Тема 2.8 Универсальные и специализированные микропроцессоры

По назначению различают универсальные и специализированные микропроцессоры.

Универсальные МП предназначены для решения широкого круга задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач В системе команд МП заложена алгоритмическая универсальность, означающая, что выполняемый машиной состав команд позволяет получить преобразование информации в соответствии с любым заданным алгоритмом. К универсальным МП относятся и секционные микропроцессоры, поскольку для них система команд может быть оптимизирована в каждом частном проекте создания секционного микропроцессора. Эта группа МП наиболее многочисленна.

Специализированные МП предназначены для решения определенного класса задач, а иногда только для решения одной конкретной задачи. Их существенными особенностями являются простота управления, компактность аппаратурных средств, низкая стоимость и малая мощность потребления. Специализированные МП имеют ориентацию на ускоренное выполнение определенных функций, что позволяет резко увеличить эффективную производительность при решении только определенных задач. Среди специализированных микропроцессоров можно выделить различные микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций; математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения; МП для обработки данных в различных областях применений и т. д.

В настоящее время выпускается целый ряд типов микроконтроллеров. Все эти приборы можно условно разделить на три основных класса:

- 8-разрядные микроконтроллеры для встраиваемых приложений;
- 16- и 32-разрядные микроконтроллеры;
- цифровые сигнальные процессоры (DSP).

Наиболее распространенным представителем семейства микроконтроллеров являются 8-разрядные приборы, широко используемые в промышленности, бытовой и компьютерной технике. Они прошли в своем развитии путь от простейших приборов с относительно слаборазвитой периферией до современных многофункциональных контроллеров, обеспечивающих реализацию сложных алгоритмов управления в реальном масштабе времени. Причиной жизнеспособности 8-разрядных микроконтроллеров является использование их для управления реальными объектами, где применяются, в основном, алгоритмы с преобладанием логических операций, скорость обработки которых практически не зависит от разрядности процессора.

Росту популярности 8-разрядных микроконтроллеров способствует постоянное расширение номенклатуры изделий, выпускаемых такими известными фирмами, как Motorola, Microchip, Intel, Zilog, Atmel и многими другими. Современные 8-разрядные МК обладают, как правило, рядом отличительных признаков. Перечислим основные из них:

- модульная организация, при которой на базе одного процессорного ядра (центрального процессора) проектируется ряд (линейка) микроконтроллеров, различающихся объемом и типом памяти программ, объемом памяти данных, набором периферийных модулей, частотой синхронизации;
- использование закрытой архитектуры микроконтроллера, которая характеризуется отсутствием линий магистралей адреса и данных на выводах корпуса микроконтроллера. Таким образом, микроконтроллеры представляет собой законченную систему обработки данных, наращивание возможностей которой с использованием параллельных магистралей адреса и данных не предполагается;
- использование типовых функциональных периферийных модулей (таймеры, процессоры событий, контроллеры последовательных интерфейсов, аналого-цифровые преобразователи и др.), имеющих незначительные отличия в алгоритмах работы в микроконтроллерах различных производителей;

- расширение числа режимов работы периферийных модулей, которые задаются в процессе инициализации регистров специальных функций микроконтроллера.

При модульном принципе построения все микроконтроллеры одного семейства содержат процессорное ядро, одинаковое для всех микроконтроллеров данного семейства, и изменяемый функциональный блок, который отличает микроконтроллеры разных моделей. Структура модульного микроконтроллера приведена на рис.

Процессорное ядро включает в себя:

- центральный процессор;
- внутреннюю контроллерную магистраль (ВКМ) в составе шин адреса, данных и управления;
- схему синхронизации микроконтроллера;
- схему управления режимами работы микроконтроллера, включая поддержку режимов пониженного энергопотребления, начального запуска (сброса) и т.д.

Изменяемый функциональный блок включает в себя модули памяти различного типа и объема, порты ввода/вывода, модули тактовых генераторов (Г), таймеры. В относительно простых микроконтроллерах модуль обработки прерываний входит в состав процессорного ядра. В более сложных МКон представляет собой отдельный модуль с развитыми возможностями. В состав изменяемого функционального блока могут входить и такие дополнительные модули как компараторы напряжения, аналого-цифровые преобразователи (АЦП) и другие. Каждый модуль проектируется для работы в составе микроконтроллера с учетом протокола ВКМ. Данный подход позволяет создавать разнообразные по структуре микроконтроллеры в пределах одного семейства.

