М. В. Марченко

УСТРОЙСТВА НА МИКРОКОНТРОЛЛЕРАХ

Ульяновск
2007

Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

М. В. МАРЧЕНКО

УСТРОЙСТВА НА МИКРОКОНТРОЛЛЕРАХ

Учебное пособие по дисциплине
«Устройства на PIC-процессорах»
для студентов дневной формы обучения специальности 200700
«Радиотехника»

Ульяновск 2007

УДК 004.318 (075)
ББК 34.9я7
М37

Рецензенты: начальник лаборатории УГК ОКБ УМЗ кандидат технических наук В. А. Гульшин;
кафедра «Многоканальная электропроводная и волоконно-оптическая связь» УВВИУС

Утверждено редакционно-издательским советом университета в качестве учебного пособия

М37 **Марченко, М. В.** Устройства на микроконтроллерах : учебное пособие по дисциплине «Устройства на PIC-процессорах» для студентов дневной формы обучения специальности 200700 «Радиотехника». – Ульяновск: УлГТУ, 2007. – 66 с.

ISBN 978-5-9795-0084-3

Пособие составлено в соответствии с программой курса «Устройства на PIC-процессорах».

Рассмотрены основные вопросы дисциплины «Устройства на PIC-процессорах». Пособие содержит теоретический и практический материал и включает в себя следующие вопросы: структура PIC-контроллеров, узлы микроконтроллеров и принципы их организации и работы, описание команд, принципы построения программ. Отдельно рассмотрен пример построения устройства контроля и управления на базе микроконтроллера. В пособие включен список рекомендуемой литературы.

Учебное пособие предназначено для студентов дневной формы обучения специальности 200700 «Радиотехника».

Работа подготовлена на кафедре «Радиотехника».

УДК 004.318 (075)
ББК 34.9я7

ISBN 978-5-9795-0084-3

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Учебное пособие «Устройства на микроконтроллерах» предназначено для студентов, обучающихся по специальности «Радиотехника», изучающих курс «Устройства на PIC-процессорах». В пособии собран материал, полностью отражающий программу курса и позволяющий студентам самостоятельно освоить проектирование устройств на микроконтроллерах семейства PIC.

Пособие состоит из разделов, включающих общие сведения о микроконтроллерах PIC, их внутренней структуре, принципах программирования и схемотехническом построении устройств.

Разделы об общих сведениях и структуре микроконтроллеров содержат информацию о классификации микроконтроллеров, назначении их основных узлов и принципах их работы. Рассматриваются схемы инициализации микроконтроллеров, способы формирования сигнала тактовой частоты, особенности организации внутренней памяти.

Принципы программирования на ассемблере рассмотрены на небольших примерах с подробными комментариями. В примерах показаны методы организации ветвления, циклов, простейших арифметических и логических действий. Кратко рассмотрена программная среда MPLAB, позволяющая выполнять отладку программ для микроконтроллеров PIC.

Разделы, посвящённые схемотехнике устройств на микроконтроллерах, содержат сведения о подключении к микроконтроллерам устройств отображения информации, устройств ввода, сигнальных устройств и датчиков.

Принцип проектирования законченного устройства на микроконтроллере PIC изложен в виде примера: рассматривается устройство контроля давления масла и температуры охлаждающей жидкости двигателя внутреннего сгорания с элементами блокировки двигателя при выходе данных параметров за допустимые пределы. Рассмотрена функциональная схема устройства, алгоритм и блок-схемы его работы, приведена рабочая программа микроконтроллера.

СТРУКТУРА МИКРОКОНТРОЛЛЕРА

Микроконтроллер состоит из трёх структурных частей: ядра, периферийных и специальных модулей. Ядро микроконтроллера обеспечивает обработку данных. Периферийные модули обеспечивают ввод и вывод данных из ядра микроконтроллера. Специальные модули выполняют дополнительное обслуживание ядра микроконтроллера. Структурный состав микроконтроллера изображён на рис. 1.

Рис. 1. Структура микроконтроллера

Рис. 2. Классификация микроконтроллеров

Микроконтроллеры классифицируются по трём признакам: по типу памяти, по диапазону питающего напряжения и по типу корпуса (рис. 2).

Тип памяти микроконтроллера указывается в его маркировке. Для памяти типа EPROM используется символ «С» (например, PIC16C84). Для масочной памяти используется сочетание символов «CR» (например, PIC16CR84). Электрически перепрограммируемая память обозначается символом «F» (например, PIC16F84).

Вид корпуса определяет назначение микроконтроллера. Керамический корпус с окном (для стирания памяти) удобно использовать при разработке устройств на микроконтроллерах. В серийной продукции используется пластмассовый корпус. Бескорпусный вариант микроконтроллера необходим для минимизации устройства.

Разработка устройств на микроконтроллерах включает в себя четыре основных этапа. На первом этапе разрабатывается алгоритм работы устройства и блок-схема алгоритма. На основе блок-схемы разрабатывается текст программы и генерируется объектный код с помощью ассемблера или компилятора, если программа написана на языке высокого уровня. На втором этапе производится отладка программы микроконтроллера. Для этого используется специальное программное обеспечение, позволяющее симулировать работу программы микроконтроллера. По изменению состояния регистров микроконтроллера оценивается корректность работы программы. На третьем этапе осуществляется конструирование и сборка устройства на микроконтроллере. На этом этапе выполняется также запись отлаженной программы в микроконтроллер с помощью программатора. Четвёртый этап заключается в отладке собранного устройства.

Существуют специальные стенды, предназначенные для моделирования проектирования микроконтроллерных устройств. Такие стенды позволяют оперативно изменять программную и аппаратную части всего устройства, обеспечивая тем самым их быструю отладку. Впоследствии микроконтроллер с перепрограммируемой памятью в готовом устройстве может быть заменён микроконтроллером с однократно записываемой памятью. Такой подход позволяет существенно сократить стоимость готового изделия.

Устройства, предполагающие фуикциональное расширение, используют в своём конечном варианте микроконтроллеры с перепрограммируемой памятью.

ГЕНЕРАТОР ТАКТОВЫХ СИГНАЛОВ

Генератор тактовых сигналов (ГТС) предназначен для формирования тактовых сигналов, по которым происходит выполнение инструкций МК. Сигналы ГТС также используются для управления периферийными модулями. Существуют два режима работы ГТС (рис. 3). Первый режим реализуется на базе кварцевого резонатора. Второй режим – на базе RC-цепочки. Кварцевый резонатор позволяет получить высокую стабильность работы ГТС, а значит, и высокую точность работы прибора, построенного на базе микроконтроллера. Генератор на RC-цепочке не обладает высокой стабильностью, но даёт выигрыш в стоимости всего прибора. Для резонаторного генератора существуют три режима работы: низкочастотный (LP), стандартный (XT) и высокочастотный (HS). При высокочастотном режиме ГТС потребляет большее количество энергии, чем в других режимах с кварцевым резонатором. Для построения RC-генератора может использоваться как внешняя RC-цепочка, так и внутренняя. Кроме того, для тактирования микроконтроллера может использоваться внешний генератор. Выбор типа генератора и его режима работы осуществляется с помощью конфигурирующего слова, которое записывается в микроконтроллер при его программировании.

Рис. 3. Режимы работы ГТС

Большинство бытовых микроконтроллерных устройств реализуют функции, не требующие высокой временной точности работы, поэтому в них используется режим работы генератора с встроенной внутренней RC-цепочной.

Внешние кварцевые генераторы применяются в устройствах, предназначенных для измерения каких-либо величин, поскольку в этом случае, как правило, предъявляются высокие требования к стабильности работы задающего генератора. Значение измеряемой величины в цифровых устройствах часто определяется подсчётом количества импульсов за фиксированный промежуток времени. Так как точность промежутка времени, в течение которого происходит подсчёт импульсов, связана со стабильностью задающего генератора, то точность измерения прибора и будет определяться точностью работы генератора.

ВНЕШНИЕ ЦЕПИ ГЕНЕРАТОРОВ

Рассмотрим внешние цепи при работе с кварцевым резонатором. На рис. 4 приведена стандартная схема подключения кварцевого резонатора к микроконтроллеру. Для уверенного запуска генератора значения ёмкостей конденсаторов C1 и C2 должны незначительно отличаться друг от друга. В общем случае значения ёмкостей этих конденсаторов составляют 15...30 пФ. Резистор R1 предназначен для предотвращения самовозбуждения генератора на низкой частоте. Сопротивление резистора R1 находится в пределах 10 кОм.

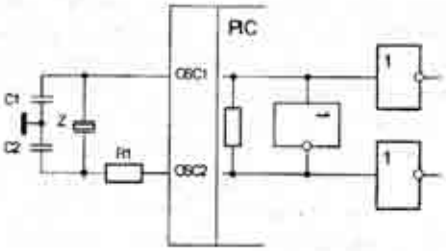

Рис. 4. Подключение кварцевого резонатора к микроконтроллеру

В случае использования внешнего генератора его выход подключается ко входу OSC1 микроконтроллера через буферный элемент (рис. 5). Неиспользуемый вывод OSC2 через резистор сопротивлением 1...2 кОм соединяется с «землёй».

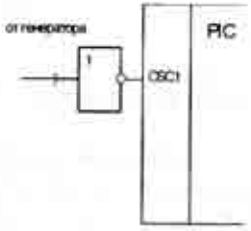

Рис. 5. Подключение внешнего генератора к микроконтроллеру

При работе ГТС с внутренней RC-цепью внешние цепи к микроконтроллеру не подключаются. При внешней RC-цепи подключение изображено на рис. 6.

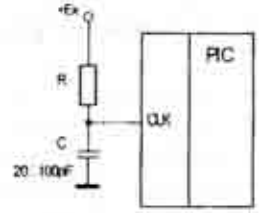

Рис. 6. Подключение внешней RC-цепи к микроконтроллеру

Критерии выбора типа генератора отражены в табл. 1. При необходимости максимально удешевить стоимость прибора или при отсутствии потребности в точной работе прибора следует выбирать RC-генератор. При необходимости построения высокоточного прибора необходимо использовать кварцевый генератор.

Таблица 1

Критерии выбора типа генератора

Тип генератора	На кварцевом резонаторе	RC-генератор
Достоинства	Высокая точность Стабильность работы Пониженное энергопотребление	Низкая стоимость
Недостатки	Высокая цена	Низкая стабильность Увеличенное энергопотребление

При выборе типа генератора микроконтроллерного устройства следует руководствоваться, прежде всего, его назначением. Как было указано выше, в измерительных приборах предпочтительно использовать кварцевые генераторы, в системах контроля и сбора информации в большинстве случаев достаточно точности встроенного RC-генератора.

ВНЕШНИЕ КВАРЦЕВЫЕ ГЕНЕРАТОРЫ

Внешние генераторы используются в случае необходимости синхронизировать работу микроконтроллера с другими внешними устройствами. Во внешних кварцевых резонаторах может использоваться параллельный и последовательный резонанс. Схема генератора с параллельным резонансом приведена на рис. 7. Генератор выполнен на логическом элементе D1, который с помощью делителя на резисторах R1 и R2 и резистора обратной связи R3 переведён в линейный режим работы. Частота генератора определяется кварцевым резонатором и ёмкостями C1 и C2. К выходу генератора подключается микроконтроллер через буферный элемент D2. Буферный элемент обеспечивает стабильную работу генератора, снижая зависимость его выходных параметров от нагрузки. Значения сопротивлений R3 ≈ 4...5 кОм, R4 ≈ 10 кОм, ёмкостей C1 и C2 ≈ 15…30 пФ.

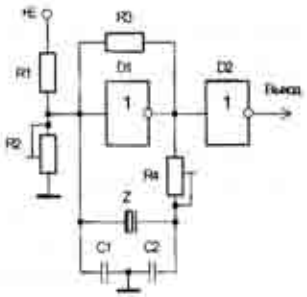

Рис. 7. Схема генератора с параллельным резонансом

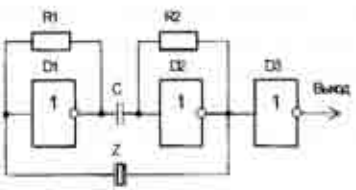

Рис. 8. Схема генератора с последовательным резонансом

На рис. 8 приведена схема генератора с последовательным резонансом. Сопротивления обратной связи R1 и R2 переводят логические элементы D1 и D2 в линейный режим работы. Конденсатор C выполняет разделительную функцию. Последовательно включенные логические элементы, работающие в линейном режиме, образуют усилитель с суммарным сдвигом фазы в 360°. Кварцевый резонатор включен в цепь положительной обратной связи. Так как на частоте последовательного резонанса сопротивление контура минимальное, то на этой частоте и происходит возбуждение генератора. Как

и в предыдущей схеме, для повышения стабильности работы генератора при его подключении к микроконтроллеру используется буферный элемент D3.

Для увеличения выходной мощности генераторов можно использовать буферные элементы с открытым коллектором.

При монтаже тактового генератора необходимо обеспечить близость расположения всех элементов его схемы с целью минимизировать наводки на другие узлы устройства.

Внешние кварцевые генераторы в современных миниатюрных микроконтроллерных устройствах используются относительно редко ввиду того, что для их построения требуются дополнительные элементы, что сказывается на размерах готового изделия.

Использование внешних генераторов имеет смысл в случае организации синхронной работы нескольких микроконтроллеров или при подключении внешних систем обмена информацией, тактируемых от одного источника.

Поскольку память микроконтроллеров является статической, то частота работы микроконтроллера определяет только скорость выполнения им инструкций и не влияет на его функциональные возможности. На практике тактовая частота может находиться в диапазоне от десятка килогерц до десятка мегагерц.

Выбор конкретного значения тактовой частоты зависит от назначения микроконтроллерного устройства и его скорости работы.

СХЕМА СБРОСА МИКРОКОНТРОЛЛЕРА

Схема сброса предназначена для начальной инициализации микроконтроллера. После выполнения операции сброса микроконтроллер начинает исполнять инструкции с адреса 0000h.

