|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **MS-DOS для программиста**  *© Александр Фролов, Григорий Фролов Том 4, М.: Диалог-МИФИ, 1993, 254 стр.* |   **4. Прерывания**   * [4.1. Таблица векторов прерываний](http://www.frolov-lib.ru/books/bsp.old/v18/ch4.html#b4.1) * [4.2. Маскирование прерываний](http://www.frolov-lib.ru/books/bsp.old/v18/ch4.html#b4.2) * [4.3. Изменение таблицы векторов прерываний](http://www.frolov-lib.ru/books/bsp.old/v18/ch4.html#b4.3) * [4.4. Особенности обработки аппаратных прерываний](http://www.frolov-lib.ru/books/bsp.old/v18/ch4.html#b4.4)   Для обработки событий, происходящих асинхронно по отношению к выполнению программы, лучше всего подходит механизм прерываний. Прерывание можно рассматривать как некоторое особое событие в системе, требующее моментальной реакции.  Практически все системы ввода/вывода в компьютере работают с использованием прерываний. В частности, когда вы нажимаете клавиши или щелкаете мышью, аппаратура вырабатывает прерывания. В ответ на них система, соответственно, считывает код нажатой клавиши или запоминает координаты курсора мыши. Прерывания вырабатываются контроллером диска, адаптером локальной сети, портами последовательной передачи данных, звуковым адаптером и другими устройствами.  Кажется очевидным, что возможны самые разнообразные прерывания по самым различным причинам. Поэтому с прерыванием связывают число - так называемый номер прерывания.  Этот номер однозначно соответствует тому или иному событию. Система умеет распознавать прерывания и при их возникновении запускает процедуру, соответствующую номеру прерывания.  Некоторые прерывания (первые пять по порядку номеров) зарезервированы для использования центральным процессором на случай каких-либо особых событий вроде попытки деления на нуль, переполнения и т. п.  Программы могут сами вызывать прерывания с заданным номером. Для этого они используют команду INT. Это так называемые программные прерывания. Программные прерывания не являются асинхронными, так как вызываются из программы.  Программные прерывания удобно использовать для организации доступа к отдельным, общим для всех программ функциям. Например, функции операционной системы доступны прикладным программам именно через прерывания. При вызове этих модулей нет необходимости знать их текущий адрес в памяти.  Прикладные программы и драйверы могут сами устанавливать свои обработчики прерываний для их последующего использования другими программами. Для этого встраиваемые обработчики прерываний должны быть резидентными в памяти.  Аппаратные прерывания вызываются физическими устройствами и потому приходят асинхронно по отношению к выполнению любых программ. Эти прерывания информируют систему о событиях, связанных с работой устройств. Например, о том, что наконец-то завершилась печать символа на принтере и неплохо было бы выдать следующий символ, или о том, что сектор диска уже прочитан и его содержимое доступно программе.  Использование прерываний при работе с медленными внешними устройствами позволяют совместить ввод/вывод с обработкой данных в центральном процессоре. В результате этого повышается общая производительность системы.  Иногда желательно сделать систему нечувствительной ко всем или отдельным аппаратным прерываниям . Для этого используют так называемое маскирование прерываний, о котором мы еще будем говорить. Но существует и немаскируемое прерывание (которое, кстати, все-таки можно замаскировать, или, точнее говоря, заблокировать).  Заметим, что обработчики прерываний могут сами вызывать программные прерывания, например, для получения доступа к сервису BIOS или MS-DOS.  Составление собственных программ обработки прерываний и замена стандартных обработчиков MS-DOS и BIOS является достаточно сложной задачей. Необходимо учитывать все тонкости работы аппаратуры, а также взаимодействия программного и аппаратного обеспечения. При отладке возможно разрушение операционной системы с непредсказуемыми последствиями, поэтому надо очень внимательно следить за тем, что делает ваша программа.  **4.1. Таблица векторов прерываний**  Для того чтобы связать адрес обработчика прерывания с номером прерывания, используется таблица векторов прерываний , занимающая первый килобайт оперативной памяти. Эта таблица находится в диапазоне адресов от 0000:0000 до 0000:03FFh и состоит из 256 элементов - дальних адресов обработчиков прерываний.  Элементы таблицы векторов прерываний называются векторами прерываний. В первом слове элемента таблицы записана компонента смещения, а во втором - сегментная компонента адреса обработчика прерывания.  Вектор прерывания с номером 0 находится по адресу 0000:0000, с номером 1 - по адресу 0000:0004 и т. д.  Для программиста, использующего язык С, таблицу векторов прерываний можно описать следующим образом:  void (far\* interrupt\_table[256])();  Инициализация таблицы выполняется частично системой базового ввода/вывода BIOS после тестирования аппаратуры и перед началом загрузки операционной системой, частично при загрузке MS-DOS. Операционная система MS-DOS может изменить некоторые вектора прерываний, установленные BIOS.  Расскажем о назначении наиболее важных векторов прерываний.   |  |  | | --- | --- | | **Номер** | **Описание** | | 0 | Ошибка деления.Вызывается автоматически после выполнения команд DIV или IDIV, если в результате деления происходит переполнение (например, при делении на 0). Обычно при обработке этого прерывания MS-DOS выводит сообщение об ошибке и останавливает выполнение программы. При этом для процессора i8086 адрес возврата указывает на команду, следующую после команды деления, а для процессора i80286 и более старших моделей - на первый байт команды, вызвавшей прерывание | | 1 | Прерывание пошагового режима.Вырабатывается после выполнения каждой машинной команды, если в слове флагов установлен бит пошаговой трассировки TF. Используется для отладки программ. Это прерывание не вырабатывается после пересылки данных в сегментные регистры командами MOV и POP | | 2 | Аппаратное немаскируемое прерывание.Это прерывание может использоваться по-разному в разных машинах. Обычно оно вырабатывается при ошибке четности в оперативной памяти и при запросе прерывания от сопроцессора | | 3 | Прерывание для трассировки.Генерируется при выполнении однобайтовой машинной команды с кодом CCh и обычно используется отладчиками для установки точки прерывания | | 4 | Переполнение. Генерируется машинной командой INTO , если установлен флаг переполнения OF. Если флаг не установлен, команда INTO выполняется как NOP. Это прерывание используется для обработки ошибок при выполнении арифметических операций | | 5 | Печать копии экрана. Генерируется, если пользователь нажал клавишу <PrtSc>. В программах MS-DOS обычно используется для печати образа экрана. Для процессора i80286 и более старших моделей генерируется при выполнении машинной команды BOUND, если проверяемое значение вышло за пределы заданного диапазона | | 6 | Неопределенный код операции или длина команды больше 10 байт | | 7 | Особый случай отсутствия арифметического сопроцессора | | 8 | IRQ0 - прерывание интервального таймера, возникает 18,2 раза в секунду | | 9 | IRQ1 - прерывание от клавиатуры. Генерируется, когда пользователь нажимает и отжимает клавиши. Используется для чтения данных из клавиатуры | | A | IRQ2 - используется для каскадирования аппаратных прерываний | | B | IRQ3 - прерывание асинхронного порта COM2 | | C | IRQ4 - прерывание асинхронного порта COM1 | | D | IRQ5 - прерывание от контроллера жесткого диска (только для компьютеров IBM PC/XT) | | E | IRQ6 - прерывание генерируется контроллером НГМД после завершения операции ввода/вывода | | F | IRQ7 - прерывание от параллельного адаптера. Генерируется, когда подключенный к адаптеру принтер готов к выполнению очередной операции. Обычно не используется | | 10 | Обслуживание видеоадаптера | | 11 | Определение конфигурации устройств в системе | | 12 | Определение размера оперативной памяти | | 13 | Обслуживание дисковой системы | | 14 | Работа с асинхронным последовательным адаптером | | 15 | Расширенный сервис | | 16 | Обслуживание клавиатуры | | 17 | Обслуживание принтера | | 18 | Запуск BASIC в ПЗУ, если он есть | | 19 | Перезагрузка операционной системы | | 1A | Обслуживание часов | | 1B | Обработчик прерывания, возникающего, если пользователь нажал комбинацию клавиш <Ctrl+Break> | | 1C | Программное прерывание, вызывается 18,2 