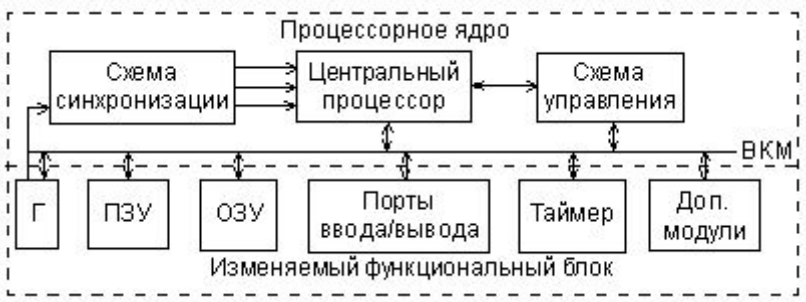

Рис. Модульная организация микроконтроллера

Основными характеристиками, определяющими производительность *процессорного ядра* микроконтроллера, являются:

- набор регистров для хранения промежуточных данных;
- *система команд* процессора;
- *способы адресации* операндов в пространстве памяти;
- организация процессов выборки и исполнения *команды*.

С точки зрения *системы команд* и способов адресации операндов *процессорное ядро* современных 8-разрядных микроконтроллеров реализует один из двух принципов построения процессоров:

- процессоры с *CISC*-архитектурой, реализующие так называемую полную *систему команд* (Complicated *Instruction Set* Computer);
- процессоры с RISC-архитектурой, реализующие сокращенную *систему команд* (*Reduced Instruction Set* Computer).

CISC-процессоры выполняют большой набор *команд* с развитыми возможностями адресации, давая разработчику возможность выбрать наиболее подходящую *команду* для выполнения необходимой операции. В применении к 8-разрядным *МК* процессор с *CISC*-архитектурой может иметь однобайтовый, двухбайтовый и трехбайтовый (редко четырехбайтовый) формат *команд*. При этом *система команд*, как правило, неортогональна, то есть не все *команды* могут использовать любой из способов адресации применительно к любому из регистров процессора. Выборка *команды* на исполнение осуществляется побайтно в течение нескольких *циклов работы МК*. Время выполнения *команды* может составлять от 1 до 12 циклов.

В процессорах с RISC-архитектурой набор исполняемых *команд* сокращен до минимума. Для реализации более сложных операций приходится комбинировать *команды*. При этом все *команды* имеют формат фиксированной длины (например, 12, 14 или 16 бит), выборка *команды* из памяти и ее исполнение осуществляется за один цикл (такт) синхронизации. *Система команд* RISC-процессора предполагает возможность равноправного использования всех регистров процессора. Это обеспечивает дополнительную гибкость при выполнении ряда операций. К *МК* с RISC-процессором относятся *МК AVR* фирмы Atmel, *МК* PIC16 и PIC17 фирмы *Microchip* и другие.

На первый взгляд, *МК* с RISC-процессором должны иметь более высокую производительность по сравнению с *CISC МК* при одной и той же тактовой частоте внутренней магистрали. Однако на практике вопрос о производительности более сложен и неоднозначен.

Во-первых, оценка производительности *МК* по времени выполнения *команд* различных систем (RISC и *CISC*) не совсем корректна. Обычно производительность МП и *МК* принято оценивать числом операций пересылки "регистр-регистр", которые могут быть выполнены в течение одной секунды. В *МК* с *CISC*-процессором время выполнения операции "регистр-регистр" составляет от 1 до 3 циклов, что, казалось бы, уступает производительности *МК* с RISC-процессором. Однако стремление к сокращению формата *команд* при сохранении ортогональности *системы команд* RISC-процессора приводит к вынужденному ограничению числа доступных в одной *команде* регистров. Так, например, *системой команд МК* PIC16 предусмотрена возможность пересылки результата операции только в один из двух регистров — регистр-источник операнда f или рабочий регистр W. Таким образом, операция пересылки содержимого одного из доступных регистров в другой (не источник операнда и не рабочий) потребует использования двух *команд*. Такая необходимость часто возникает при пересылке содержимого одного из регистров общего назначения (РОН) в один из портов *МК*. В то же время, в *системе команд* большинства *CISC*-процессоров присутствуют *команды* пересылки содержимого РОН в один из портов ввода/вывода. То есть более сложная *система команд* иногда позволяет реализовать более эффективный способ выполнения операции.

