечение персонального компьютера. -М.: ДИАЛОГ-МИФИ, 1997. - 304 с. - (Библиотека системного программиста; Т. 33).
9. Новиков Ю.В., Калашников О.А., Гуляев С.Э. Разработка устройств сопряжения для персонального компьютера типа IBM PC. Под общей редакцией Ю.В. Новикова. Практ. пособие - М.: ЭКОМ., 1997 - 224 с.: ил.
10. Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю. М. Казаринов, В. Н. Номоконов, Г. С. Подклктнова, В. Ф. Филиппов; Под ред. Ю. М. Казаринова. - М.: Высш. шк., 1990. - 296 с.: ил.
11. Дао Л. Программирование микропроцессора 8088/Пер. с англ.- М.: Мир,1988.
12. Казаринов Ю. М., Номоконов В. Н., Филиппов Ф. В. Применение микропроцессоров и микроЭВМ в радиотехнических системах.- М.: Высшая школа, 1988.
13. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микрокомпьютерных систем /Пер. с англ.- М.: Радио и связь, 1987. - 512 с.: ил.
14. Левкин Г. Н., Левкина В. Е. Введение в схемотехнику ПЭВП PC/AT. - М.: Изд-во МПИ, 1991. - 96 с.: ил.