раза в секунду обработчиком аппаратного прерывания таймера | | 1D | Адрес видеотаблицы для контроллера видеоадаптера 6845 | | 1E | Указатель на таблицу параметров дискеты | | 1F | Указатель на графическую таблицу для символов с кодами ASCII 128-255 | | 20-5F | Используется MS-DOS или зарезервировано для MS-DOS | | 60-67 | Прерывания, зарезервированные для программ пользователя | | 68-6F | Не используются | | 70 | IRQ8 - прерывание от часов реального времени | | 71 | IRQ9 - прерывание от контроллера EGA | | 72 | IRQ10 - зарезервировано | | 73 | IRQ11 - зарезервировано | | 74 | IRQ12 - зарезервировано | | 75 | IRQ13 - прерывание от арифметического сопроцессора | | 76 | IRQ14 - прерывание от контроллера жесткого диска | | 77 | IRQ15 - зарезервировано | | 78 - 7F | Не используются | | 80-85 | Зарезервировано для BASIC | | 86-F0 | Используются интерпретатором BASIC | | F1-FF | Не используются |   Прерывания, обозначенные как IRQ0 - IRQ15 являются аппаратными прерываниями.  **4.2. Маскирование прерываний**  Часто при выполнении критических участков программ приходится запрещать прерывания для того чтобы гарантировать непрерываемое выполнение определенной последовательности команд. Это можно сделать командой CLI. Ее нужно поместить в начало критической последовательности команд, а в конце расположить команду STI, разрешающую процессору воспринимать прерывания. Команда CLI запрещает только маскируемые прерывания, на обработку немаскируемого прерывания эта команда никакого влияния не оказывает.  Если вы используете запрет прерываний с помощью команды CLI, следите за тем, чтобы прерывания не отключались на длительный период времени, так как это может привести к нежелательным последствиям. Например, к отставанию системных часов или неправильной работе периферийных устройств компьютера.  Если вам надо запретить не все прерывания, а только некоторые, например, от клавиатуры, то для этого надо воспользоваться услугами контроллера прерываний. Записывая в этот контроллер определенную управляющую информацию, можно замаскировать прерывания от отдельных устройств.  **4.3. Изменение таблицы векторов прерываний**  Вашей программе может потребоваться организовать обработку некоторых прерываний. Для этого программа должна установить векторы нужных прерываний на свой обработчик. Это можно сделать, изменив содержимое соответствующего элемента таблицы векторов прерываний.  Очень важно не забыть перед завершением работы восстановить содержимое измененных векторов в таблице прерываний.  Дело в том, что память, которая была распределена программе, после завершения работы программы освобождается. Она может быть использована, например, для загрузки другой программы. Если вы забыли восстановить вектор и пришло прерывание, то система может разрушиться - вектор теперь указывает на область, которая может содержать что угодно.  Поэтому последовательность действий для нерезидентных программ, желающих обрабатывать прерывания, должна быть такой:   * *прочитать содержимое элемента таблицы векторов прерываний для вектора с нужным вам номером;* * *запомнить это содержимое (адрес старого обработчика прерывания) в области данных программы;* * *установить новый адрес в таблице векторов прерываний так, чтобы он указывал на начало вашей программы обработки прерывания;* * *перед завершением работы программы прочитать из области данных адрес старого обработчика прерывания и записать его в таблицу векторов прерываний.*   Кроме того, операция изменения вектора прерывания должна быть непрерывной в том смысле, что во время изменения не должно произойти прерывание. Если вы, например, запишете новое значение смещения, а сегментный адрес обновить не успеете, то по какому адресу будет передано управление в случае прерывания и что при этом произойдет? Об этом можно только догадываться.  **Функции MS-DOS для работы с таблицей прерываний**  Для облегчения работы по замене векторов прерываний MS-DOS предоставляет в ваше распоряжение специальные функции, предназначенные для чтения элемента таблицы векторов прерывания и для записи в нее нового адреса. Если вы будете использовать эти функции, MS-DOS гарантирует, что операция по замене вектора будет выполнена правильно. Вам не надо заботиться о непрерывности процесса замены вектора прерывания.  