Сброс может осуществляться тремя способами. Сброс по питанию происходит при подаче питающего напряжения на микроконтроллер. Сброс может происходить по внешним сигналам. Также сброс происходит при возникновении внутренних ошибок микроконтроллера. Структура схемы сброса приведена на рис. 9. Логика формирования сигнала «сброс» обрабатывает события от трёх источников. Первый источник V_{DD} – линия питающего напряжения, второй источник $\overline{MCLR}$ – линия внешнего сброса, третий источник – внутренняя шина управления микроконтроллера. Синхронизация схемы сброса осуществляется по сигналу тактовой частоты OSC.

Рис. 9. Сигналы, формирующие сброс микроконтроллера

Стандартная внешняя цепь схемы сброса приведена на рис. 10. При появлении напряжения питания происходит запуск работы микроконтроллера. Недостатком данной цепи является наличие высокой вероятности неустойчивой работы микроконтроллера при включении питания, так как коммутация питания обычно сопровождается импульсными помехами, которые могут быть восприняты микроконтроллером как дополнительные сигналы сброса.

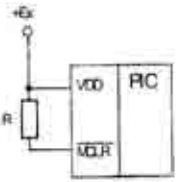

Рис. 10. Стандартная внешняя цепь схемы сброса

На рис. 11 показана усовершенствованная внешняя цепь схемы сброса микроконтроллера, которая лишена описанного выше недостатка. RC-цепь предназначена для задержки запуска микроконтроллера после подачи на него напряжения питания. Время задержки определяется ёмкостью конденсатора. Диод предназначен для быстрой разрядки конденсатора при исчезновении напряжения питания.

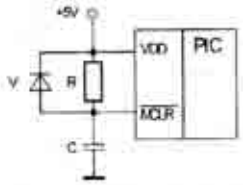

Рис. 11. Усовершенствованная стандартная внешняя цепь схемы сброса

При необходимости обеспечить более надёжный запуск микроконтроллера по уровню питающего напряжения используются более сложные внешние цепи. На рис. 12 изображена схема сброса по уровню питающего напряжения, где в качестве элемента, определяющего пороговое напряжение, используется стабилитрон. Сброс происходит, если уровень питающего напряжения E опускается ниже (Vст+0,7 В).

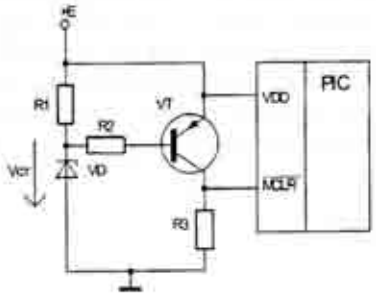

Рис. 12. Устройство формирования сигнала сброса

Упрощённый вариант описанной выше схемы изображён на рис. 13. Здесь пороговое напряжение определяется значением падения напряжения на переходе эмиттер-база ключевого транзистора. Таким образом сброс микроконтроллера происходит при выполнении условия $E \cdot R1 / (R1+R2) < 0{,}7$ В.

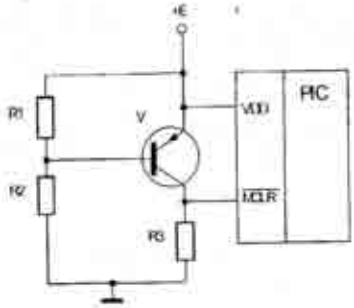

Рис. 13. Внешняя цепь схемы сброса

После исполнения процедуры сброса микроконтроллера информация об источнике сброса записывается в регистр состояния STATUS.

В большинстве случаев для начальной инициализации микроконтроллерных устройств вполне достаточно цепи, изображённой на рис. 11. Для ручного сброса в этой цепи параллельно конденсатору С подключают кнопку с нормально разомкнутыми контактами. При замыкании контактов ёмкость быстро разряжается и на входе микроконтроллера появляется низкий уровень, в результате чего происходит инициализация. После отпускания кнопки цепь работает как при включении питания.

Если предполагается работа микроконтроллера в системе обмена информацией, то для инициализации сигнал может подаваться от внешних устройств. В этом случае также может использоваться схема, изображённая на рис. 11. Линия внешнего сброса подключается к входу MCLR. Длительность отрицательного импульса сброса будет определяться ёмкостью конденсатора С. Величина сопротивления R не должна быть менее 1 кОм.

Существуют два направления построения процессорных систем, которые определяются способами использования памяти. Первое направление подразумевает использование общей памяти и для кода программы, и для данных (рис. 14). В этом случае код и данные извлекаются из памяти процессором последовательно друг за другом. Соответственно и выполнение инструкций процессора происходит последовательно друг за другом. Архитектура таких систем носит название *Фон-неймановской*. Второе направление подразумевает разделение памяти на память кода или память программ и память данных (рис. 14). В этом случае обращение процессора к памяти программ и памяти данных может происходить одновременно. Процессор может исполнять текущую инструкцию и одновременно выбирать из памяти код следующей инструкции. В результате этого удаётся повысить производительность системы. Архитектура процессорных систем с разделением памяти получила название *гарвардской*, а процессоры в рамках данной архитектуры – *RISC-процессоры*. Микроконтроллеры строятся по гарвардской архитектуре.

Рис. 14. Архитектура построения процессорных систем

Рис. 15. Конвейерная обработка инструкций

В гарвардской архитектуре организован принцип конвейерной обработки инструкций процессора (рис. 15). В один машинный цикл происходит выборка текущей инструкции и исполнение предыдущей. При традиционной архитектуре системы эти действия происходят

последовательно, поэтому, при прочих равных условиях, RISC-процессоры позволяют проводить обработку данных быстрее.

Архитектура фон Неймана в настоящее время используется для упрощения вычислительных систем. Такая архитектура характеризуется сложностью изменения рабочей программы микропроцессорного устройства. Перепрограммирование устройств фон-неймановской архитектуры может быть связано с внесением различных изменений в аппаратную часть.

Математическая модель, описывающая работу таких вычислительных систем, называется машиной фон Неймана (предложена фон Нейманом в начале 20 века). Согласно принципам машины фон Неймана вычислительная система должна удовлетворять трём основным принципам: система должна иметь оперативную память, в которой хранятся данные и управляющая программа; алгоритм работы системы определяется управляющей программой; код программы хранится вместе с данными, выполнение кода программы происходит последовательно, согласно его линейному расположению в оперативной памяти.

Поскольку скорость работы вычислительных систем с архитектурой фон Неймана была сравнительно низкой, проводились исследования способов её увеличения. В конце 30-х годов 20 века в Гарвардском университете Говардом Эйкеном была предложена новая архитектура вычислительной системы, которая получила название гарвардской. Главная идея предложенной архитектуры была в оптимизации времени исполнения вычислительных операций и работы памяти системы. Физически это выражалось в разделении памяти на память команд и память данных, что приводило к увеличению стоимости устройства, так как для подключения к памяти к процессору требовалось в два раза большее количество линий.

Для решения возникшей проблемы стали использовать принцип мультиплексирования внешних шин данных и адреса, которые в процессоре разделялись на шины данных и адреса для памяти данных и памяти команд соответственно (модифицированная гарвардская архитектура). Дальнейшее совершенствование гарвардской архитектуры привело к появлению внутренней памяти процессора (кэш-память), которая используется для хранения инструкций, в результате чего данные могут передаваться одновременно по обеим шинам (расширенная гарвардская архитектура).

СТРУКТУРНАЯ СХЕМА МИКРОКОНТРОЛЛЕРА

Структурная схема микроконтроллера приведена на рис. 16. Все арифметико-логические операции выполняются в арифметико-логическом устройстве АЛУ. Основным рабочим регистром при проведении операций является аккумулятор W. Результаты выполнения операций могут сохраняться в аккумуляторе или памяти данных, при этом в зависимости от результата может изменяться регистр состояния STATUS.

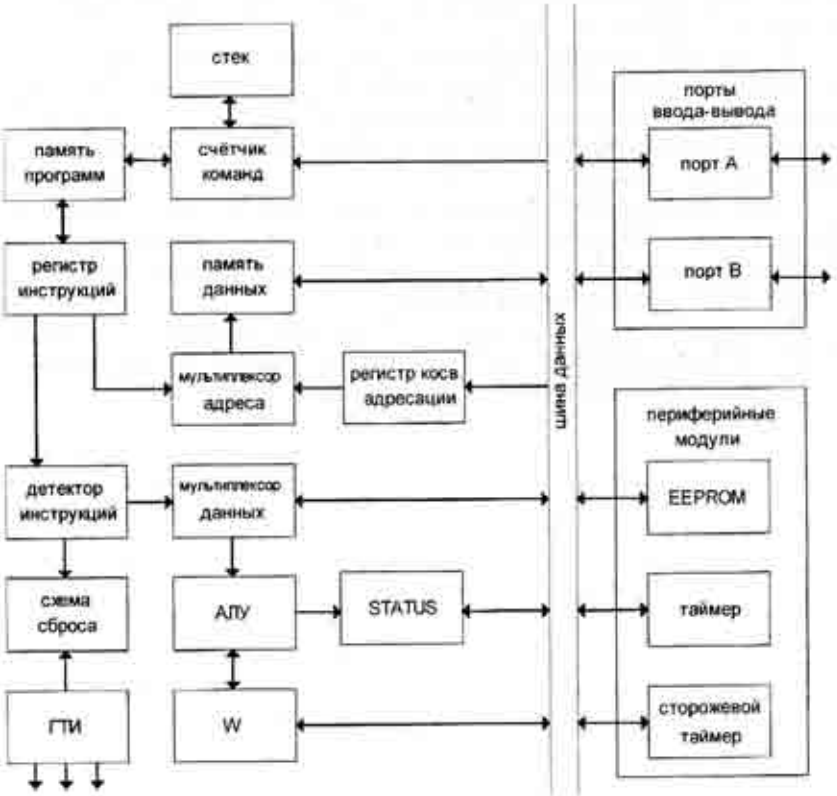

Рис. 16. Структурная схема микроконтроллера

Выборка инструкции из памяти программ для исполнения осуществляется по счётчику команд. Команда записывается в регистр инструкций, где из неё выделяется адресная часть, указывающая на память данных, и собственно инструкция для выполнения. Инструкция управляет работой АЛУ и указывает место расположения исходных и конечных данных. Адрес в памяти данных может указываться как непосредственно в

команде, так и косвенно через регистр косвенной адресации. При переходе в подпрограмму в стек из счётчика команд записывается адрес возврата, а в счётчик команд помещается адрес начала подпрограммы. При завершении работы подпрограммы в счётчик возвращается адрес из стека. Допустимое количество вложенных подпрограмм определяется размером памяти стека. В случае переполнения стека адрес точки основной программы, из которой была запущена первая подпрограмма, теряется.

Данные для обработки в АЛУ или для записи/чтения в память данных могут быть получены от портов ввода-вывода или периферийных модулей. Каждое такое устройство отображается в адресном пространстве одной или несколькими ячейками памяти.

В некоторых микроконтроллерах часть периферийных устройств не может работать одновременно. При программировании пользователь должен выбрать необходимый модуль и определить его в служебной информации (обычно в регистре OPTION). В этом случае определяются функции внешних линий микроконтроллера.

Часть периферийных модулей не имеют линий связи с внешними устройствами и предназначены для формирования внутренних сигналов микроконтроллера. Работа таких модулей также определяется программно через регистры специального назначения.

Периферийные модули могут формировать сигналы прерывания, согласно которым основная программа должна остановить свою работу и передать управление подпрограмме обработки прерываний.

ЦЕНТРАЛЬНОЕ ПРОЦЕССОРНОЕ УСТРОЙСТВО

Центральное процессорное устройство (ЦПУ) предназначено для выборки команд из памяти и их последующей обработки. Архитектура микроконтроллера позволяет использовать малое число команд, причём все команды являются ортогональными, то есть любая команда может оперировать с любым регистром памяти. Все команды микроконтроллера подразделяются на три типа.

Первый тип команд – команды для проведения операций над регистрами (рис. 17). Команды этого типа могут быть бит- и байт-ориентированными. Один операнд всегда находится в аккумуляторе, второй – в памяти данных. Результат может сохраняться либо в аккумуляторе, либо в памяти данных в зависимости от значения поля «b/d».

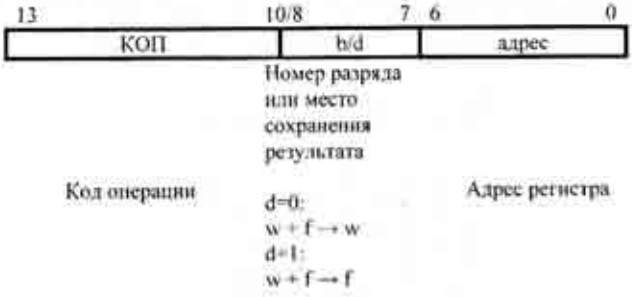

Рис. 17. Формат инструкции для работы с регистрами

Второй тип команд – команды для работы с константами. Константа располагается непосредственно в инструкции в поле «константа» (рис. 18). Результат выполнения команды записывается в аккумулятор.

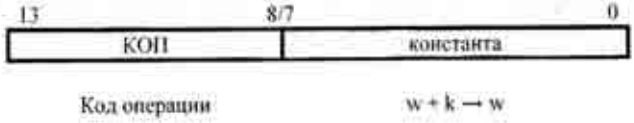

Рис. 18. Формат инструкции для работы с константами

Третий тип команд – команды перехода. Эти команды используются для перехода в подпрограмму, возвращения из подпрограммы и организации ветвления (рис. 19).

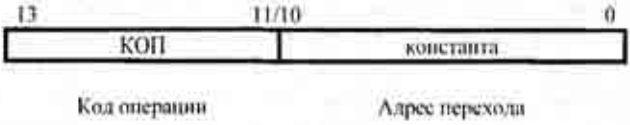

Рис. 19. Формат инструкций переходов

Выполнение команд осуществляется в четыре этапа (рис. 20). На первом этапе происходит выборка инструкции из памяти программ. Далее происходит выборка данных из памяти данных или пустая операция, если таковые данные отсутствуют. На третьем этапе производится обработка данных. На четвёртом этапе осуществляется запись результата в память данных или пустая операция, если результат не предполагается.

