PIC16F628

MICROCHIP

1 Введение в микропроцессорную технику

1.1 Классификация микропроцессоров

Микропроцессор — это программно управляемое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки, выполненное в виде одной (или нескольких) интегральной схемы с высокой степенью интеграции электронных элементов.

Микропроцессоры можно разделить на четыре больших класса.

1. Универсальные микропроцессоры — это микропроцессоры для микропроцессорных систем общего назначения. Характеризуются полным набором команд, развитой системой адресации, высоким быстродействием за счет применения конвейерной обработки команд, сверхоперативной памяти, высокой тактовой частоты и разрядности. Универсальные МП ориентированы на выполнение любого алгоритма за приемлемое время и применяются для решения широкого круга вычислительных задач.

2. Однокристальные микроЭВМ (микроконтроллеры) — специализированные программируемые устройства, предназначенные для задач управления. МК строятся по гарвардской архитектуре, т. е. их память программ и данных физически и логически разделены. В одном кристалле МК присутствует АЛУ, память данных, память программ, генератор тактовых импульсов и множество разнообразных устройств ввода вывода, причем как для цифровых, так и для аналоговых сигналов. Чаще всего МК характеризуются сокращенным набором команд. Система команд ориентирована на задачи логического анализа, управления и регулирования в системах автоматики. Внешний интерфейс предназначен для прямого подключения к объекту, т. е. посредством усилительных устройств, но минуя устройства, выполняющие логическую обработку цифровых сигналов МК.

3. Сигнальные процессоры или цифровые процессоры обработки сигнала — это специализированные микропроцессоры, ориентированные на задачи цифровой фильтрации в реальном времени. ЦПОС имеют сокращенный набор команд и построены по гарвардской и супер-гарвардской архитектуре. Характерны встроенные память программ и данных, а также некоторые устройства ввода-вывода. ЦПОС отличаются от универсальных МП наличием обычно трех вычислительных устройств: АЛУ, умножителя-накопителя, сдвигателя. Производительность сигнальных процессоров при решении специальных задач существенно превышает производительность универсальных МП за счет параллельного выполнения операций и одновременной пересылки разной информации по отдельным магистралям: шинам адреса и данных памяти программ и данных и шине результатов.

4. Транспьютеры — это специальные МП с сокращенным набором команд, ориентированные на коммуникационные задачи. Структурно-логическая организация транспьютеров проста. Характерен расширенный интерфейс, приспособленный для быстрого обмена данными по многим каналам одновременно.

1.2 Общие сведения по микроконтроллерам Microchip

Фирма Microchip выпускает широкий спектр 8-разрядных однокристальных микроконтроллеров. В дельнейшем для всех МК этой фирмы будет использоваться краткое название «PIC-контроллеры». В настоящее время существует четыре семейства PIC-контроллеров. Кратко охарактеризуем каждое из них.

1. PIC12 — простейшие дешевые микроконтроллеры в восьмивыводных корпусах. Обладают минимумом возможностей, хотя, в последнее время семейство пополнилось новыми микросхемами, со встроенной FLASH-памятью программ и десятиразрядным АЦП. Это существенно расширяет круг задач, где могут найти свое применение PIC12. Система команд включат всего 33-35 инструкций в зависимости от конкретного типа.

2. PIC16 — среднее семейство микроконтроллеров. Включает множество микроконтроллеров с самыми разнообразными возможностями. Предназначены для широкого круга относительно несложных задач. Наиболее развитые модели оснащены FLASH-ПЗУ программ, многоканальным 10-разрядным АЦП, 16-разрядным таймером, поддерживают несколько стандартных интерфейсов для включения в вычислительную сеть. В настоящее время семейство быстро пополняется новыми моделями. Система команд совпадает с семейством PIC12.

3. PIC17 — семейство микроконтроллеров с расширенной периферией. Отличаются большим количеством цифровых выводов: от 33 до 66, расширенной системой команд из 58 инструкций. Тактовая частота повышена до 33 МГц. Некоторые контроллеры имеют мощные выводы с нагрузочной способностью 35 и даже 60 мА. Память программ расширяется до 65536 слов при помощи внешних микросхем ПЗУ. Имеется два модуля УСАПП. Недостатком является отсутствие вариантов с FLASH-ПЗУ программ. В настоящее время семейство не развивается.