Во-вторых, оценка производительности *МК* по скорости пересылки "регистр-регистр" не учитывает особенностей конкретного реализуемого алгоритма управления. Так, при разработке быстродействующих устройств автоматизированного управления основное внимание следует уделять времени выполнения операций умножения и деления при реализации уравнений различных *передаточных функций*. А при реализации пульта дистанционного управления бытовой техникой следует оценивать время выполнения логических функций, которые используются при опросе клавиатуры и генерации последовательной кодовой посылки управления. Поэтому в критических ситуациях, требующих высокого быстродействия, следует оценивать производительность на множестве тех операций, которые преимущественно используются в алгоритме управления и имеют ограничения по времени выполнения.

В-третьих, необходимо еще учитывать, что указанные в справочных данных на *МК* частоты синхронизации обычно соответствуют частоте подключаемого кварцевого резонатора, в то время как длительность цикла центрального процессора определяется частотой обмена по ВКМ. Соотношение этих частот индивидуально для каждого *МК* и должно быть принято в расчет при сравнении производительности различных моделей контроллеров.

С точки зрения организации процессов выборки и исполнения *команды* в современных 8-разрядных *МК* применяется одна из двух уже упоминавшихся архитектур МПС: фон-неймановская (принстонская) или гарвардская.

Основной особенностью фон-неймановской архитектуры является использование общей памяти для хранения программ и данных, как показано на рис.

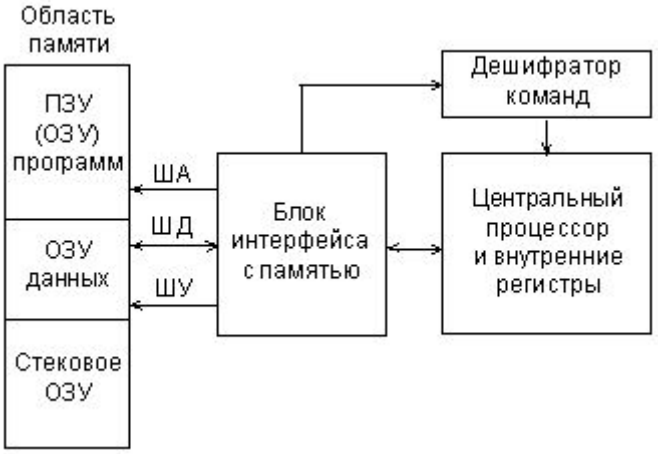

Рис. Структура МПС с фон-неймановской архитектурой

Основное преимущество архитектуры Фон-Неймана – упрощение устройства МПС, так как реализуется обращение только к одной общей памяти. Кроме того, использование единой области памяти позволяло оперативно перераспределять ресурсы между областями программ и данных, что существенно повышало гибкость МПС с точки зрения разработчика программного обеспечения. Размещение *стека* в общей памяти облегчало доступ к его содержимому. Неслучайно поэтому *фон-неймановская архитектура* стала основной архитектурой универсальных компьютеров, включая персональные компьютеры.

Основной особенностью гарвардской архитектуры является использование раздельных адресных пространств для хранения *команд* и данных, как показано на рис.

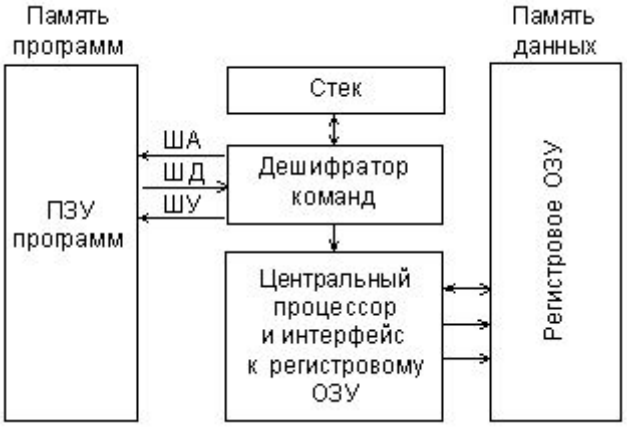

Рис. Структура МПС с гарвардской архитектурой

Гарвардская архитектура почти не использовалась до конца 70-х годов, пока производители *МК* не поняли, что она дает определенные преимущества разработчикам автономных систем управления.