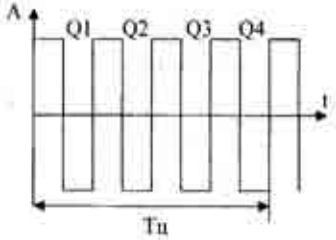

Рис. 20. Этапы выполнения команды

Таким образом, на выполнение одной команды требуется время,равное длительности четырёх периодов тактовой частоты. Если частота тактового генератора составляет 4 МГц, то на исполнение одной команды требуется 1 мкс.

ОРГАНИЗАЦИЯ ПАМЯТИ

Память микроконтроллера делится на два вида: память программ и память данных (рис. 21). Память данных включает в себя регистры специального назначения и регистры общего назначения. Регистры специального назначения располагаются в зарезервированном адресном пространстве, в котором находятся порты ввода-вывода, таймеры и прочие модули. Регистры специального назначения используются для работы со встроенными в микроконтроллер модулями. Регистры общего назначения представляют собой обычные ячейки памяти, которые могут использоваться для временного хранения данных.

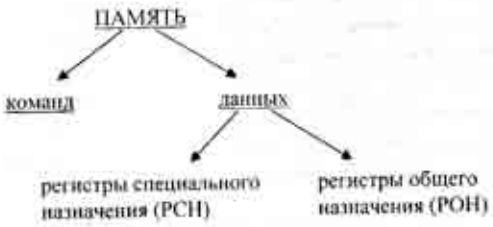

Рис. 21. Память микроконтроллера

Память данных состоит из четырёх страниц (рис. 22). Верхняя часть адресного пространства выделена для регистров специального назначения. Адресация к регистрам общего назначения может производиться независимо от текущей страницы. Выбор страницы памяти происходит после установки соответствующих разрядов в регистре состояния STATUS.

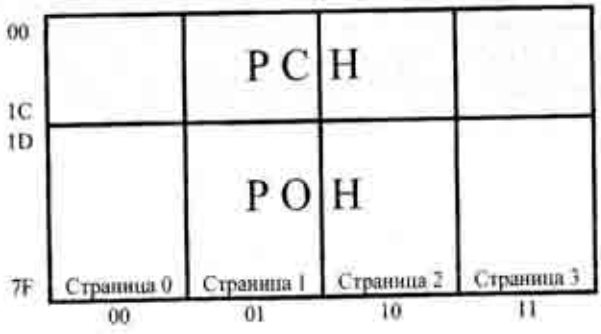

Рис. 22. Организация памяти данных

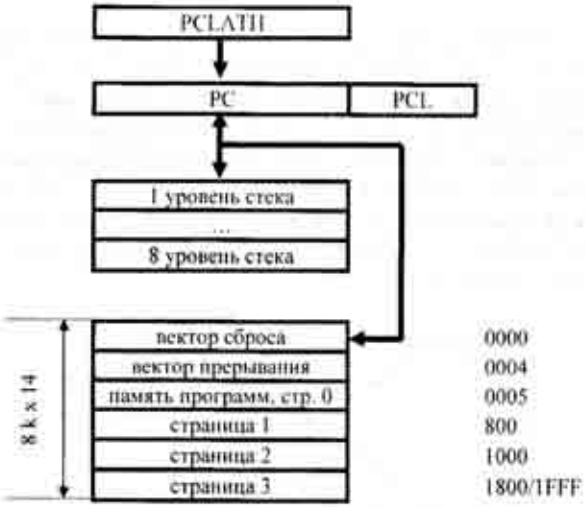

Рис. 23. Организация памяти программ

Память программ (рис. 23) адресуется командным счётчиком PC, который состоит из двух регистров: PCL и PCLATH. Оба регистра расположены в адресном пространстве памяти данных. Счётчик может загружаться через шину данных или из стека, который может содержать до восьми адресов. В первой странице памяти команд расположены векторы сброса и прерывания.

Доступ к регистрам электрически перепрограммируемой памяти (EEPROM) осуществляется с помощью дополнительных регистров. Адрес ячейки заносится в регистр EEADR, содержимое для считывания или записи – в регистр EEDATA. Управление чтением и записью в EEPROM осуществляется с помощью регистров EECON1 и EECON2. Объём памяти EEPROM составляет 64 байта.

РЕГИСТР СОСТОЯНИЯ STATUS

Регистр состояния включает восемь разрядов. Старшие разряды указывают на страницы памяти. Остальные разряды выполняют функцию флагов результата совершённой арифметической или логической операции (рис. 24). К таковым относятся: нулевой результат, перенос из старшего разряда и десятичная коррекция. Четвёртый и пятый разряды указывают на срабатывание сторожевого таймера и включение питания микроконтроллера.

R/W	R/W	R/W	RO	RO	R/W	R/W	R/W
IRP	RP1	RP0	-TO	-PD	Z	DC	C
Страница памяти (косв. адресация)	Страница памяти (непоср. адресация)		Флаг сторож. таймера	Флаг вкл. питания	Флаг нулевого результата	Флаг десятичн. переноса	Флаг переноса

Рис. 24. Регистр состояния STATUS

ПРЕРЫВАНИЯ

Прерывания служат для инициализации выполнения определённого кода при возникновении некоторых событий. Источником прерывания могут быть

1) внешний источник;
2) таймер;
3) периферийные модули;
4) порты ввода-вывода.

Информация об источнике прерывания записывается в регистр специального назначения PIR. Управление прерываниями, их запрет и разрешение осуществляется через регистр INCON.

ПОРТЫ ВВОДА-ВЫВОДА

Порты ввода-вывода состоят из двух регистров. Первый регистр TRIS служит для управления работой порта на ввод или вывод. Второй регистр PORT предназначен непосредственно для записи или чтения информации. Например, в микроконтроллере 16F84 два порта A и B. В порту A используется пять разрядов, в порту B используется восемь разрядов.

КОМАНДЫ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА PIC16

Все команды разбиты на три группы. Первая группа ориентирована для работы с байтами, вторая – с битами. В третью группу вошли команды ветвления и вызова подпрограмм.

БАЙТ-ОРИЕНТИРОВАННЫЕ КОМАНДЫ

Команды сложения.

1. Сложение значений аккумулятора W и байта из ячейки памяти с адресом f. Мнемоника команды ADDWF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флагов заёма C, десятичного заёма DC и нулевого результата Z.
2. Вычитание значения аккумулятора W из значения регистра, расположенного в ячейке памяти с адресом f. Мнемоника команды SUBWF f,d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флагов заёма C, десятичного заёма DC и нулевого результата Z.
3. Сложение значений аккумулятора W и байта константы k. Мнемоника команды ADDLW k. Результат сохраняется в аккумуляторе W. Выполнение команды может привести к изменению значений флагов заёма C, десятичного заёма DC и нулевого результата Z.
4. Вычитание значения аккумулятора W из значения константы k. Мнемоника команды SUBLW k. Выполнение команды может привести к изменению значений флагов заёма C, десятичного заёма DC и нулевого результата Z.
5. Декремент на единицу содержимого регистра с адресом f производится по команде DECF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флага нулевого результата Z.
6. Инкремент на единицу содержимого регистра с адресом f производится по команде INCF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флага нулевого результата Z.
7. Декремент содержимого регистра с адресом f на единицу с пропуском. Мнемоника команды DECFSZ f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. В случае ненулевого результата исполняется следующая команда. При нулевом результате вместо следующей команды выполняется виртуальная команда NOP, то есть происходит пропуск исполнения следующей команды.
8. Инкремент содержимого регистра с адресом f на единицу с пропуском. Мнемоника команды INCFSZ f, d. Если d=0, то результат сохраняется в

аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. В случае ненулевого результата исполняется следующая команда. При нулевом результате вместо следующей команды выполняется виртуальная команда NOP, то есть происходит пропуск исполнения следующей команды.

Команды пересылки данных.

1. Содержимое регистра с адресом f пересылается в аккумулятор или обратно по своему адресу. Мнемоника команды MOVF f,d . Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. При нулевом значении содержимого устанавливается флаг нулевого результата Z.
2. Пересылка данных из аккумулятора в ячейку с адресом f. Мнемоника команды MOVWF f. Команда не оказывает влияние на флаги.
3. Пересылка в аккумулятор W константы k. Мнемоника команды MOVLW k. Неиспользуемые биты принимаются равными нулю. Команда не оказывает влияние на флаги.
4. Взаимная смена мест старшего и младшего полубайтов в регистре по адресу f. Мнемоника команды SWAPF f,d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f.
5. Пустая команда NOP используется для увеличения значения программного счётчика на единицу, затрачивая на это один машинный цикл.

Команды логических операций.

1. Побитная логическая операция «И» над содержимым аккумулятора W и содержимым ячейки памяти с адресом f. Мнемоника команды ANDWF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флага нулевого результата Z.
2. Побитная логическая операция «ИЛИ» над содержимым аккумулятора W и содержимым ячейки памяти с адресом f. Мнемоника команды IORWF f,d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флага нулевого результата Z.
3. Побитная логическая операция «Исключающее ИЛИ» над содержимым аккумулятора W и содержимым ячейки памяти с адресом f. М немоника команды XORWF f,d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флага нулевого результата Z.
4. Побитная логическая операция «И» над содержимым аккумулятора W и константой k. Мнемоника команды ANDLW k. Результат сохраняется в аккумуляторе W. Выполнение команды может привести к изменению значения флага нулевого результата Z.
5. Побитная логическая операция «ИЛИ» над содержимым аккумулятора W и константой k. Мнемоника команды IORLW k. Результат сохраняется в

аккумуляторе W. Выполнение команды может привести к изменению значения флага нулевого результата Z.

6. Побитная логическая операция «Исключающее ИЛИ» над содержимым аккумулятора W и константой k. Мнемоника команды XORLW k. Результат сохраняется в аккумуляторе W. Выполнение команды может привести к изменению значения флага нулевого результата Z.

7. Для очистки содержимого регистра с адресом f используется команда CLRF f. После выполнения данной команды устанавливается флаг нулевого результата Z.

8. Для очистки содержимого аккумулятора W используется команда CLRW. После выполнения данной команды устанавливается флаг нулевого результата Z.

9. Инверсия всех разрядов в регистре с адресом f производится по команде COMF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Выполнение команды может привести к изменению значений флага нулевого результата Z.

10. Команда циклического сдвига содержимого регистра с адресом f влево через флаг «переноса» C регистра состояния STATUS. Мнемоника команды RLF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Может измениться состояние флага «переноса» C.

11. Команда циклического сдвига содержимого регистра с адресом f вправо через флаг «переноса» C регистра состояния STATUS. Мнемоника команды RRF f, d. Если d=0, то результат сохраняется в аккумуляторе W, если d=1, то результат сохраняется в регистре с адресом f. Может измениться состояние флага «переноса» C.

БИТ-ОРИЕНТИРОВАННЫЕ КОМАНДЫ

1. Сбросить в 0 бит с номером b в регистре по адресу f. Мнемоника команды BCF f,b. Команда не влияет на флаги.

2. Установить в 1 бит с номером b в регистре по адресу f. Мнемоника команды BSF f,b. Команда не влияет на флаги.

3. Условная проверка бита с номером b в регистре по адресу f. Если значение бита нулевое, следующая команда пропускается. Если значение ненулевое, то выполнение программы продолжается в обычном режиме. Мнемоника команды BTFSC f,b.

4. Условная проверка бита с номером b в регистре по адресу f. Если значение бита ненулевое, следующая команда пропускается. Если значение нулевое, то выполнение программы продолжается в обычном режиме. Мнемоника команды BTFSS f,b.

СПЕЦИАЛЬНЫЕ КОМАНДЫ УПРАВЛЕНИЯ

1. Команда безусловного перехода. Мнемоника команды GOTO addr. При выполнении перехода в счётчик команд загружается адрес addr.
2. Команда перехода к подпрограмме. Мнемоника команды CALL addr. При выполнении перехода адрес следующей команды записывается в вершину стека. В счётчик команд загружается адрес addr подпрограммы.
3. Команда возврата из подпрограммы. Мнемоника команды RETURN. При выполнении перехода адрес следующей команды загружается из вершины стека в счётчик команд.
4. Команда возврата из подпрограммы с загрузкой в аккумулятор константы k. Мнемоника команды RETLW k. В аккумулятор записывается значение константы k, после чего адрес следующей команды загружается из вершины стека в счётчик команд.
5. Команда возврата из подпрограммы после обработки прерывания. Мнемоника команды RETFIE. Прерывания разрешаются, после чего адрес следующей команды загружается из вершины стека в счётчик команд.
6. Команда сброса сторожевого таймера. Мнемоника команды CLRWDT. Изменяются флаги TO и PD в регистре состояния.
7. Команда перехода в спящий режим. Мнемоника команды SLEEP. Микроконтроллер переводится в спящий режим, тактовый генератор отключается.

РЕГИСТРЫ МИКРОКОНТРОЛЛЕРА

Память микроконтроллера представлена набором регистров, часть из которых служит для выполнения специальных функций, остальные предназначены для временного хранения информации.

Регистр состояния STATUS. Регистр состояния содержит в своём составе флаги переноса, десятичного переноса и нуля. Кроме того, в регистре состояния содержатся биты страничной адресации к памяти и биты режима включения.

Регистр программного счётчика PCL. Программный счётчик содержит адрес текущей выполняемой команды в памяти команд. После начала выполнения команды, содержание регистра увеличивается на единицу и происходит выборка следующей команды, если предыдущая команда не предусматривает принудительную установку адреса следующей команды.

Регистр косвенной адресации и регистр выбора FSR. Регистры используются для косвенного указания адреса одного из 64 возможных регистров. Фактически косвенно адресоваться могут 36 регистров общего назначения и 15 регистров специального назначения.