4. PIC18 — старшее семейство микроконтроллеров Microchip. Семейство включает широкое многообразие микросхем с разнообразными возможностями. Все контроллеры отличаются эффективной системой команд из 77 инструкций, высоким быстродействием (тактовая частота до 40 МГц). Наиболее развитые модели имеют очень большой объем памяти программ и памяти данных — 128 Кбайт и более 3,5 Кбайт, соответственно. Количество цифровых выводов достигает 68. Наиболее развитые модели имеют 16-канальный 10-разрядный АЦП, аналоговые компараторы, три 16-разрядных таймера, два модуля УСАПП. Память программ в старших моделях расширяется. В настоящее время семейство бурно развивается.

2 Основные характеристики МК PIC16

PIC16 представляет собой семейство 8-разрядных однокристальных микроконтроллеров. Упрощенная структурная схема МК PIC16F628 показана на рисунке 3.1. Как все микроконтроллеры, PIC16 построены по гарвардской архитектуре. Это означает, что память программ и данных физически и логически разделены. Каждое из запоминающих устройств представляет собой отдельный узел. Обращение к ним производится разными способами.

Память программ выполнена в виде ПЗУ, в настоящее время чаще всего электрически стираемого (ПЗУ типа FLASH). ПП имеет 14-битную организацию; каждая ячейка (слово) вмещает одну команду. Объем варьируется от 512 до 8192 слов. Наращивание ПП при помощи внешних БИС невозможно. Во многих МК ПП является программно доступной для чтения или даже для чтения и записи.

Память данных представляет собой набор регистров. Их количество от 24 до 368. Кроме ПП и ПД многие МК семейства имеют встроенное ЭППЗУ данных объемом от 64 до 128 байт.

Прежде чем продолжить рассмотрение основных характеристик PIC-контроллеров разъясним понятия машинного такта и машинного цикла. Длительность *машинного такта* — это единица дискретного времени, которая задается тактовым генератором. Другими словами, машинный такт — это период тактовых импульсов. *Машинным циклом* называют процедуру и время одного обмена процессора с внешним устройством. Машинный цикл состоит из нескольких машинных тактов. Применительно к PIC-контроллерам будем называть машинным циклом интервал времени, равный четырем машинным тактам. Иначе говоря, машинный цикл — это время выполнения одной команды, но не команды передачи управления, которая выполняется за два машинных цикла.

В дальнейшем чаще всего предполагается, что тактовая частота МК составляет 20 МГц и, следовательно, длительность машинного цикла 0,2 мкс.

Система команд PIC16 сокращенная. (RISC-архитектура). Поскольку набор команд тесно связан с архитектурой контроллера, то говорят об архитектуре с сокращенным

набором команд (RISC-архитектуре). Для RISC-архитектуры характерно небольшое количество команд (обычно от 30 до 100), минимальная длина и как следствие минимальное время выполнения команды. Система команд PIC16 включает 37 инструкций. Все команды имеет одинаковую длину — 14 байт и занимают одно слово памяти программ. Все команды, кроме команд передачи управления выполняются за один машинный цикл. Команды передачи управления — за два машинных цикла. Тактовая частота достигает 20 МГц. Длительность машинного цикла равна четырем периодам тактовой частоты и составляет 200 нс.

Количество линий портов ввода-вывода варьируется от 13 до 33 (PIC16C9xx имеет 52 линии).

МК семейства PIC16 могут оснащаться разнообразными периферийными устройствами. Перечислим наиболее распространенные.

1. АЦП разрядностью 8, 10 бит (число каналов до 8);
2. аналоговые компараторы;
3. программируемый источник опорного напряжения;
4. модули захвата-сравнения и ШИМ;
5. один 16-разрядный и два 8-разрядных таймера-счетчика, сторожевой таймер;
6. электрически перепрограммируемое ПЗУ данных;
7. универсальный синхронно-асинхронный приемопередатчик (УСАПП);
8. модули поддержки интерфейсов I^2C, SPI;
9. параллельный ведомый порт PSP.