Для чтения вектора используйте функцию 35h прерывания INT 21h . Перед ее вызовом регистр AL должен содержать номер вектора в таблице. После выполнения функции в регистрах ES:BX будет искомый адрес обработчика прерывания.  Для вектора с номером, находящимся в регистре AL, функция 25h прерывания INT 21h устанавливает новый обработчик прерывания. Адрес обработчика прерываний следует передать через регистры DS:DX.  Разумеется, вы можете также обращаться к таблице векторов прерываний непосредственно, но тогда при записи необходимо замаскировать прерывания командой CLI, не забыв разрешить их после записи командой STI.  Пользователям языка С доступны функции \_dos\_getvect и \_dos\_setvect . Первая функция получает адрес из таблицы векторов прерываний, вторая устанавливает новый адрес. Обе эти функции обращаются к описанным выше функциям 35h и 25h прерывания INT 21h .  Какие требования предъявляются к программе обработки прерывания?  Если прерывания происходят часто, то их обработка может сильно замедлить работу прикладной программы. Поэтому обработчик прерывания должен быть короткой, быстро работающей программой, которая выполняет только самые необходимые действия. Например, чтение очередного символа из порта принтера и запись его в буфер, увеличение значения какого-либо глобального счетчика прерываний и т. п.  **Цепочки обработчиков прерываний**  Если вам надо добавить какие-либо собственные действия к тем, что выполняет стандартный обработчик прерывания, то можно организовать цепочку прерываний.  Для организации цепочки прерываний нужно записать в векторную таблицу адрес своего обработчика, не забыв сохранить прежнее содержимое таблицы. Ваш обработчик получает управление по прерыванию, выполняет какие-либо действия, затем передает управление старому обработчику.  Можно сделать и по-другому: ваш обработчик вызывает старый обработчик как подпрограмму, а затем после возврата из старого обработчика выполняет дополнительные действия. Иными словами, вы можете вставить дополнительную обработку как до вызова старого обработчика, так и после его вызова.  В библиотеке транслятора C имеется функция для организации цепочки прерываний с именем \_chain\_intr .  Для описания функции, выполняющей роль обработчика прерывания, следует использовать ключевое слово interrupt.  Такая функция завершается командой возврата из обработки прерывания IRET. Для нее автоматически генерируются команды сохранения регистров на входе и их восстановления при выходе из обработчика прерывания. Пример использования ключевого слова interrupt для определения функции обработки прерывания:  void interrupt far int\_funct(...)  {  // Тело обработчика прерывания  }  Функция обработки прерывания должна быть дальней функцией, так как таблица векторов прерываний содержит полные адреса в формате <сегмент:смещение>.  Ключевое слово interrupt используется также для описания переменных, предназначенных для хранения векторов прерываний:  void (interrupt (far \*oldvect)(...);  Для установки своего обработчика прерываний используйте функцию \_dos\_setvec . Эта функция имеет два параметра - номер прерывания и указатель на новую функцию обработки прерывания.  Например:  \_dos\_setvect (0x16, my\_key\_intr);  В этом примере для прерывания с номером 16h (программное прерывание, предназначенное для чтения данных из клавиатуры) устанавливается новый обработчик прерывания my\_key\_intr.  Если вам надо узнать адрес старого обработчика прерывания по его номеру, лучше всего воспользоваться функцией \_dos\_getvect , которая принимает в качестве параметра номер прерывания и возвращает указатель на соответствующий этому номеру обработчик.  Например:  old\_vector = \_dos\_getvect (0x16);  Для организации цепочки прерываний используйте функцию \_chain\_intr . В качестве параметра эта функция принимает адрес старого обработчика прерываний.  **Программа BEEPER**  Программа BEEPER (листинг 4.1) - простой пример, который иллюстрирует применение всех трех перечисленных выше функций, предназначенных для работы с прерываниями.  **Листинг 4.1. Файл beeper\beeper.cpp**  #include <dos.h>  #include <stdio.h>  #include <conio.h>  void main(void);  void interrupt far timer(...);  void interrupt (far \*oldvect)(...);  // Переменная для подсчета прерываний таймера  volatile long ticks;  void main(void)  {  // Сбрасываем счетчик  ticks = 0L;  // Запоминаем адрес старого обработчика  // прерывания  oldvect = \_dos\_getvect (0x1c);  // Устанавливаем новый обработчик прерывания  \_dos\_setvect (0x1c, timer);  printf("\nТаймер установлен. Нажмите любую"  " клавишу...\n");  getch();  // Восстанавливаем старый обработчик прерывания  \_dos\_setvect (0x1c,oldvect);  }  void interrupt far timer(...)  {  // Увеличиваем счетчик прерываний таймера  ticks++;  // Если значение счетчика кратно 20,  // выдаем сигнал на громкоговоритель компьютера  if((ticks % 20) == 0)  {  asm mov bx,0  asm mov ax, 0E07h  asm int 10h  }  // Вызываем старый обработчик прерывания  \_chain\_intr (oldvect);  }  Эта программа встраивает собственный обработчик прерывания таймера, который будет вызываться примерно 18,2 раза в секунду. Встраиваемый обработчик прерывания подсчитывает прерывания таймера и, если значение соответствующего счетчика кратно 20, громкоговоритель компьютера издает звуковой сигнал.  В конце работы новая программа обработки прерывания таймера вызывает старый обработчик с помощью функции \_chain\_intr .  После установки нового обработчика прерывания таймера основная программа ждет, когда пользователь нажмет любую клавишу. Затем она восстанавливает старое содержимое вектора прерывания.  **4.4. Особенности обработки аппаратных прерываний**  Аппаратные прерывания вырабатываются устройствами компьютера, как правило, при завершении ими операций обмена данными или при изменении состояния. В зависимости от типа устройства обработчик прерывания может выполнять те или иные функции. Например, по прерыванию таймера соответствующий обработчик увеличивает содержимое счетчика, расположенного в оперативной памяти. По содержимому этого счетчика программы могут определить текущее время.  В отличие от программных прерываний, вызываемых запланировано программой или драйвером, аппаратные прерывания всегда происходят асинхронно по отношению к выполняющимся программам. Кроме того, может возникнуть одновременно сразу несколько прерываний!  Для того чтобы система "не растерялась", решая какое прерывание обслуживать в первую очередь, существует специальная схема приоритетов. Каждому прерыванию назначается свой приоритет. Если происходит одновременно несколько прерываний, система отдает предпочтение самому высокоприоритетному, откладывая на время обработку остальных прерываний.  Система приоритетов реализована на двух микросхемах Intel 8259 (или аналогичных). Каждая микросхема является контроллером прерывания и обслуживает до восьми приоритетов. Микросхемы можно объединять (каскадировать) для увеличения количества уровней приоритетов в системе.  Уровни приоритетов обозначаются сокращенно IRQ0 - IRQ15 .  В компьютере типа IBM PC/XT была установлена только одна микросхема контроллера прерывания. Приоритеты линейно зависели от номера уровня прерывания. Прерывание IRQ0 соответствовало самому высокому приоритету, за ним шли прерывания IRQ1 , IRQ2 , IRQ3 и так далее.  Прерывание IRQ2 в компьютерах IBM PC/XT было зарезервировано для дальнейшего расширения системы. В компьютерах IBM PC/AT прерывание IRQ2 стало использоваться для каскадирования двух контроллеров прерывания 8259. Добавленные приоритетные уровни прерываний IRQ8 - IRQ15 в этих компьютерах располагаются по приоритету между прерываниями IRQ1 и IRQ3 .  Приведем список аппаратных прерываний, расположенных в порядке убывания приоритета:   |  |  | | --- | --- | | **Номер** | **Описание** | | 8 | IRQ0 - прерывание интервального таймера, возникает 18,2 раза в секунду | | 9 | IRQ1 - прерывание от клавиатуры | | A | IRQ2 - используется для каскадирования аппаратных прерываний | | 70 | IRQ8 - прерывание от часов реального времени | | 71 | IRQ9 - прерывание от контроллера EGA | | 72 | IRQ10 - зарезервировано | | 73 | IRQ11 - зарезервировано | | 74 | IRQ12 - зарезервировано | | 75 | IRQ13 - прерывание от арифметического сопроцессора | | 76 | IRQ14 - прерывание от контроллера жесткого диска | | 77 | IRQ15 - зарезервировано | | B | IRQ3 - прерывание асинхронного порта COM2 | | C | IRQ4 - прерывание асинхронного порта COM1 | | D | IRQ5 - прерывание от контроллера жесткого диска (только в компьютерах IBM PC/XT) | | E | IRQ6 - прерывание генерируется контроллером НГМД | | F | IRQ7 - прерывание принтера |   Из этого списка видно, что самый высокий приоритет у прерываний от интервального таймера, затем идет прерывание от клавиатуры. Наименьший приоритет имеет прерывание принтера.  