Регистр таймера TMR0. Обращение к регистру таймера может производиться как к обычному регистру. Значение регистра может увеличиваться по внешним импульсам, подаваемым на вывод RTCC. Также увеличение содержимого регистра может происходить по импульсам тактовой частоты. Коэффициент деления при счёте задаётся программным способом. Таймер используется для подсчёта числа внешних событий и измерения отрезков времени.

Регистры параллельных портов PORTA и PORTB и соответствующие регистры управления портами TRISA и TRISB. В порте А пять разрядов PA4-PA0 могут быть независимо друг от друга запрограммированы на ввод и вывод. Порт В состоит из 8 разрядов PB7-PB0. Программирование разряда порта на ввод производится записью логической единицы в соответствующий разряд регистра управления портом. Порт в режиме вывода представляет собой буфер, информация в котором защёлкивается во время записи. При работе порта на ввод каждый разряд отражает текущее состояние соответствующей линии порта.

Регистры доступа к электрически перепрограммируемой памяти EEDATA и EEADR и регистры управления перепрограммируемой памятью EECON1 и EECON2. Размер встроенной перепрограммируемой памяти составляет 64 байта. Чтение и запись данных в память осуществляется через регистр EEDATA . Адрес регистра памяти предварительно записывается в регистр EEADR. Режим записи или чтения выбирается через регистр EECON1. Контроль записи осуществляется через регистр EECON2.

Пример программы, работающей с энергонезависимой памятью, приведён ниже. Подпрограмма чтения считывает данные из EEPROM в аккумулятор.

```
EEPROMRD                ; Подпрограмма чтения
                        ; из энергонезависимой памяти.
    BCF STATUS,RP0      ; Выбрать нулевую страницу банка.
    MOVLW CONFIG_ADDR   ; Загрузить в аккумулятор адрес
                        ; регистра EEPROM, по которому будет
                        ; выполняться чтение.
    MOVWF EEADR         ; Записать адрес EEPROM, по которому будет
                        ; выполняться чтение, в регистр адреса.
    BSF STATUS,RP0      ; Выбрать первую страницу памяти.
    BSF EECON1,RD       ; Перевести EEPROM в режим чтения.
    BCF STATUS,RP0      ; Выбрать нулевую страницу банка.
    MOVF EEDATA,W       ; Считать данные из EEPROM в аккумулятор.
```

Подпрограмма записи сохраняет данные в EEPROM из аккумулятора. Адрес и данные заносятся в соответствующие регистры, после чего производится операция записи. Длительность записи составляет около 10 мс. Перед записью выполняется обязательная запись в управляющий регистр EECON2 байта 55h и байта AAh. Во время процесса записи прерывания должны быть запрещены. Окончание записи определяется по обнулению бита WR. При завершении записи также устанавливается флаг EEIF.

```
EEPROMWR                ; Подпрограмма записи
                        ; в энергонезависимую память.
    BSF STATUS,RP0      ; Выбрать первую страницу памяти.
    BCF INTCON,GIE      ; Запретить прерывания.
    MOVLW 55h           ; Загрузить в аккумулятор число 55h.
    MOVWF EECON2        ; Записать число 55h в управляющий регистр.
    MOVLW AAh           ; Загрузить в аккумулятор число AAh
    MOVWF EECON2        ; Записать число AAh в управляющий регистр.
    BSF EECON1,WR       ; Инициировать запись.
LOOPWR                  ; Цикл ожидания конца записи.
    BTFSC  EECON1,WR    ; Проверять завершение записи
    GOTO   LOOPWR       ; до её окончания.
    BSF INTCON,GIE      ; Разрешить обработку прерываний.
```

Регистр аккумулятора W. Регистр аккумулятора используется для выполнения большинства логических и арифметических операций над данными.

Регистр прерываний INTCON. Регистр прерываний осуществляет управление прерываниями. Часть разрядов регистра предназначена для разрешения прерываний от заданных источников. Другая часть разрядов отведена под флаги прерываний.

Регистр режимов OPTION. Регистр режимов определяет источник сигнала для таймера, коэффициент деления таймера и позволяет подключать нагрузочные сопротивления ко входам порта B, работающего на ввод.

Сторожевой таймер WDT. Сторожевой таймер осуществляет защиту микроконтроллера от сбоев в работе программы. Таймер запрограммирован на некоторый критический отрезок времени, в течение которого он обязательно должен получить команду сброса CLRWDT. При зацикливании программы такая команда получена не будет и таймер сформирует сигнал сброса микроконтроллера. Таймер имеет свой собственный RC-генератор, поэтому его работа не зависит от наличия тактового сигнала микроконтроллера. К сторожевому таймеру можно подключать делитель частоты, увеличивая тем самым время его счёта. Сторожевой таймер также может использоваться для подсчёта времени.

Режим работы микроконтроллера с пониженным энергопотреблением. Данный режим инициируется по команде SLEEP. Обработка команд в этом режиме прекращается, а состояние всех регистров сохраняется. Ток потребления микроконтроллера в режиме пониженного энергопотребления не превышает 1 мкА. Выход из данного режима производится по внешнему событию или по концу счёта сторожевого таймера.

Регистры общего назначения. Регистры общего назначения являются ячейками ОЗУ и могут использоваться для временного хранения данных программы. Ячейки ОЗУ являются статическими и не требуют сигналов регенерации. Количество регистров общего назначения составляет 36.

ПРИЁМЫ ПРОГРАММИРОВАНИЯ

В данном разделе будут рассмотрены способы построения основных элементов алгоритмов с помощью команд микроконтроллера. Примеры программ ориентированы на микроконтроллер PIC16F84.

Команда NOP выполняет один холостой цикл. Состояние никаких регистров при этом не изменяется. Команда используется для организации временной задержки при необходимости синхронизировать работу микроконтроллера с внешним устройством.

Команда загрузки константы в аккумулятор MOVLW k. После выполнения данной команды в аккумуляторе содержится константа k. Пример работы:

```
MOVLW h'15'     ; Загрузить в аккумулятор число 00010101.
                ; После выполнения данной команды
                ; в аккумуляторе будет содержаться
                ; число 00010101.
```

Команды пересылки данных между регистрами MOVWF f и MOVF f,d. Команда MOVWF f загружает данные в регистр f из аккумулятора. Команда MOVF f,d загружает данные из регистра f в аккумулятор или обратно в исходный регистр f в зависимости от состояния флага d. При перемещении содержимого регистра из регистра обратно в регистр флаг ZERO изменяется в соответствии с содержимым регистра. Пример работы:

```
MOVLW h'15'     ; Загрузить в аккумулятор число 00010101.
MOVWF CNTR      ; Загрузить в регистр CNTR содержимое
                ; аккумулятора. После выполнения данной
                ; команды в аккумуляторе будет
                ; содержаться число 00010101.
INCF CNTR,1     ; Инкрементировать содержимое CNTR.
                ; После выполнения команды в регистре
                ; CNTR содержится число 00010110.
MOVF CNTR,0     ; Загрузить в аккумулятор содержимое
                ; регистра CNTR.
                ; После выполнения данной команды
                ; в аккумуляторе будет содержаться
                ; число 00010110.
CLRW            ; Очистить аккумулятор.
MOVWF CNTR      ; Загрузить в регистр CNTR содержимое
                ; аккумулятора.
MOVF CNTR,1     ; Проверить содержимое регистра CNTR
                ; на нуль. После выполнения данной команды
                ; в случае нулевого содержимого
                ; регистра устанавливается флаг ZERO.
```

Команда очистки аккумулятора CLRW. После выполнения данной команды в аккумулятор заносится число 0. Пример работы:

```
MOVLW  h'FF'   ; Загрузить в аккумулятор число 11111111.
               ; После выполнения этой команды аккумулятор
               ; содержит число 11111111.
CLRW           ; Очистить аккумулятор.
               ; После выполнения этой команды
               ; аккумулятор содержит число 00000000.
```

Команда очистки содержимого регистра памяти, расположенного по адресу f, CLRF f выполняет такое же действие как и команда очистки аккумулятора. Пример работы:

```
MOVLW  h'FF'   ; Загрузить в аккумулятор число 11111111.
               ; После выполнения этой команды аккумулятор
               ; содержит число 11111111.
MOVWF CNTR     ; Загрузить в ячейку CNTR число из аккумулятора.
               ; После выполнения этой команды аккумулятор
               ; и ячейка CNTR будут содержать число 11111111.
CLRF CNTR      ; Очистить ячейку CNTR.
               ; После выполнения этой команды
               ; аккумулятор содержит число 11111111,
               ; а ячейка CNTR содержит 00000000.
```

Команда вычитания из содержимого регистра, расположенного по адресу f, содержимого аккумулятора SUBWF f,d. В зависимости от значения флага d результат операции может помещаться либо в аккумулятор (d=0), либо в исходный регистр, расположенный по адресу f (при d=1). При выполнении команды могут установиться флаги регистра состояния STATUS: флаг переноса CARRY, флаг десятичного переноса D. CARRY и флаг нуля ZERO. Использование флагов позволяет осуществлять ветвление. Пример работы:

```
MOVLW  h'FF'   ; Загрузить в аккумулятор число 11111111.
               ; После выполнения этой команды аккумулятор
               ; содержит число 11111111.
MOVWF CNTR     ; Загрузить в ячейку CNTR число из аккумулятора.
               ; После выполнения этой команды аккумулятор
               ; и ячейка CNTR будут содержать число 11111111.
MOVLW  h'01'   ; Загрузить в аккумулятор число 00000001.
               ; После выполнения этой команды
               ; аккумулятор содержит число 00000001.
SUBWF CNTR,1   ; Провести операцию <CNTR> - <W> → <CNTR>.
               ; После выполнения этой команды
               ; в ячейке CNTR будет содержаться число 11111110.
```

Команда сложения значений аккумулятора и регистра по адресу f ADDWF f,d. Как и для команды вычитания, результат может помещаться либо в аккумулятор, либо в регистр f в зависимости от значения флага d. Пример работы:

```
MOVLW  h'FF'  ; Загрузить в аккумулятор число 11111111.
MOVWF CNTR    ; Загрузить в ячейку CNTR число из аккумулятора.
MOVLW  h'01'  ; Загрузить в аккумулятор число 00000001.
ADDWF CNTR,1  ; Провести операцию <CNTR> + <W> → <CNTR>.
              ; После выполнения этой команды
              ; в ячейке CNTR будет содержаться число 00000000,
              ; установятся флаги переноса CARRY
              ; и нулевого результата ZERO.
```

Команды сложения или вычитания с константой (из константы) k содержимого аккумулятора ADDLW k и. Результат выполнения команды помещается в аккумулятор. В зависимости от результата устанавливаются флаги переноса, десятичного переноса и флаг нуля. Пример работы:

```
MOVLW  h'10'  ; Загрузить в аккумулятор число 00010000.
SUBLW  h'FF'  ; Вычесть из числа 11111111 значение аккумулятора.
              ; После выполнения данной команды в аккумуляторе
              ; будет содержаться число 11101111.
ADDLW  h'01'  ; Сложить число 00000001 с аккумулятором.
              ; После выполнения данной команды в аккумуляторе
              ; будет содержаться число 11110000.
```

Команда логического умножения содержимого аккумулятора и содержимого регистра по адресу f ANDWF f,d. Результат команды помещается в зависимости от состояния флага d в аккумулятор или обратно в регистр. Исполнение команды влияет на состояние флага нулевого результата. Пример работы:

```
MOVLW  h'10'  ; Загрузить в аккумулятор число 00010000.
MOVWF CNTR    ; Загрузить в ячейку CNTR число из аккумулятора.
INCF CNTR,1   ; Инкрементировать содержимое CNTR и сохранить
              ; результат в CNTR. После выполнения команды
              ; в регистре CNTR содержится число 00010001.
ANDWF CNTR,0  ; Умножить содержимое регистра CNTR
              ; на содержимое аккумулятора, результат
              ; поместить в аккумулятор. После выполнения
              ; данной команды в аккумуляторе будет содержаться
              ; число 00010000.
```

Команды поразрядного логического сложения и сложения по модулю два IORWF f,d и XORWF f,d содержимого аккумулятора с содержимым

регистра f. Результат команды помещается в исходный регистр
аккумулятор в соответствии с флагом d. Результат выполнения
влияет на флаг нулевого результата. Команду сложения можно использовать для установки отдельных разрядов байта. Проверить состояние конкретного разряда в байте можно операцией сложения по модулю два. Пример работы:

```
MOVLW  h'10'   ; Загрузить в аккумулятор число 00010000.
MOVWF CNTR     ; Загрузить в ячейку CNTR число из аккумулятора.
INCF CNTR,1    ; Инкрементировать содержимое CNTR и сохранить
               ; результат в CNTR. После выполнения команды
               ; в регистре CNTR содержится число 00010001.
IORWF CNTR,0   ; Сложить содержимое регистра CNTR
               ; с содержимым аккумулятора, результат
               ; поместить в аккумулятор. После выполнения
               ; данной команды в аккумуляторе будет содержаться
               ; число 00010001.
XORWF CNTR,0 ; Сложить по модулю два содержимое регистра CNTR
               ; с содержимым аккумулятора, результат
               ; поместить в аккумулятор. После выполнения
               ; данной команды в аккумуляторе будет содержаться
               ; число 00000000.
```

Команды поразрядного логического умножения, сложения и сложения по модулю два ANDLW k, IORLW k и XORLW k содержимого аккумулятора и константы k. Результат исполнения команды влияет на флаг ZERO и помещается в аккумулятор. Пример работы:

```
CLRW
IORLW b'00000001'   ; Загрузить в аккумулятор число 00000000.
                    ; Установить младший бит в аккумуляторе.
                    ; После выполнения данной команды
                    ; в аккумуляторе будет содержаться
                    ; число 00000001.
XORLW b'00000001'   ; Проверить содержимое аккумулятора на
                    ; равенство числу 00000001.
                    ; После выполнения данной команды
                    ; в аккумуляторе будет содержаться
                    ; число 00000000, флаг ZERO будет установлен.
```