Некоторые типы МК имеют следующие периферийные устройства:

1. 12-разрядный АЦП;
2. 8-разрядный ЦАП;
3. модули интерфейсов LIN и USB;
4. модуль управления ЖКИ
5. операционный усилитель.

2 Структура микроконтроллера PIC16F628

На рисунке 2.1 показана упрощенная структурная схема микроконтроллера PIC16F628. В целом она соответствует схеме, приведенной в техническом описании [7]. Здесь из структуры исключены тактовый генератор, узлы, отвечающие за формирования сигналов сброса и другие элементы, которые не важны для понимания основных принципов работы контроллера. Кроме того, показать на схеме связь перечисленных узлов с остальными элементами контроллера представляется крайне затруднительным.

Основу МК составляют память программ, память данных и арифметико-логическое устройство (АЛУ). Рассмотрим взаимодействие узлов контроллера, опираясь на структурную схему.

Выборка очередной команды из памяти программ производится по адресу, хранимому в счетчике команд. Адрес передается из счетчика по 13-разрядной шине. Счетчик команд имеет двунаправленную связь со стеком. Она необходима для загрузки значения из стека и для сохранения в стек адреса возврата. Кроме того, счетчик команд может загружаться информацией из ПД, поступающей по 8-разрядной шине данных.

Команда выбирается из памяти программ и защелкивается в регистре команд. 14-разрядная команда поступает на дешифратор команд и устройство управления. Кроме того, 7 бит из адресной части команды поступают на вход мультиплексора адреса, и 8-бит через мультиплексор на вход АЛУ.

Рассмотрим работу арифметико-логического устройства. На вход АЛУ поступают два операнда. Один из них выбирается из аккумулятора и не требует указания адреса. Другой поступает с выхода мультиплексора (MUX). Мультиплексор выполняет функцию коммутатора, подавая на один из входов АЛУ либо информацию из памяти данных (по шине данных), либо из памяти программ. В первом случае источником информации явля-

ется регистр ПД, адресуемый прямо или косвенно, во втором — это непосредственно адресуемая константа, содержащаяся в адресной части команды.

Информация на выходе АЛУ направляется в память данных или в аккумулятор в зависимости от значения модификатора d.

Рассмотрим процесс выборки информации из памяти данных. Ключевым моментом здесь является формирование адреса. Адрес ячейки памяти формируется мультиплексором адреса (MUX адреса) и имеет разрядность 9 бит. Мультиплексор служит для выбора источника адреса. Адрес может извлекаться из адресной части команды (прямая адресация) или из регистра указателя FSR (косвенная адресация).

Доступ к периферийным устройствам контроллера (изображены в нижней части схемы) осуществляется через шину данных памяти данных.