Дело в том, что, судя по опыту использования МПС для управления различными объектами, для реализации большинства алгоритмов управления такие преимущества фон-неймановской архитектуры как гибкость и универсальность не имеют большого значения. Анализ реальных программ управления показал, что необходимый объем *памяти данных МК*, используемый для хранения промежуточных результатов, как правило, на порядок меньше требуемого объема *памяти программ*. В этих условиях использование *единого адресного пространства* приводило к увеличению формата *команд* за счет увеличения числа разрядов для адресации операндов. Применение отдельной небольшой по объему *памяти данных* способствовало сокращению длины *команд* и ускорению поиска информации в *памяти данных*.

Кроме того, гарвардская архитектура обеспечивает потенциально более высокую скорость выполнения программы по сравнению с фон-неймановской за счет возможности реализации

параллельных операций. Выборка следующей *команды* может происходить одновременно с выполнением предыдущей, и нет необходимости останавливать процессор на время выборки *команды*. Этот метод реализации операций позволяет обеспечивать выполнение различных *команд* за одинаковое число тактов, что дает возможность более просто определить время выполнения циклов и критичных участков программы.

Большинство производителей современных 8-разрядных *МК* используют гарвардскую архитектуру. Однако гарвардская архитектура является недостаточно гибкой для реализации некоторых программных процедур. Поэтому сравнение *МК*, выполненных по разным архитектурам, следует проводить применительно к конкретному приложению.

Система команд процессора МК. Так же, как и в любой *микропроцессорной системе*, набор *команд* процессора *МК* включает в себя четыре основные группы *команд*:

- *команды* пересылки данных;
- *арифметические команды* ;
- *логические команды* ;
- *команды переходов*.

Для реализации возможности независимого управления разрядами портов (регистров) в большинстве современных *МК* предусмотрена также группа *команд* битового управления (булевый или битовый процессор). Наличие *команд* битового процессора позволяет существенно сократить объем кода управляющих программ и время их выполнения.

В ряде *МК* выделяют также группу *команд* управления ресурсами контроллера, используемую для настройки режимов работы портов ввода/вывода, управления таймером и т.п. В большинстве современных *МК* внутренние ресурсы контроллера отображаются на *память данных*, поэтому для целей управления ресурсами используются *команды* пересылки данных.

Система команд МК по сравнению с *системой команд* универсального МП имеет, как правило, менее развитые группы арифметических и логических *команд*, зато более мощные группы *команд* пересылки данных и управления. Эта особенность связана со сферой применения *МК*, требующей, прежде всего, контроля окружающей обстановки и формирования управляющих воздействий.

Схема синхронизации МК. Схема синхронизации *МК* обеспечивает формирование сигналов синхронизации, необходимых для выполнения командных циклов центрального процессора, а также обмена информацией по внутренней магистрали. В зависимости от исполнения центрального процессора командный цикл может включать в себя от одного до нескольких (4 — 6) тактов синхронизации. Схема синхронизации формирует также метки времени, необходимые для работы таймеров *МК*. В состав схемы синхронизации входят делители частоты, которые формируют необходимые последовательности синхросигналов.

Память программ и данных МК. В *МК* используется три основных вида памяти. *Память программ* представляет собой постоянную *память* (*ПЗУ*), предназначенную для хранения программного кода (*команд*) и констант. Ее содержимое в ходе выполнения программы не изменяется. *Память данных* предназначена для хранения переменных в процессе выполнения программы и представляет собой *ОЗУ*. Регистры *МК* — этот вид памяти включает в себя внутренние регистры процессора и регистры, которые служат для управления периферийными устройствами (регистры специальных функций).

Память программ. Основным свойством *памяти программ* является ее энергонезависимость, то есть возможность хранения программы при отсутствии питания.

Память данных. *Память данных МК* выполняется, как правило, на основе статического ОЗУ. Термин "статическое" означает, что содержимое ячеек ОЗУ сохраняется при снижении тактовой частоты *МК* до сколь угодно малых значений (с целью снижения *энергопотребления*). Большинство *МК*имеют такой параметр, как "напряжение хранения информации" — $U_{STANDBY}$. При снижении напряжения питания ниже минимально допустимого уровня U_{DDMIN}, но выше уровня $U_{STANDBY}$ работа программы *МК* выполняться не будет, но информация в ОЗУ сохраняется. При восстановлении напряжения питания можно будет сбросить *МК* и продолжить выполнение программы без потери данных. Уровень напряжения хранения составляет обычно около 1 В, что позволяет в случае необходимости перевести МК на питание от автономного источника (батареи)

и сохранить в этом режиме данные ОЗУ.