Команды инкремента и декремента содержимого регистра по адресу f INCF f,d и DECF f,d соответственно прибавляют или вычитают единицу из содержимого регистра f. Результат операции помещается в аккумулятор (d=0) или в исходный регистр f (d=1). Если при вычитании получился нулевой результат, устанавливается флаг ZERO. Команды используются преимущественно при организации циклов. Пример работы:

```
MOVLW  h'10'  ; Загрузить в аккумулятор число 00010000.
MOVWF CNTR    ; Загрузить в ячейку CNTR число из аккумулятора.
INCF   CNTR,0 ; Увеличить содержимое регистра CNTR
              ; на единицу, результат поместить в аккумулятор.
              ; После выполнения данной команды в аккумуляторе
              ; будет содержаться число 00010001, в ячейке CNTR
              ; останется значение 00010000.
DECF CNTR,1   ; Уменьшить содержимое регистра CNTR
              ; на единицу, результат поместить обратно
              ; в регистр CNTR. После выполнения данной
              ; команды в регистре CNTR
              ; будет содержаться число 00001111.
```

Команды инкремента и декремента с пропуском по условию INCFSZ f,d и DECFSZ f,d. Команды выполняют соответственно инкремент или декремент содержимого регистра f. Результат помещается либо в аккумулятор, либо обратно в исходный регистр в зависимости от флага d. Если в результате выполнения команды получается нуль, то следующая за командой инкремента или декремента команда пропускается. Команды используются совместно с командой безусловного перехода GOTO для организации циклов. Пример работы:

```
MOVLW d'10'     ; Загрузить в аккумулятор число 00001010.
MOVWF CNTR      ; Загрузить в регистр CNTR содержимое
                ; аккумулятора.
LOOP
DECFSZ CNTR,1   ; Уменьшить содержимое регистра CNTR на 1.
                ; Результат поместить обратно в регистр.
GOTO LOOP       ; Если при выполнении предыдущей команды
                ; результат не равен нулю, то вернуться
                ; к метке LOOP. Цикл будет выполняться до тех
                ; пор, пока очередное уменьшение не приведёт
                ; к нулевому результату. Так как начальное
                ; значение счётчика цикла CNTR равно 10,
                ; то выполнение будет повторяться 10 раз.
                ; При  тактовой частоте 4 МГц на выполнение
                ; одной команды требуется 1 мкс.
                ; На выполнение всего цикла из двух команд
                ; уйдёт время
                ; 1 мкс * 2 команды  *  10 повторов =
                ; 20 мкс.
                ; После окончания цикла выполняются
                ; следующие команды.
MOVWF CNTR      ; Перезагрузка в регистр CNTR
                ; содержимого аккумулятора.
```

Команда инверсии COMF f,d. Команда выполняет инверсию всех разрядов содержимого регистра f. Результат выполнения команды может быть помещён в аккумулятор или в исходный регистр в зависимости от состояния флага d. Пример работы:

```
MOVLW h'15'       ; Загрузить в аккумулятор число 00010101.
MOVWF CNTR        ; Переслать в аккумулятор содержимое
                  ; регистра CNTR.
COMF CNTR,0       ; Инвертировать содержимое регистра CNTR.
                  ; Результат поместить в аккумулятор.
                  ; После выполнения команды в аккумуляторе
                  ; будет содержаться число 11101010.
```

Команда обмена нибблами в регистре SWAPF f,d . При выполнении команды в регистре f старший и младший нибблы меняются местами. Результат записывается либо в аккумулятор, либо в исходный регистр в соответствии с флагом d. Пример работы:

```
MOVLW h'50'       ; Загрузить в аккумулятор число 01010000.
MOVWF CNTR        ; Загрузить в регистр CNTR содержимое
                  ; аккумулятора.
SWAPF CNTR,0      ; Выполнить обмен нибблов в регистре CNTR.
                  ; Результат поместить в аккумулятор.
                  ; После выполнения команды в аккумуляторе
                  ; будет содержаться число 00000101.
```

Команды циклического сдвига разрядов вправо и влево. RRF f,d и RLF f,d. Команды предназначены для выполнения сдвига по кольцу через флаг переноса CARRY. По команде RRF f,d младший разряд из регистра f записывается в флаг CARRY, следующий бит занимает его место и так далее. Место старшего бита занимает значение флага переноса. Аналогично, но в противоположную сторону, выполняется сдвиг влево по кольцу через флаг переноса. Результат может сохраняться либо в аккумуляторе, либо в исходном регистре в соответствии с флагом d. Команда используется для умножения или деления чисел на 2, 4, 8 и так далее. Пример работы:

```
        MOVLW d'3'        ; Загрузить в аккумулятор степень двойки
                          ; делителя: 2^3=8.
        MOVWF CNTR        ; Загрузить в регистр CNTR содержимое
                          ; аккумулятора.
        MOVLW h'20'       ; Загрузить в аккумулятор делимое
                          ; число 00100000 (d'32').
        MOVWF NUMBER      ; Загрузить в регистр NUMBER делимое.
LOOP                      ; Цикл деления повторяется число раз,
                          ; равное степени двойки делителя.
        RRF NUMBER,1      ; Разделить содержимое регистра
```

```
                    ; NUMBER на два. Результат записать
                    ; в исходный регистр.
ANDLW b'01111111'   ; Замаскировать старший разряд, полученный
                    ; из флага переноса.
DECFSZ CNTR,1       ; Уменьшить значение счётчика на 1.
GOTO LOOP           ; Продолжить умножение требуемое число раз.
MOVF NUMBER,0       ; После окончания деления поместить результат
                    ; в аккумулятор. После выполнения команды в
                    ; аккумуляторе будет число 00000100 (d'04').
```

Команды сброса и установки разрядов BSF f,b и BCF f,b. Команды позволяют изменять в регистре f состояние разряда с номером b. Команда BSF f,b служит для установки бита, команда BCF f,b – для сброса бита. Выполнение команд не изменяет состояние флагов регистра STATUS. Пример работы:

```
MOVLW b'11111111'   ; Загрузить в аккумулятор число 11111111.
MOVWF CNTR          ; Переслать в регистр CNTR содержимое
                    ; аккумулятора.
BCF CNTR,5          ; Сбросить 5 разряд в регистре CNTR.
                    ; После выполнения данной команды в регистре
                    ; CNTR будет содержаться число 11011111.
CLRW                ; Очистить аккумулятор.
MOVWF CNTR          ; Переслать в регистр CNTR содержимое
                    ; аккумулятора.
BSF CNTR,4          ; Установить 4 разряд в регистре CNTR.
                    ; После выполнения данной команды в регистре
                    ; CNTR будет содержаться число 00010000.
```

Команды проверки состояния битов с условным переходом BTFSS f,b и BTFSC f,b. Команды осуществляют проверку разряда с номером b в регистре f. Команда BTFSC проверяет разряд на ноль, команда BTFSS – на единицу. В случае успешной проверки следующая за командой проверки команда пропускается. Команда используется для создания циклов ожидания, а также для проверки состояния регистра STATUS. Пример работы:

```
LOOP                ; Цикл ожидания.
    BTFSS PORTA,7   ; Проверить старший разряд порта А на наличие
                    ; логической единицы.
    GOTO LOOP       ; Продолжить ожидание.
    MOVF PORTA,0    ; После появления единицы в старшем разряде
                    ; переслать данные из порта А в аккумулятор.
    MOVWF PORTRD    ; Сохранить данные из порта А в регистре
                    ; PORTRD.
    RLF PORTRD,1    ; Сдвинуть старший бит в флаг переноса.
```

```
BTFSS STATUS,C      ; Проверить флаг переноса.
GOTO LOOP           ; Если флаг переноса сброшен, то чтение
                    ; из порта А произошло после исчезновения
                    ; единицы на старшем разряде порта. Повторить
                    ; цикл ожидания готовности данных.
                    ; Если чтение прошло успешно, бит установлен,
                    ; продолжить работу программы.
```

Команды работы с подпрограммами CALL k и RETURN. Команда CALL k осуществляет вызов подпрограммы по адресу k. После выполнения вызова управление передаётся команде по адресу k, а в стек заносится адрес следующей за командой вызова подпрограммы команды. Подпрограмма должна заканчиваться командой RETURN. При исполнении этой команды из стека берётся верхнее значение и переносится в программный счётчик. Далее выполнение программы продолжается с места перехода в подпрограмму. Всего можно вызывать до восьми вложенных подпрограмм, что ограничено объёмом стека. Содержимое аккумулятора и регистра STATUS при вызове подпрограммы не сохраняется. Пример работы:

```
LOOP                ; Цикл формирования последовательности
                    ; импульсов на младшем выходе порта А.
    CALL PULSE      ; Перейти к выполнению подпрограммы
                    ; формирования импульса.
    GOTO LOOP       ; Сформировать новый импульс.
PULSE               ; Подпрограмма формирования
                    ; одиночного импульса.
    BSF PORTA,0     ; Сформировать положительный фронт
                    ; импульса.
    MOVLW d'10'     ; Загрузить в аккумулятор число 10.
    MOVWF CNTR      ; Переслать в регистр CNTR содержимое
                    ; аккумулятора. Инициализировать счётчик.
DELAY
    DECFSZ CNTR,1   ; Выполнить задержку при тактовой частоте
                    ; 4 МГц на
                    ; 1 мкс * 2 команды * 10 раз = 20 мкс.
    GOTO LOOP       ;
    BCF PORTA,0     ; Сформировать отрицательный фронт
                    ; импульса.
    RETURN          ; Выйти из подпрограммы.
```

Команды специального возврата из подпрограмм RETLW k и RETFIE. Команда RETLW k производит возврат из подпрограммы с аккумулятором, заполненным константой k. Команда RETFIE разрешает прерывания после выхода из подпрограммы и используется для выхода из подпрограмм обработки прерываний. Пример работы:

```
INITE                   ; Программа формирования
                        ; последовательностей из 100 импульсов
                        ; с паузой между ними.
        MOVLW d'100'    ; Загрузить в аккумулятор число 100.
        MOVWF SERIAL    ; Переслать в регистр CNTR содержимое
                        ; аккумулятора. Инициализировать счётчик
                        ; последовательности импульсов.
LOOP                    ; Цикл формирования последовательности
                        ; импульсов на младшем выходе порта А.
        CALL PULSE      ; Перейти к выполнению подпрограммы
                        ; формирования импульса.
        INCFSZ SERIAL,0 ; Проверить аккумулятор на нуль, то есть
                        ; выполнить проверку: сформировано ли 100
                        ; импульсов?
        GOTO LOOP       ; Не сформировано, продолжить формировать.

        CALL DELAY      ; Серия из 100 импульсов сформирована.
                        ; Сформировать паузу.
        GOTO INITE      ; Перейти к формированию новой
                        ; серии импульсов.
PULSE                   ; Подпрограмма формирования
                        ; одиночного импульса.
        BSF PORTA,0     ; Сформировать положительный фронт
                        ; импульса.
        CALL DELAY      ; Сформировать задержку.
        BCF PORTA,0     ; Сформировать отрицательный фронт
                        ; импульса.
        DECFSZ SERIAL,1 ; Проверить на формирование 100 импульсов.
        RETLW h'FF'     ; Выйти из подпрограммы с аккумулятором
                        ; равным 11111111, если 100 ещё нет.
        RETLW h'00'     ; Выйти из подпрограммы с аккумулятором
                        ; равным 00000000, если 100 импульсов
                        ; сформировано.
DELAY                   ; Подпрограмма формирования задержки.
        MOVLW d'10'     ; Загрузить в аккумулятор число 10.
        MOVWF CNTR      ; Переслать в регистр CNTR содержимое
                        ; аккумулятора. Инициализировать счётчик.
DELLOOP
        DECFSZ CNTR,1   ; Выполнить задержку при тактовой частоте
                        ; 4 МГц на
                        ; 1 мкс * 2 команды * 10 раз = 20 мкс.
        GOTO DELLOOP    ;
        RETURN          ; Выйти из подпрограммы задержки.
```

Команда сброса сторожевого таймера CLRWDT. Команда используется для обнуления счётчика сторожевого таймера. Рекомендуется исполнять эту команду через период времени равный половине длительности счёта сторожевого таймера. Связано это с тем, что задающий генератор таймера имеет высокую погрешность, в результате чего сброс микроконтроллера может произойти раньше предусмотренного времени.

Команда перевода в ждущий режим SLEEP. Команда используется для перевода микроконтроллера в режим с пониженным потреблением энергии. Работа микроконтроллера приостанавливается до появления прерывания или срабатывания сторожевого таймера. Также работу можно восстановить сигналом сброса.

ОТЛАДКА ПРОГРАММ В СРЕДЕ MPLAB

Для отладки программ микроконтроллеров семейства PIC используется интегрированная среда разработки MPLAB. При запуске программной оболочки среды на исполнение открывается окно (рис. 25). Окно оболочки содержит меню «Файл», «Редактирование», «Вид», «Проект», «Отладка» и «Настройки». Меню «Файл» позволяет открывать старые проекты и создавать новые. Меню «Редактирование» и «Вид» позволяют проводить стандартные операции над текстом программ и изменять их представление на экране, а также управлять отображением рядом вспомогательных окон.

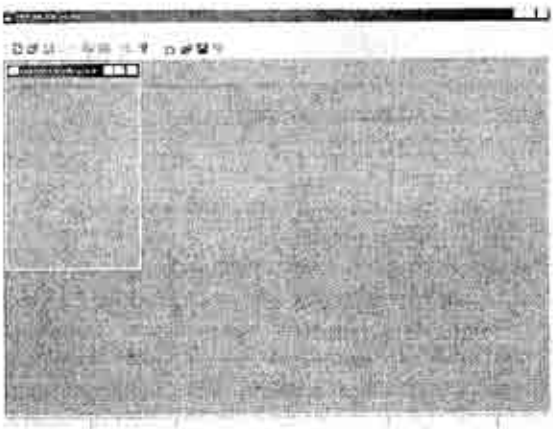

Рис. 25. Окно программной оболочки MPLAB

Меню «Проект» служит для управления проектами. Через меню «Проект» осуществляется вызов мастера проектов, с помощью которого можно быстро определить все основные параметры проектируемого устройства на микроконтроллере. Мастер проектов делит процесс создания проекта на четыре шага. Первый шаг определяет тип микроконтроллера, используемого в проекте. Второй шаг позволяет выбрать тип используемых в проекте ассемблера, компилятора и сборщика. На третьем шаге указывается имя проекта и место для его размещения. На четвёртом шаге к проекту добавляются ранее созданные файлы. В меню «Проект» также присутствует команда ассемблирования и сборки всего проекта.