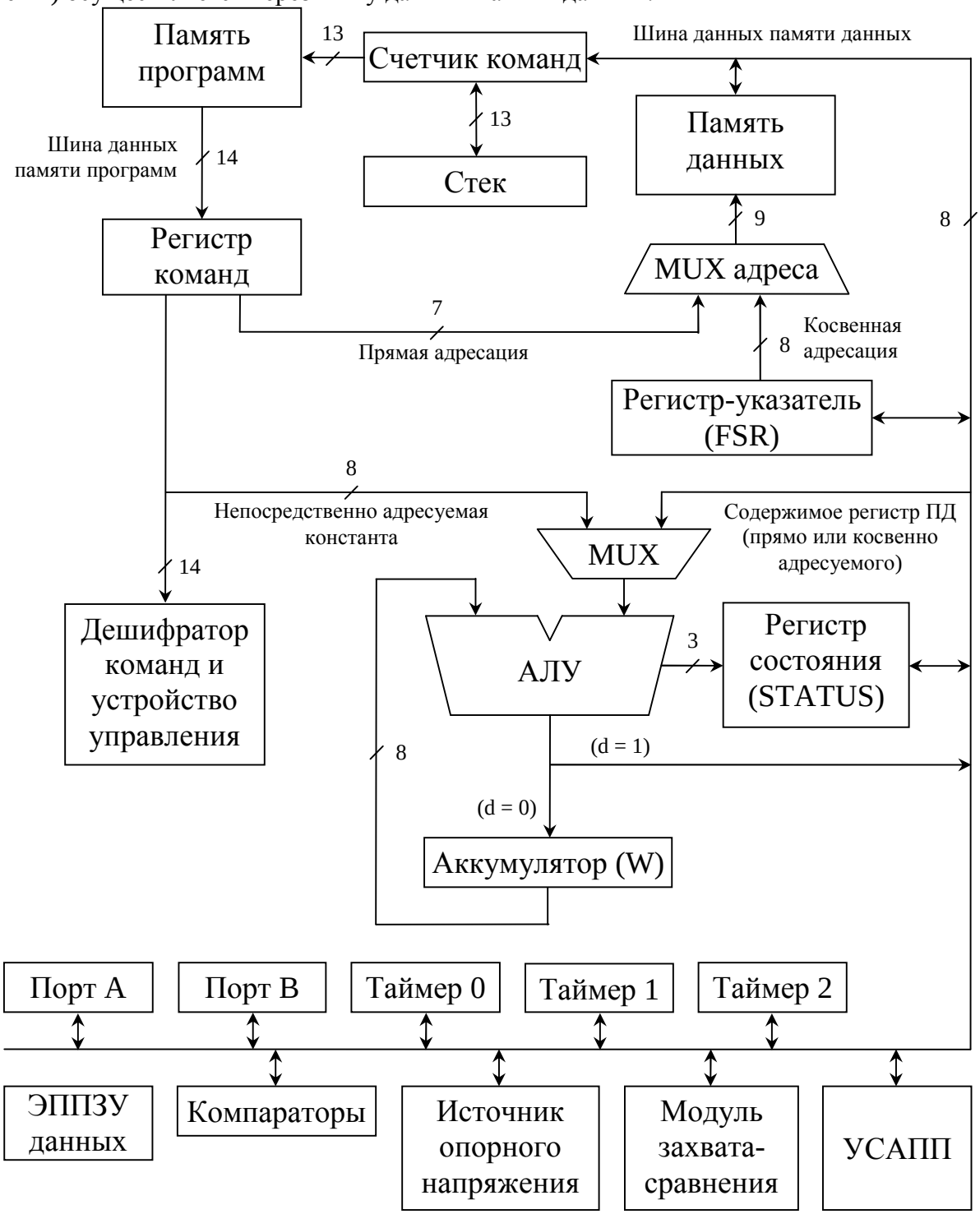

Рисунок 2.1 - Упрощенная структурная схема микроконтроллера PIC16F628

3 Микроконтроллер с точки зрения программиста

Составляя программу для микроконтроллера, программист абстрагируется от всего многообразия элементов микроконтроллера и имеет дело лишь с системой команд и ограниченным числом регистров. Эти регистры характеризуются тем, что их имена или номера (адреса) могут применяться в программе, а содержимое регистров может быть изменено или прочитано с помощью соответствующих команд.

Следовательно, с точки зрения программиста микроконтроллер представляет собой систему команд и набор регистров, которые каким-то образом связаны с остальными элементами контроллера. Совокупность регистров, доступных программисту, будем называть «программистской моделью контроллера».

Семейство МК PIC16 включает множество представителей, отличающихся своими возможностями. Привести общую модель, справедливую для всех МК семейства, сложно, поэтому здесь рассматривается один из самых современных и в то же время простых типов PIC16F628. Вся предлагаемая информация касается только этого типа. Другие микроконтроллеры этого семейства имеют некоторые отличия, но основные принципы организации остаются общими для всех МК семейства.