Для управления схемами приоритетов необходимо знать внутреннее устройство контроллера прерываний 8259.  Поступающие прерывания запоминаются в регистре запроса на прерывание IRR. Каждый бит из восьми в этом регистре соответствует своему прерыванию.  Перед выдачей в процессор запроса на прерывание проверяется содержимое восьмибитового регистра маски прерываний IMR. Если прерывание данного уровня не замаскировано, то запрос на прерывание выдается.  Наиболее интересными с точки зрения программирования контроллера прерываний являются регистры маски прерываний IMR и управляющий регистр прерываний.  В компьютерах IBM PC/XT регистр маски прерываний имеет адрес 21h, управляющий регистр прерываний - 20h. В компьютерах IBM PC/AT первый контроллер 8259 имеет такие же адреса, что и в IBM PC/XT. Регистр маски прерываний второго контроллера имеет адрес A1h, управляющий регистр прерываний - адрес A0h.  Разряды регистра маски прерываний соответствуют номерам IRQ. Для того чтобы замаскировать аппаратное прерывание какого-либо уровня, надо записать в регистр маски байт масок. В этом байте следует установить в 1 те биты, которые соответствуют маскируемым прерываниям . Например, для маскирования прерываний от НГМД в порт 21h надо записать двоичное число 01000000.  Приведем строку программы, маскирующей прерывание от флоппи-диска:  outp(0x21, 0x40);  Чтобы "оживить" прерывания от НГМД , используйте следующую строку (которая размаскирует все прерывания):  outp(0x21, 0);  Заметьте, что в приведенном выше примере мы замаскировали прерывание именно от НГМД , все остальные устройства продолжали нормально работать. Если бы мы выдали машинную команду CLI, то отключились бы все аппаратные прерывания. Это привело бы, например, к тому, что клавиатура была бы заблокирована.  Еще одно замечание, касающееся обработки аппаратных прерываний.  Если вы полностью заменяете стандартный обработчик аппаратного прерывания, не забудьте в конце программы записать байт 20h в порт с адресом 20h (A0h для второго контроллера 8259). Эти действия необходимы для очистки регистра обслуживания прерывания ISR. При этом разрешается обработка прерываний с более низким приоритетом чем то, которое только что обрабатывалось.  Если вы обрабатываете прерывание 1Ch, то указанная выше добавка в конце программы обработки прерывания не нужна, так как это прерывание является программным и вызывается из обработчика аппаратного прерывания таймера.  Перед тем как завершить изучение прерываний, зададимся вопросом - можно ли замаскировать немаскируемое прерывание ? Оказывается можно!  Конечно, если сигнал прерывания пришел на вход немаскируемого прерывания процессора, ничего сделать нельзя - прерывание произойдет неизбежно. Но в компьютере предусмотрены схемы, блокирующие вход немаскируемого прерывания процессора NMI.  Для компьютера IBM PC/XT маскированием немаскируемого прерывания управляет порт с адресом 0A0h. Если записать в него 0, немаскируемое прерывание будет запрещено, если 80h - разрешено.  Аналогично для IBM PC/AT маскированием немаскируемого прерывания управляет бит 7 порта 70h. Запись байта 0ADh в порт 70h запретит немаскируемое прерывание, а байта 2Dh - разрешит прохождение прерывания.  Заметим, что мы не запрещаем немаскируемое прерывание "внутри" процессора - это невозможно по определению, мы "не пускаем" сигнал прерывания на вход NMI.   |  |  |  | | --- | --- | --- | | [[Назад]](http://www.frolov-lib.ru/books/bsp.old/v18/ch3.html) | [[Содеожание]](http://www.frolov-lib.ru/books/bsp.old/v18/index.html) | [[Дальше]](http://www.frolov-lib.ru/books/bsp.old/v18/ch5.html) | |  |

|  |  |  |
| --- | --- | --- |
| http://www.frolov-lib.ru/images/bot_pict.jpg | Создание интернет-магазинов: http://www.shop2you.ru/ | © Александр Фролов, Григорий Фролов, 1991-2016 |