Меню «Отладка» содержит команды, предназначенные для управления отладчиком. В данном меню можно выбрать тип отладчика, а также управлять его работой. Возможно как пошаговое, так и непрерывное исполнение программы.

Работа с программой начинается с открытия файла проекта. При этом появляется окно проекта, в котором содержатся все прикреплённые к проекту файлы. Выбор соответствующего файла ассемблерного текста позволяет открыть его в окне редактора. В редакторе все служебные слова, переменные,

константы и комментарии выделяются своими цветами, которые можно настроить через меню «Настройки».

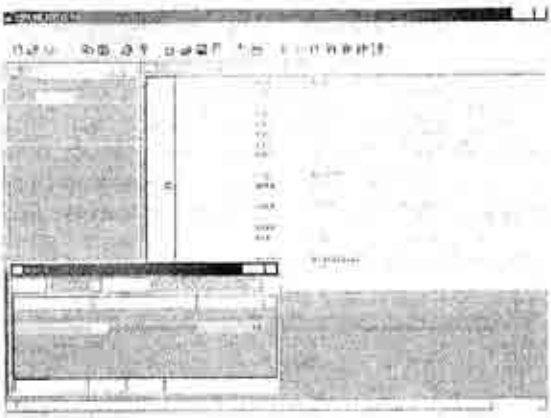

Рис. 26. Симуляция исполнения программы

Ассемблирование программы сопровождается появлением окна «Вывод». В данном окне отражаются все сообщения ассемблера о возникших ошибках. При отсутствии ошибок появляется возможность отладки программы путём симулирования её выполнения. Текущая исполняемая команда отмечается стрелкой-указателем. При необходимости допускается принудительное изменение значения счётчика команд через контекстное меню в редакторе с предварительной установкой курсора на требуемую строку.

Наблюдение за работой программы осуществляется через окно «Просмотр» (рис. 26). В окно помещаются переменные, значения которых необходимо исследовать при выполнении программы. Также окно позволяет наблюдать за состоянием всех регистров микроконтроллера, включая порты ввода-вывода. При необходимости можно оперативно изменять значения переменных или регистров, указанных в окне просмотра.

ОРГАНИЗАЦИЯ ВЫВОДА ИНФОРМАЦИИ НА ДИСПЛЕЙ

Для отображения информации о текущем состоянии устройства на микроконтроллере может использоваться светодиодный дисплей. Обычно в качестве базового используется семисегментный светодиодный индикатор (рис. 27). Один светодиод соответствует одному сегменту. Каждый сегмент индикатора обозначается латинской буквой. Все аноды светодиодов индикатора соединяются вместе, таким образом получается диодная матрица с общим анодом. Чтобы зажечь какой-либо сегмент индикатора, необходимо приложить разность потенциалов в прямом направлении к общему аноду и соответствующему катоду светодиода. В цифровых устройствах общий анод подключается к линии питающего напряжения +5В, а катоды индикатора через балластные резисторы подключаются к буферным элементам, которые, в свою очередь, подключаются к регистрам. Состояние регистров соответствует отображаемому символу на индикаторе.

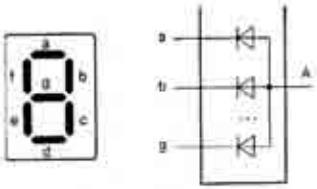

Рис. 27. Светодиодный индикатор

Объединяя светодиодные индикаторы в матрицу, можно формировать многоразрядные дисплеи. Отображение информации на многоразрядных дисплеях сопряжено с пропорциональным увеличением количества регистров, буферных элементов, балластных сопротивлений. Формирование изображения путём записи информации для отображения в каждый регистр индикатора на всё время отображения называется *статическим*. При этом все аноды индикаторов соединяются вместе и подключаются к линии питания, а катоды каждого индикатора – к своему регистру через балластные резисторы и буферные элементы. С целью снижения аппаратных затрат был разработан *динамический* метод формирования изображения. Динамический метод основан на свойстве инерционности зрения человека. Изображение, которое находится перед глазами человека, в случае его исчезновения остаётся на некоторое время в памяти и воспринимается как фактическое в течение 1/10...1/12 с. Метод динамической индикации заключается в последовательном отображении всех знаков на индикаторе. В текущий момент времени отображается только один разряд, все остальные индикаторы погашены. При реализации данного метода все одноимённые катоды индикаторов соединятся вместе и через балластные резисторы и буферные элементы подключаются к регистрам. Аноды подключаются к линии питания через ключевые элементы. Управление ключами

осуществляется от отдельных регистров. Информация для отображения записывается в «катодные» регистры, после этого в «анодных» регистрах устанавливается соответствующий поджигаемому индикатору разряд. Через некоторое время индикатор гасится, и всё повторяется для следующего индикатора. Время горения индикатора определяется количеством разрядности дисплея. Чем больше разрядность, тем меньшее время будет гореть один индикатор. Время одного цикла для каждого индикатора не должно превышать критическое, после которого начинается заметное мерцание дисплея.

Схема дисплея при статическом методе формирования изображения приведена на рис. 28. В качестве регистров используются параллельные порты ввода-вывода микроконтроллера. Соответственно в этом случае количество разрядов дисплея ограничено количеством свободных портов.

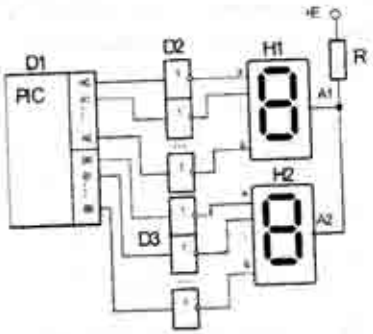

Рис. 28. Статический метод отображения информации

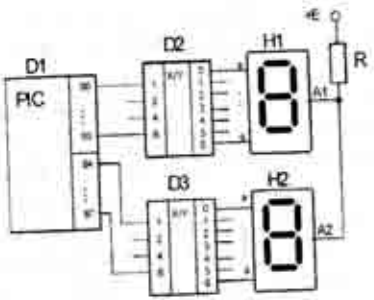

Рис. 29. Статический метод отображения с использованием преобразователей

С целью экономии свободных портов можно использовать внешние преобразователи кода из двоичного кода в код семисегментного индикатора. Такие преобразователи могут выполняться на базе однократно программируемых ПЗУ. В этом случае для каждого индикатора резервируются только четыре линии порта (рис. 29).

Схема дисплея с динамическим отображением информации приведена на рис. 30. При использовании преобразователя кода и дешифратора методом динамической индикации можно управлять шестнадцатиразрядным индикатором с помощью одного восьмиразрядного порта (рис. 31).

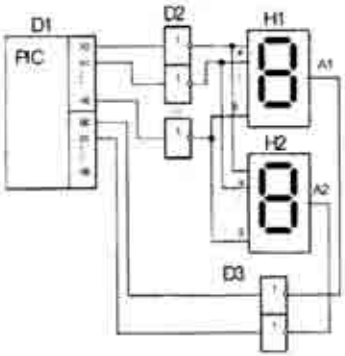

Рис. 30. Динамический метод отображения информации

Алгоритм работы программы, выполняющей динамическое отображение информации, состоит в следующем:

1) В порт «А» записать код символа первого индикатора;
2) Установкой разряда B0 порта B «зажечь» первый индикатор;
3) Выдержать паузу для отображения символа;
4) Погасить символ на первом индикаторе сбросом разряда B0 порта B;
5) Повторить пункты 1 – 4 для второго индикатора.

Преобразование двоичного кода в двоично-десятичный может выполняться программным методом. Для этого используется таблица перекодирования (табл. 2). Таблица состоит из последовательности байт, каждый из которых соответствует коду символа. Таблица размещается в памяти, начиная с некоторого базового адреса BASE. Номер байта семисегметного кода в таблице соответствует двоичному числу. Двоичное число, которое требуется отобразить, суммируется с базовым адресом. Полученный адрес указывает на расположение соответствующего семисегметного кода в памяти. Таким образом, для использования программного перекодирования подпрограмме отображения символов необходимо знать базовый адрес таблицы. Принимая двоичный код числа, подпрограмма складывает его с базовым адресом таблицы и считывает байт по полученному адресу. Полученный байт выводится через порт на индикатор.

Таблица 2

Таблица перекодирования

Адрес в таблице	Смещение/ отображаемая цифра	7-сегментный код/ abcdefg
BASE	0	1111110
BASE+1	1	0110000
BASE+2	2	1100101
BASE+3	3	1111001
BASE+4	4	0110011
BASE+5	5	1011011
BASE+6	6	1011111
BASE+7	7	1110000
BASE+8	8	1111111
BASE+9	9	1111011

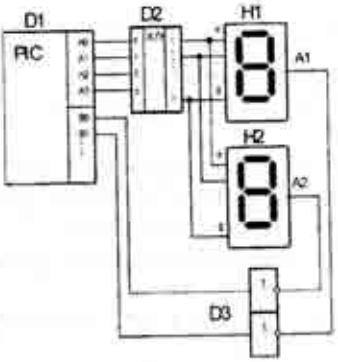

Рис. 31. Динамический метод с использованием дешифратора и преобразователя кода

Динамический метод отображения информации является наиболее экономным с точки зрения аппаратурных затрат, но требует более сложной программной реализации. Кроме того, при недостаточной производительности микроконтроллера, может наблюдаться мигание выводимой информации на дисплей, что может сказываться на утомляемости оператора. 11ри динамическом отображении информации мигание также может наблюдаться, если в качестве источника света используются люминесцентные лампы.

ФОРМИРОВАНИЕ ЗВУКОВЫХ СИГНАЛОВ

Для формирования звуковых сигналов можно использовать малогабаритные пьезоэлектрические излучатели. Излучатель подключается к одному из разрядов порта микроконтроллера. Подключение может быть выполнено через буферный элемент или напрямую (рис. 32). Второй вывод излучателя подключается к «земле» или к питающей линии. Для подачи звукового сигнала на выходе микроконтроллера формируется импульсный сигнал с частотой 300...1500 Гц. Для увеличения громкости звучания излучатель может быть подключен параллельно инвертору.

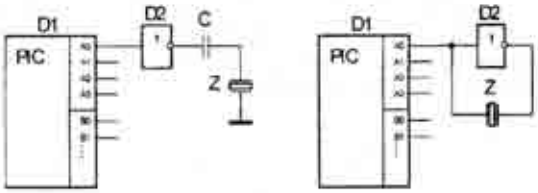

Рис. 32. Подключение пьезоэлектрического излучателя

В некоторых случаях для формирования звуковых сигналов могут использоваться маломощные динамические излучатели. В этом случае к выходу микроконтроллера напрямую или через буферный элемент может подключаться усилительный каскад, нагрузкой которого является динамическая головка.

ИСПОЛЬЗОВАНИЕ ДАТЧИКОВ УГЛА ПОВОРОТА

В качестве датчика угла поворота можно использовать переменные сопротивления. Угол поворота ползунка сопротивления пропорционален значению сопротивления. Если составить из сопротивления и ёмкости интегрирующую цепочку, то время зарядки этой цепочки будет пропорционально значению сопротивления, а значит, и углу поворота. Схема цепи показана на рис. 33. Резистор R1 ограничивает минимальное значение сопротивления при крайнем верхнем положении ползунка переменного резистора. Инициализация датчика угла поворота заключается в переводе разряда порта, к которому подключен датчик, в режим вывода информации и записи в него единицы. После этого конденсатор заряжается. При необходимости определения угла поворота разряд порта переводится в режим ввода. Конденсатор начинает разряжаться через R1 и R2. Время разряда будет пропорционально углу поворота.

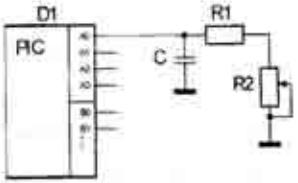

Рис. 33. Датчик угла поворота

Датчики угла поворота могут использоваться для определения уровня жидкости в различных цистернах с помощью поплавков, для указателей направления, для устройств ввода информации оператором, где каждому значению угла поворота соответствует значение вводимой информации.

ВВОД ИНФОРМАЦИИ С КЛАВИАТУРЫ

Для ручного ввода информации во время работы устройства используется клавиатура. Клавиатура может строиться по разным схемам. Наиболее простой схемой является подключение разрядов порта микроконтроллера, работающих на ввод, через токоограничивающие сопротивления к источнику питания и через кнопки к земле (рис. 34). Порт программируется для работы на ввод. При необходимости получения данных с клавиатуры считывается информация из порта. Нулевой разряд будет соответствовать номеру нажатой кнопки. Недостаток данного схемотехнического решения – в использовании числа разрядов порта, равного числу кнопок клавиатуры. Кроме того, нажатие и отпускание кнопки сопровождается так называемым «дребезгом контактов», что выражается в фиксации микроконтроллером ложных нажатий кнопки из-за возникновения коротких импульсов при замыкании и размыкании контактов. Для исключения явления «дребезга контактов» программным методом опрос состояния клавиатуры производится несколько раз в течение заданного промежутка времени, например, в течение 0,3 с. Если каждый опрос в течение данного времени будет давать одинаковый результат, то принимается решение, что кнопка нажата.