3.1 Организация памяти

Память МК PIC16 включает память программ, память данных и ЭППЗУ данных. При описании программисткой модели речь пойдет только о памяти данных, т. к. лишь к памяти данных возможен доступ непосредственно при помощи команд. Обращение к памяти программ в PIC16F628 вообще невозможно, а обращение к ЭППЗУ данных производится через связанные с ним регистры специальных функций, поэтому его скорее следует отнести к периферийным устройствам МК.

Для обозначения ячейки памяти данных в МК Microchip принято абстрактное определение «регистр». Все ячейки памяти называются регистрами независимо от выполняемых функций. Все регистры являются равноправными между собой. Т. е. любая команда, способная обращаться к регистру, может обратиться к любому регистру, но только в пределах текущей страницы ПД (см. ниже).

Каждый регистр имеет свой порядковый номер (адрес), который указывается в команде, для того чтобы ссылаться на регистр. ПД разделена на страницы по 128 адресов. В каждый момент времени выбрана одна из страниц, и, соответственно, возможно обращение к регистрам только выбранной (текущей) страницы.

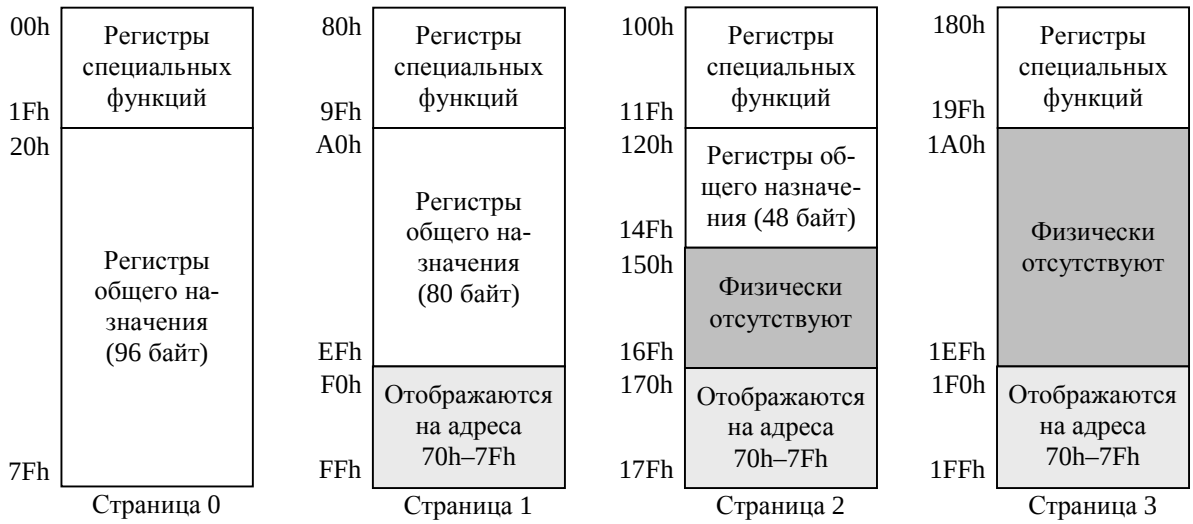

Рисунок 3.1 – Схема организации памяти данных в МК PIC16F628

Количество страниц ПД в PIC16F628 четыре, как и в большинстве МК этого семейства. В младших моделях, обладающих ПД меньшего объема, память разбита на две страницы.

Различают два типа регистров: регистры общего назначения (РОН) и регистры специальных функций (РСФ). РОН служат для хранения данных и могут использоваться по усмотрению программиста. РСФ предназначены для управления работой МК. Каждый РСФ связан с тем или иным узлом или периферийным устройством МК. Запись в такой регистр может производиться только с целью воздействия на устройство, с которым данный регистр связан. Тем не менее, как уже было сказано выше, все регистры равноправны. Обращение к РОН и РСФ производится одними и теми же командами.

Рассмотрим схему адресного пространства ПД в МК PIC16F628 (см. рисунок 5.1). На рисунке изображены четыре страницы. Каждая страница представляет собой блок из 128 адресов. Адреса на рисунке записаны в шестнадцатеричной форме. Регистры специальных функций распределены по всем четырем страницам и занимают младшие 32 адреса на каждой. Некоторые адреса ПД не связаны с физическими регистрами. Например, физически отсутствуют регистры с адресами с 150h по 16Fh. А адресам с F0h по FFh в действительности соответствуют регистры 70h–7Fh нулевой страницы памяти. Обозначенное пространство регистров специальных функций тоже на самом деле занято не полностью.