Объем *памяти данных МК*, как правило, невелик и составляет обычно десятки и сотни байт. Это обстоятельство необходимо учитывать при разработке программ для *МК*. Так, при программировании *МК* константы, если возможно, не хранятся как переменные, а заносятся в ПЗУ программ. Максимально используются аппаратные возможности *МК*, в частности, таймеры. Прикладные программы должны ориентироваться на работу без использования больших массивов данных.

Регистры МК. Как и все МПС, *МК* имеют набор регистров, которые используются для управления его ресурсами. В число этих регистров входят обычно регистры процессора (аккумулятор, регистры состояния, индексные регистры), регистры управления (регистры управления прерываниями, таймером), регистры, обеспечивающие ввод/вывод данных (регистры данных портов, регистры управления параллельным, последовательным или аналоговым вводом/выводом). Обращение к этим регистрам может производиться по-разному.

В *МК* с RISC-процессором все регистры (часто и аккумулятор) располагаются по явно задаваемым адресам. Это обеспечивает более высокую гибкость при работе процессора.

Одним из важных вопросов является размещение регистров в адресном пространстве *МК*. В некоторых *МК* все регистры и *память данных* располагаются в одном адресном пространстве. Это означает, что *память данных* совмещена с регистрами. Такой подход называется "отображением ресурсов *МК* на память".

В других *МК* адресное пространство устройств ввода/вывода отделено от общего пространства памяти. Отдельное пространство ввода/вывода дает некоторое преимущество процессорам с гарвардской архитектурой, обеспечивая возможность считывать *команду* во время обращения к регистру ввода/вывода.

Стек МК. В *микроконтроллерах* ОЗУ данных используется также для организации вызова подпрограмм и обработки прерываний. При этих операциях содержимое программного счетчика и основных регистров (аккумулятор, регистр состояния и другие) сохраняется и затем восстанавливается при возврате к основной программе.

В фон-неймановской архитектуре единая область памяти используется, в том числе, и для реализации *стека*. При этом снижается производительность устройства, так как одновременный доступ к различным видам памяти невозможен. В частности, при выполнении *команды* вызова подпрограммы следующая *команда* выбирается после того, как в *стек* будет помещено содержимое программного счетчика.

В гарвардской архитектуре стековые операции производятся в специально выделенной для этой цели памяти. Это означает, что при выполнении программы вызова подпрограмм процессор с гарвардской архитектурой производит несколько действий одновременно.

Необходимо помнить, что *МК* обеих архитектур имеют ограниченную емкость памяти для хранения данных. Если в процессоре имеется отдельный *стек* и объем записанных в него данных превышает его емкость, то происходит циклическое изменение содержимого указателя *стека*, и он начинает ссылаться на ранее заполненную ячейку *стека*. Это означает, что после слишком большого количества вызовов подпрограмм в *стеке* окажется неправильный адрес возврата. Если *МК* использует общую область памяти для размещения данных и *стека*, то существует опасность, что при переполнении *стека* произойдет запись в область данных либо будет сделана попытка записи загружаемых в *стек* данных в область ПЗУ.

Внешняя память. Несмотря на существующую тенденцию по переходу к закрытой архитектуре *МК*, в некоторых случаях возникает необходимость подключения дополнительной внешней памяти (как *памяти программ*, так и данных).

Если *МК* содержит специальные аппаратные средства для подключения внешней памяти, то эта операция производится штатным способом (как для МП).

Второй, более универсальный, способ заключается в том, чтобы использовать порты ввода/вывода для подключения внешней памяти и реализовать обращение к памяти программными средствами. Такой способ позволяет задействовать простые устройства ввода/вывода без реализации сложных шинных интерфейсов, однако приводит к снижению быстродействия системы при обращении к внешней памяти.