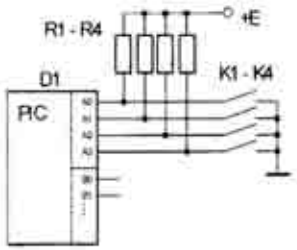

Рис. 34. Простейшая кнопочная клавиатура

Для экономного использования разрядов портов применяется матричное построение клавиатур. Например, для клавиатуры с числом кнопок, равным 16, используется матрица размером 4x4 (рис. 35). Четыре разряда используется для сканирования кнопок, четыре – для ввода информации о нажатой кнопке. Линии, подключенные к разрядам на ввод, через токоограничивающие сопротивления соединены с линией питания. Сканирование производится поразрядно, то есть в один момент времени исследуется состояние кнопок только в одной строке матрицы. Для этого на линию сканируемой строки выставляется логический ноль, на остальные линии выставляются единицы. После этого производится считывание информации из разрядов, подключенных к столбцам матрицы. Если кнопка нажата, то в соответствующем разряде появится ноль. Таким образом, получаются номер столбца и номер строки, по которым вычисляется номер

нажатой кнопки. Для устранения дребезга контактов клавиатуры программным методом производится многократное считывание состояния разрядов столбцов.

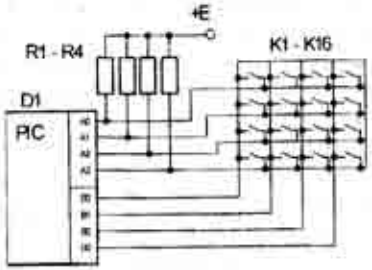

Рис. 35. Матричная клавиатура

Программный метод устранения дребезга контактов связан с повышенным использованием временных ресурсов микроконтроллера, в результате чего общее быстродействие системы может существенно снижаться.

В общем случае устройство управления на базе микроконтроллера включает в себя набор датчиков, коммутатор, аналого-цифровой преобразователь, дисплей, клавиатуру, устройство формирования управляющего сигнала (рис. 36). С помощью датчиков определяется текущее состояние управляемого объекта. Коммутатор подключает к микроконтроллеру соответствующий датчик на время его контроля. АЦП преобразует аналоговый сигнал датчика в двоичный код. Дисплей отображает текущее состояние системы управления и объекта. С помощью клавиатуры вводятся параметры, изменяющие режим работы устройства. Устройство формирования управляющего сигнала формирует сигналы, переводящие объект в требуемое состояние в соответствии со значениями датчиков.

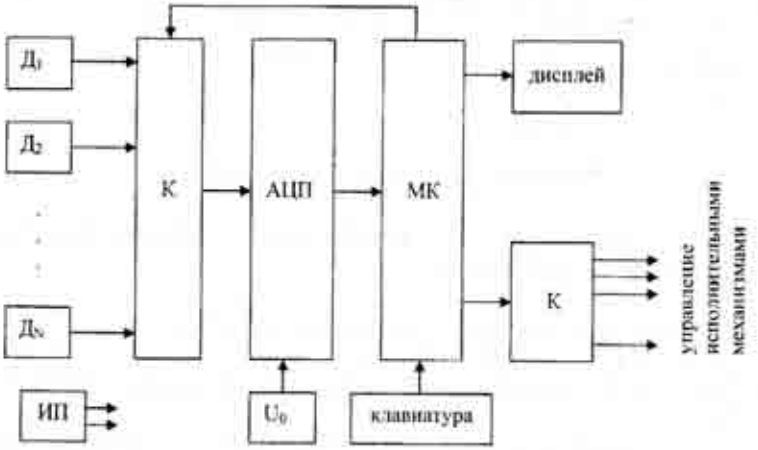

Рис. 36. Система управления объектом на базе микроконтроллера

В качестве примера устройства управления рассмотрим устройство контроля температуры и давления жидкостей в двигателе внутреннего сгорания. Функциональная схема устройства приведена на рис. 37. Устройство построено на базе микроконтроллера PIC16F84. Порт B микроконтроллера используется для ввода информации с датчиков. Порт A используется для вывода информации. Первые два разряда порта управляют коммутатором, следующие четыре разряда предназначены индикации. Седьмой разряд формирует управляющий сигнал для двигателя. Информация о температуре охлаждающей жидкости и давлении масла получается с датчиков температуры и давления. Выбор текущего датчика происходит установкой первого или второго разряда порта A микроконтроллера. Напряжение датчика через коммутатор поступает на АЦП и преобразуется в двоичный код. Код температуры или давления считывается микроконтроллером в порт B. После обработки значений температуры и

давления микроконтроллер принимает решение об остановке двигателя или о продолжении его работы. При нормальных значениях температуры и давления зажигаются зелёные светодиоды. При выходе значений за допустимые пределы зажигаются красные светодиоды. Если принимается решение об остановке двигателя, то устанавливается седьмой разряд порта А. При этом зажигается красный светодиод «двигатель остановлен».

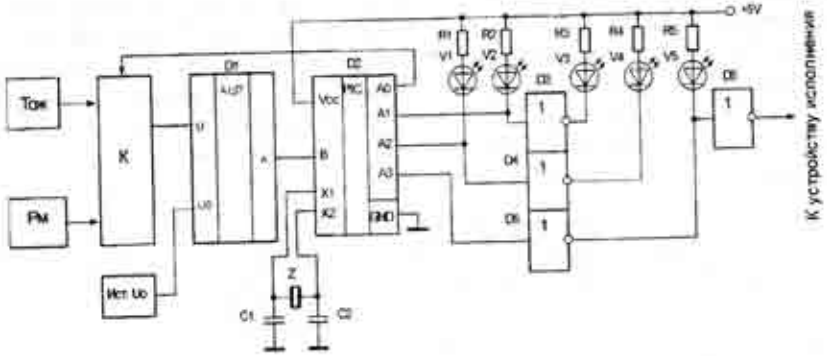

Рис. 37. Функциональная схема устройства контроля

Алгоритм работы устройства контроля при выключенном двигателе включает следующие девять пунктов.

1. Считывается значение охлаждающей жидкости $T_{ож}$;
2. Значение $T_{ож}$ сохраняется в памяти;
3. Значение $T_{ож}$ сравнивается с максимально возможным значением T_{max};
4. Если $T_{ож}$ в норме, то есть $T_{ож}<T_{max}$, то зажигается зелёный светодиод «температура двигателя в норме», и алгоритм завершает работу;
5. Если значение $T_{ож}$ превышено, то есть $T_{ож}>T_{max}$, то выдерживается пауза;
6. Повторяется считывание температуры охлаждающей жидкости $T_{ож}$;
7. Значение $T_{ож}$ сохраняется в памяти;
8. Значение $T_{ож}$ сравнивается с максимально возможным значением T_{max};
9. Если значение $T_{ож}$ повторно превышено, то есть $T_{ож}>T_{max}$, то формируется сигнал «остановка двигателя» и зажигается красный светодиод «двигатель перегрет».

Блок-схема алгоритма изображена на рис. 38.

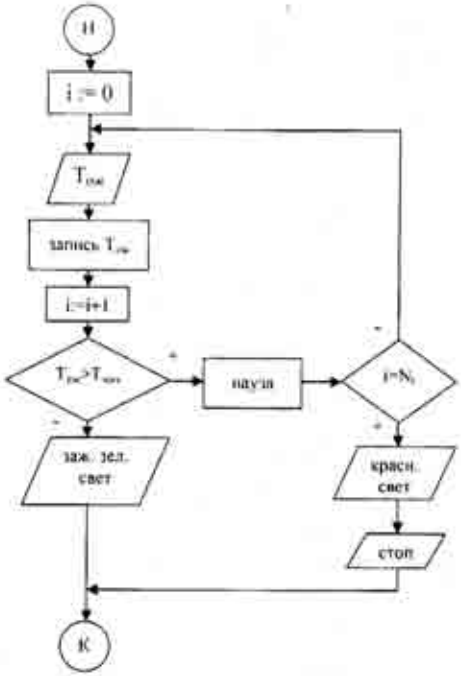

Рис. 38. Блок схема алгоритма работы устройства до запуска двигателя

При работающем двигателе алгоритм зацикливается для выполнения постоянного контроля температуры и давления.

1. Считываются значения температуры охлаждающей жидкости $T_{ож}$ и давления масла $P_м$;
2. Значения температуры $T_{ож}$ и давления $P_м$ сохраняются в памяти;
3. Значения температуры $T_{ож}$ и давления $P_м$ сравниваются с максимально допустимыми значениям T_{max} и P_{min};
4. Если значения в норме, то зажигаются зелёные светодиоды «давление масла в норме» и «температура двигателя в норме» и контроль продолжается, то есть происходит возврат к пункту 1;
5. При превышении значений максимальных уровней выдерживается пауза;
6. Производится повторное считывание и сохранение значений температуры $T_{ож}$ и давления $P_м$;
7. Значения температуры $T_{ож}$ и давления $P_м$ повторно сравниваются с максимально допустимыми значениям T_{max} и P_{min};
8. Если какое-либо из значений находятся вне нормы, то формируется сигнал остановки двигателя «остановка двигателя», зажигаются красные светодиоды в соответствии с превышенным значением температуры или давления;

9. Если повторное измерение не даёт отклонений параметров от нормы, то происходит возврат к началу алгоритма, к пункту 1.

Блок-схема алгоритма изображена на рис. 39.