Порты ввода/вывода. Каждый МК имеет некоторое количество линий ввода/вывода, которые объединены в многоразрядные (чаще 8-разрядные) параллельные *порты* ввода/вывода. В памяти МК каждому *порту* ввода/вывода соответствует свой *адрес* регистра данных. Обращение к регистру данных *порта* ввода/вывода производится теми же командами, что и обращение к памяти данных. Кроме того, во многих МК отдельные разряды *портов* могут быть опрошены или установлены командами битового процессора.

В зависимости от реализуемых функций различают следующие типы параллельных *портов*:

- однонаправленные *порты*, предназначенные только для *ввода* или только для *вывода* информации;
- двунаправленные *порты*, направление передачи которых (*ввод* или *вывод*) определяется в процессе инициализации МК;
- *порты* с альтернативной функцией (мультиплексированные *порты*). Отдельные линии этих *портов* используются совместно со встроенными периферийными устройствами МК, такими как *таймеры*, АЦП, контроллеры *последовательных интерфейсов*;
- *порты* с программно управляемой схемотехникой входного/выходного буфера.

Порты выполняют роль устройств временного согласования функционирования МК и объекта управления, которые в общем случае работают асинхронно. Различают три типа алгоритмов обмена информацией между МК и внешним устройством через параллельные *порты* ввода/вывода:

- режим простого программного ввода/вывода;
- режим ввода/вывода со стробированием;
- режим ввода/вывода с полным набором сигналов подтверждения обмена.

Таймеры и процессоры событий. Большинство задач управления, которые реализуются с помощью МК, требуют исполнения их в реальном времени. Под этим понимается способность системы получить информацию о состоянии управляемого объекта, выполнить необходимые расчетные процедуры и выдать управляющие воздействия в течение интервала времени, достаточного для желаемого изменения состояния объекта. Возлагать функции формирования управления в реальном масштабе времени только на центральный процессор неэффективно, так как это занимает ресурсы, необходимые для расчетных процедур. Поэтому в большинстве современных МК используется аппаратная поддержка работы в реальном времени с использованием таймера (таймеров). Модули таймеров служат для приема информации о времени наступления тех или иных событий от внешних датчиков событий, а также для формирования управляющих воздействий во времени.

Модуль прерываний МК. Обработка *прерываний* в МК происходит в соответствии с общими принципами обработки *прерываний* в МПС. *Модуль прерываний* принимает запросы *прерывания* и организует переход к выполнению определенной прерывающей программы. Запросы *прерывания* могут поступать как от внешних источников, так и от источников, расположенных в различных внутренних модулях МК. В качестве входов для приема запросов от внешних источников чаще всего используются *выводы* параллельных *портов* ввода/вывода, для которых эта *функция* является альтернативной. Источниками запросов внешних *прерываний* также могут быть любые изменения внешних сигналов на некоторых специально выделенных линиях *портов* ввода/вывода.

Источниками внутренних запросов *прерываний* могут служить следующие события:

- переполнение *таймеров* / *счетчиков* ;
- сигналы от каналов входного захвата и выходного сравнения *таймеров* / *счетчиков* или от *процессора событий* ;
- готовность памяти EEPROM;
- сигналы *прерывания* от дополнительных модулей МК, включая завершение передачи или приема информации по одному из последовательных *портов* и другие.

Любой *запрос прерывания* поступает на обработку, если *прерывания* в МК разрешены и разрешено *прерывание* по данному запросу. *Адрес*, который загружается в программный *счетчик* при переходе к обработке *прерывания*, называется "*вектор прерывания* ". В

зависимости от организации модуля *прерываний* конкретного МК различные источники *прерываний* могут иметь разные векторы или использовать некоторые из них совместно. Использование различными *прерываниями* одного вектора обычно не вызывает проблем при разработке программного обеспечения, так как аппаратная часть МК фиксирована, а *контроллер* чаще всего выполняет одну-единственную программу.

Вопрос о приоритетах при одновременном поступлении нескольких запросов на *прерывание* решается в различных МК по-разному. Есть МК с одноуровневой системой приоритетов (все запросы равноценны), многоуровневой системой с фиксированными приоритетами и многоуровневой программируемой системой приоритетов.

Отдельно необходимо описать аппаратные *прерывания*, связанные с включением питания, подачей сигнала "сброс" и переполнением сторожевого *таймера*. Они имеют немаскируемый характер и чаще всего разделяют один общий *вектор прерывания*. Это вполне логично, поскольку результатом каждого из событий является начальный сброс МК.