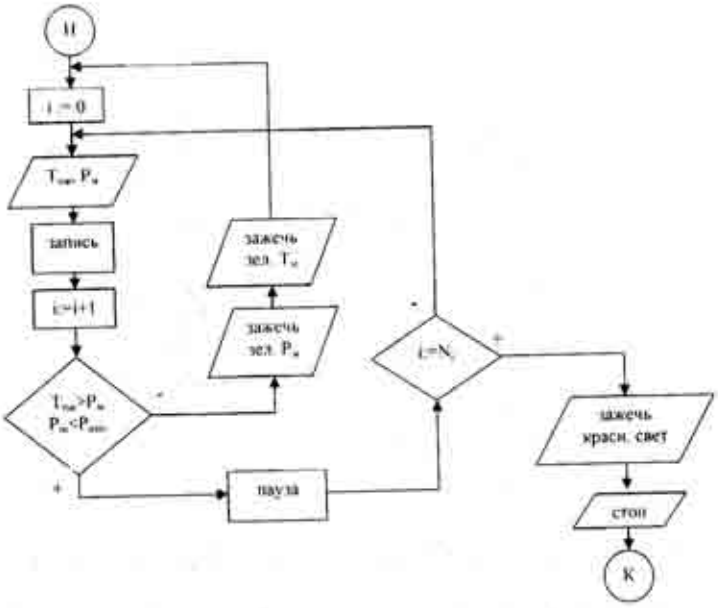

Рис. 39. Блок-схема алгоритма работы устройства при запущенном двигателе

Далее приводится программа работы микроконтроллера на ассемблере для реализации описанных выше алгоритмов.

```
LIST        P=16F84         ; Указывается тип
                            ; микроконтроллера, для которого
                            ; написана программа.
__CONFIG    03FF1H          ; Конфигурация: бит защиты
                            ; выключен, WDT выключен,
                            ; генератор- стандартный XT.
INCLUDE     <P16F84.INC>    ; Включить основные определения
                            ; для микроконтроллера 16F84.
```

```
; Резервируемые ячейки памяти для внутренних переменных.

COUNTER_J   EQU H'0C' ; Счётчик цикла по j.
DELAY       EQU H'0D' ; Счётчик подпрограммы задержки.

; Описание используемых констант.

J           EQU H'10'       ; Начальное значение счётчика j.
D           EQU D'10'       ; Начальное значение
                            ; счётчика задержки.
STOP        EQU D'3'        ; Номер разряда сигнала
                            ; «остановка двигателя».
PRESSURE    EQU D'0'        ; Номер разряда для
                            ; выбора датчика давления.
                            ; Разряд установлен.
TEMPER      EQU D'0'        ; Номер разряда для
                            ; выбора датчика температуры.
                            ; Разряд сброшен.
P_RED       EQU D'1'        ; Номер разряда красного
                            ; светодиода «давление
                            ; масла снижено».
                            ; Разряд установлен.
P_GREEN     EQU D'1'        ; Номер разряда зелёного
                            ; светодиода «давление
                            ; масла в норме».
                            ; Разряд сброшен.
T_RED       EQU D'2'        ; Номер разряда красного
                            ; светодиода «температура
                            ; двигателя превышена».
                            ; Разряд установлен.
T_GREEN     EQU D'2'        ; Номер разряда зелёного
                            ; светодиода «температура
                            ; двигателя в норме».
                            ; Разряд сброшен.
T_MAX       EQU D'127'      ; Значение максимально
                            ; допустимой температуры
                            ; двигателя.
P_MIN       EQU D'127'      ; Значение минимально
                            ; допустимого давления
                            ; масла.
START_ENG   EQU D'3'        ; Номер разряда
                            ; для запуска двигателя.
                            ; При сброшенном разряде
                            ; происходит запуск двигателя.
```

```
        ORG     0X00            ; Адрес памяти начала программы.
        GOTO    START           ; Перейти в начало программы.
START
        CLRF    PORTA           ; Очистить порт A.
        CLRF    PORTB           ; Очистить порт B.
        BSF     STATUS, RP0     ; Выбрать страницу памяти
        MOVLW   B'11111111'     ; Поместить в аккумулятор
                                ; маску работы порта A.
        MOVWF   TRISA           ; Перенести маску из аккумулятора
                                ; в управляющий регистр порта A,
                                ; установив тем самым его режим
                                ; работы на вывод.
        BCF     STATUS, RP0     ; Вернуть страницу памяти.

; Программа первого алгоритма.

A1START
        MOVLW   J               ; Записать в аккумулятор
                                ; начальное значение счётчика j.
        MOVWF   COUNTER_J       ; Сохранить это значение
                                ; в ячейке памяти счётчика.
LOOP_J
        CALL    CHECK_T         ; Считать значение температуры.
        ADDLW   H'01'           ; Проверить значение
                                ; температуры на выход
                                ; за допустимые пределы.
        BTFSS   STATUS, Z       ;
        GOTO    A2_START        ; Температура в норме,
                                ; перейти ко второму алгоритму.
        CALL    DELAY           ; Формирование паузы.
        DECFSZ  COUNTER_J, F    ; Уменьшаем значение
                                ; счётчика j на 1.
        GOTO    LOOP_J          ; Если счёт не окончен, повторить
                                ; измерение температуры.
        BSF     PORTA, STOP     ; При всех измерениях температура
                                ; вышла за пределы допустимого
                                ; значения. Сформировать сигнал
                                ; блокировки запуска двигателя
                                ; «двигатель остановлен».
        BSF     PORTA, T_RED    ; Зажечь красный светодиод
                                ; «температура превышена».

; Подпрограмма проверки температуры.
; Выходное значение подпрограммы содержится в аккумуляторе.
; Если W=FFh, то температура превышена,
; если W=0, то температура в норме.
```

```
CHECK_T
      BCF  PORTA, TEMPER      ; Подключить к микроконтроллеру
                              ; датчик температуры.
      MOVF     PORTB,W        ; Загрузить в аккумулятор
                              ; значение температуры.
      SUBLW    T_MAX          ; Сравнить значение температуры
                              ; с максимально допустимым.
      MOVLW    H'FF'          ; Подготовиться к выходу из
                              ; подпрограммы с аварийным
                              ; значением температуры.
      BTFSC    STATUS, C      ; Если температура превышена,
      RETURN                  ; то выйти из подпрограммы.
      MOVLW    H'00'          ; Если температура в норме,
      RETURN                  ; выйти из подпрограммы с
                              ; аккумулятором W=0.
; Подпрограмма формирования задержки.
DELAY
      MOVLW    D              ; Загрузить в аккумулятор
                              ; начальное значение
                              ; счётчика задержки.
      MOVWF    DELAY_D        ; Сохранить начальное значение
                              ; в ячейке памяти счётчика.
LOOP_D
      DECFSZ   DELAY_D, F     ; Уменьшить значение
                              ; счётчика задержки на 1.
      GOTO     LOOP_D         ; Если счёт не окончен,
                              ; повторить декремент.
      RETURN                  ; Задержка выполнена,
                              ; выйти из подпрограммы.

; Подпрограмма проверки давления масла.
; Выходное значение подпрограммы содержится в аккумуляторе.
; Если W=FFh, то давление снижено,
; если W=0, то давление в норме.

CHECK_P
      BCF  PORTA, PRESSURE    ; Подключить к микроконтроллеру
                              ; датчик давления.
      MOVF     PORTB, W       ; Считать значение давления
                              ; в аккумулятор.
      SUBLW    P_MIN          ; Сравнить давление с минимально
                              ; допустимым значением.
      MOVLW    H'FF'          ; Подготовиться к выходу
                              ; с аварийным значением давления.
      BTFSS    STATUS, C      ; Если уровень давления
```

```
                                ; ниже допустимого,
        RETURN                  ; то выйти из подпрограммы.
        MOVLW   H'00'           ; Если уровень давления в норме,
                                ; то записать в аккумулятор W=0
        RETURN                  ; и выйти.
;
; Программная реализация второго алгоритма.
;
A2_START
        BCF  PORTA, START_ENG   ; Запустить двигатель.
        CALL    DELAY2          ; Сформировать задержку,
                                ; в течение которой происходит
                                ; запуск двигателя.
A2_LOOP
        MOVLW   J               ; Сохранить в ячейке счётчика j
        MOVLW   COUNTER_J       ; его начальное значение.
LOOP_J2
        CALL    CHEK_T          ; Проверить температуру
                                ; двигателя.
        ADDLW   H'01'           ; Если температура в норме,
        BTFSS   STATUS, Z       ;
        GOTO    PRESSURE_M      ; перейти к проверке давления,
        MOVLW   H'01'           ;
        IORWF   FLAG, F         ; иначе установить флаг аварии.
        BSF     PORTA, T_RED    ; Зажечь красный светодиод
                                ; «температура превышена».
PRESSURE_M
        CALL    CHECK_P         ; Проверить давление масла.
        ADDLW   H'01'           ; Если давление в норме,
        BTFSS   STATUS, Z       ;
        GOTO    GREEN_CHECK     ; не изменять свечение индикаторов.
        BSF     PORTA, P_RED    ; Если давление превышено,
                                ; то зажечь красный светодиод
                                ; «давление превышено».
GREEN_CHECK
        IORWF   FLAG, F         ; Если флаг не устанавливался,
                                ; то есть и давление, и температура
                                ; в норме,
        BTFSC   STATUS, Z       ;
        GOTO    GREEN           ; то обойти повторную проверку,
        DECFSZ  COUNTER_J, F    ; иначе уменьшить значение
                                ; количества оставшихся
                                ; измерений на 1
        GOTO    LOOP_J2         ; и провести повторные измерения.
        BSF     PORTA, STOP     ; Если все измерения привели
                                ; к аварийным результатам,
```

```
                                ; то остановить двигатель и
        SLEEP                   ; перевести устройство управления
                                ; в спящий режим.
GREEN
        BCF  PORTA, P_GREEN     ; Если все параметры в норме,
        BCF  PORTA, T_GREEN     ; то зажечь зелёные индикаторы
                                ; давления и температуры.
        GOTO     A2_LOOP        ; Продолжать отслеживать
                                ; состояние двигателя.
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Структура микроконтроллеров семейства PIC.
2. Назначение узлов микроконтроллера.
3. Тактовый генератор и схема сброса микроконтроллера.
4. Подключение внешних цепей к микроконтроллеру.
5. Способы отображения информации на цифровых дисплеях.
6. Способы ввода информации.
7. Система команд микроконтроллера.
8. Способы организации ветвления и циклов.
9. Этапы проектирования микроконтроллерных устройств в среде MPLAB.

ЗАКЛЮЧЕНИЕ

Область использования микроконтроллеров в различных электронных устройствах постоянно расширяется, что обусловлено их большими функциональными возможностями. В пособии были рассмотрены основные принципы построения микроконтроллерных устройств: их аппаратной и программной составляющих.

В разделах пособия с практической точки зрения рассмотрены схемотехнические и программные стороны микроконтроллерных устройств. Пособие позволяет освоить студентам все ступени проектирования систем с микроконтроллерным управлением: от постановки задачи до её практической реализации.

Приложение А

Представление чисел в ассемблере

Формат представления	Синтаксис
Двоичные числа	b'11001100'
Восьмеричные числа	o'224'
Десятичные числа	d'115' или .115
Шестнадцатеричные числа	h'7F' или 0x7F
Символы	a'S' или 'S'

Приложение Б

Арифметико-логические операторы ассемблера

Оператор	Описание	Пример
$	текущий счётчик программы	goto $
(	левая скобка	a + (5*b)
)	правая скобка	
!	логическая инверсия («НЕ»)	if ! (x + y)
~	инверсия	z = ~ z
-	отрицательное число	- a * b
high	выделить старший байт слова	high x
low	выделить младший байт слова	low y
*	умножение	a * b
/	деление	a / b
%	модуль	a % 2
+	сложение	a + b
-	вычитание	a - b
<<	сдвиг влево	x << 1
>>	сдвиг вправо	y >> 1
>=	больше либо равно	if x >= y
>	больше	if x > y
<	меньше	if x < y
<=	меньше либо равно	if x <= y
==	равно	if x == y
!=	не равно	if x != y
&	поразрядное «И»	x & mask
^	поразрядное «Исключающее ИЛИ»	x ^ mask
|	поразрядное «Включающее ИЛИ»	x | mask
&&	логическое «И»	if (x == 1) && (y == 0)
||	логическое «ИЛИ»	if (x == 1) || (y == 0)
=	установить	x = 0
+=	сложить и установить	x += 1
-=	вычесть и установить	x -= 1
*=	умножить и установить	x *= y
/=	делить и установить	x /= y

Оператор	Описание	Пример
%=	модуль и установить	x %= 8
<<=	сдвиг влево и установить	x << 3
>>=	сдвиг вправо и установить	x >> 4
&=	«И» и установить	x %= mask
|=	«Включающее ИЛИ» и установить	x |= mask
^=	«Исключающее ИЛИ» и установить	x ^= mask
++	инкремент	j ++
--	декремент	j --

Директивы ассемблера

Директива	Описание	Пример
CBLOCK S ENDC	Блок констант, начиная с адреса S	CBLOCK h'10' X,Y Z,W ENDC
ORG	Установить адрес начала программы	ORG h'00'
END	Окончание программы	END
EQU	Присвоить значение константе	COUNTER EQU h'20'
SET	Присвоить значение константе. В дальнейшем значение может быть переопределено	COUNTER SET h'10'
INCLUDE	Подключить файл библиотеки	INCLUDE p16f84a.inc
__CONFIG	Установить конфигурационное слово	__CONFIG h'03FF1'
CONSTANT	Присвоить значение символьной константе	CONSTANT f = 100
VARIABLE	Присвоить значение символьной константе. В дальнейшем значение может быть переопределено	VARIABLE g = 95
RADIX HEX | DEC | OCT	Выбрать систему счисления по умолчанию	RADIX OCT

Регистры микроконтроллера

Адрес	Нулевая страница памяти	Адрес	Первая страница памяти
00h	IND0 (*) – косвенный адрес	80h	IND0 (*) – косвенный адрес
01h	TMR0	81h	OPTION
02h	PCL	82h	PCL
03h	STATUS	83h	STATUS
04h	FSR	84h	FSR
05h	PORT A	85h	TRISA
06h	POTR B	86h	TRISB
07h	0	87h	0
08h	EEDATA	88h	EECON1
09h	EEADR	89h	EECON2
0Ah	PCLATH	8Ah	PCLATH
0Bh	INTCON	8Bh	INTCON
0Ch-2Fh	36 регистров общего назначения	8Ch-AFh	Копия 36 регистров нулевой страницы
50h-7Fh	0	D0h-FFh	0

Регистры специального назначения

Регистр	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
INDF	Значение FSR используется для косвенной адресации к памяти данных. Физически регистр не существует							
TMR0	8-разрядный таймер							
PCL	Младшие 8 разрядов счётчика команд PC							
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
FSR	Регистр косвенной адресации							
PORTA	-	-	-	RA4/ TOCKI	RA3	RA2	RA1	RA0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/ INT
EEDATA	Регистр данных EEPROM							
EEADR	Регистр адреса EEPROM							
PCLATH	-	-	-	Старшие разряды счётчика команд PC				
INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
OPTION	RBPU	INTEG	TOCS	TOSE	PSA	PS2	PS1	PS0
TRISA	-	-	-	Биты управления порта A				
TRISB	Биты управления порта B							
EECON1	-	-	-	EEIF	WRERR	WREN	WR	RD
EECON2	Регистр управления EEPROM. Физически не существует.							

Назначение разрядов регистров специального назначения

Регистр	Разряд	Описание
STATUS	IRP	Разряд страницы памяти при косвенной адресации
	RP1	Разряды, определяющие рабочую страницу памяти
	RP0	
	TO	Флаги запуска микроконтроллера по включению питания, срабатыванию сторожевого таймера или выхода из спящего режима
	PD	
	Z	Флаг нулевого результата
	DC	Флаг десятичного переноса
	C	Флаг переноса
PORTA	TOCKI	Вход тактового сигнала для внутреннего таймера
	RA4- RA0	Линии порта A
PORTB	INT	Вход внешнего сигнала прерывания
	RB7-RB0	Линии порта B

Регистр	Разряд	Описание
INTCON	GIE	Разрешить все немаскированные прерывания
	EEIE	Разрешить прерывания по окончании записи в EEPROM
	TOIE	Разрешить прерывания по переполнению счётчика встроенного таймера
	INTE	Разрешить приём прерываний от внешнего источника по линии INT
	RBIE	Разрешить прерывания по изменению на линиях порта B
	TOIF	Флаг переполнения счётчика встроенного таймера
	INTF	Флаг внешнего прерывания
	RBIF	Флаг прерывания по изменению на линиях порта B
OPTION	RBPU	«Подтягивание» линий порта B
	INTEG	Выбор переднего или заднего фронта импульса для выполнения прерывания
	TOCS	Выбор внешнего или внутреннего источника тактовой частоты для таймера
	TOSE	Выбор переднего или заднего фронта импульса для счёта таймера
	PSA	Подключение предварительного счётчика к сторожевому или обычному таймеру
	PS2	Выбор коэффициента счёта предварительного счётчика
	PS1	
	PS0	
EECON1	EEIF	Флаг выполнения операции записи в EEPROM
	WRERR	Флаг ошибки при работе с EEPROM
	WREN	Разрешение записи в EEPROM
	WR	Произвести запись в EEPROM
	RD	Произвести чтение из EEPROM

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Предко, М. Руководство по микроконтроллерам : пер. с англ. : В 2 т. / М. Предко. – М. : Постмаркет, 2001. – Т. 1. – 416 с., Т. 2. – 488 с. – (Библиотека современной электроники).

2. Предко, М. Справочник по PIC-микроконтроллерам : пер. с англ. / М. Предко – М. : ДМК Пресс, 2004. – 504 с. : табл., схем.

3. PIC-микроконтроллеры. Практика применения : пер. с фр. – М. : ДМК Пресс, 2004. – 270 с. : ил. – (Серия «Справочник»).

4. Бродин, В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин. – М. : ЭКОМ, 2002. – 400 с. : ил. – (Современная микропроцессорная техника).

5. Сташин, В. В. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева; В. В. Сташин, О. Ф. Мологонцева.– М. : Энергия, 1990. – 224 с.

Учебное издание

Марченко Максим Владимирович

УСТРОЙСТВА НА МИКРОКОНТРОЛЛЕРАХ
Учебное пособие

Редактор Н. А. Евдокимова
Подписано в печать 28.09.2007. Формат 60×84/16
Бумага тип. №1. Печать трафаретная. Усл. печ. л. 3,95.
Тираж 75 экз. Заказ 1315.
Ульяновский государственный технический университет
432027, Ульяновск, Сев. Венец, 32
Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32