Разъясним назначение страниц памяти. Страничная организация памяти обусловлена разрядностью адресной части[1] команды (см. ниже), которая в свою очередь ограничивается длиной команды. Дело в том, что под адресную часть всех команд PIC16 отводятся семь двоичных разрядов. При такой разрядности возможно обращение лишь к 2^7=128 регистрам. Поэтому реальный адрес регистра складывается из семи бит, содержащихся в адресной части команды, и двухбитного номера страницы. В действительности РОН 20h и A0h имеют один и тот же адрес 20h, т. к. именно это число указывается в адресной части команды. Однозначно регистр идентифицируется адресной частью команды в совокупности с номером текущей страницы (см. рисунок 3.2). Общая разрядность равна девяти, что позволяет адресовать 2^9=512 регистров.

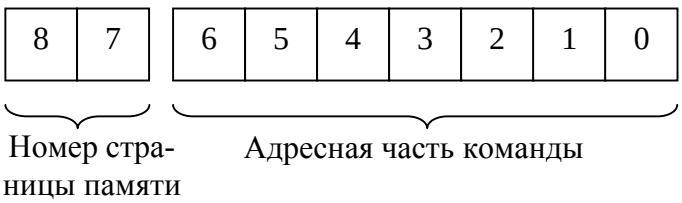

Рисунок 3.2 – Формирование адреса регистра

На рисунке 3.3 перечислены все регистры специальных функций, с указанием их адресов в памяти данных. Эти сведения будут необходимы в дальнейшем при составлении программ.

Среди РСФ следует особо выделить счетчик команд PCL (адрес 02h), регистр состояния STATUS (адрес 03h), регистр указатель FSR (адрес 04h), регистр косвенной адресации INDF (адрес 00h). Эти регистры наиболее часто используются при составлении программ. В этом разделе речь пойдет только о перечисленных регистрах специальных функций и об аккумуляторе, который к РСФ не относится. Другие регистры специальных функций, предназначенные в основном для управления периферийными устройствам МК, будут рассмотрены позже в разделах, посвященных соответствующим узлам МК.

3.2 Аккумулятор

Кроме РОН и РСФ имеется еще один программно доступный регистр — регистр-аккумулятор (рабочий регистр), называемый также регистр-W. Аккумулятор используется многими командами как источник одного из операндов и (или) приемник результата.

[1] См раздел 4.

	Страница 0		Страница 1		Страница 2		Страница 3
00	INDF	80	INDF	100	INDF	180	INDF
01	TMR0	81	OPTION	101	TMR0	181	OPTION
02	PCL	82	PCL	102	PCL	182	PCL
03	SATSUS	83	STATUS	103	STATUS	183	STATUS
04	FSR	84	FSR	104	FSR	184	FSR
05	PORTA	85	TRISA	105		185	
06	PORTB	86	TRISB	106	PORTB	186	TRISB
07		87		107		187	
08		88		108		188	
09		89		109		189	
0A	PCLATH	8A	PCLATH	10A	PCLATH	18A	PCLATH
0B	INTCON	8B	INTCON	10B	INTCON	18B	INTCON
0C	PIR1	8C	PIE1	10C		18C	
0D		8D		10D		18D	
0E	TMR1L	8E	PCON	10E		18E	
0F	TMR1H	8F		10F		18F	
10	T1CON	90		110		190	
11	TMR2	91		111		191	
12	T2CON	92	PR2	112		192	
13		93		113		193	
14		94		114		194	
15	CCPR1L	95		115		195	
16	CCPR1H	96		116		196	
17	CCP1CON	97		117		197	
18	RCSTA	98	TXSTA	118		198	
19	TXREG	99	SPBRG	119		199	
1A	RCREG	9A	EEDATA	11A		19A	
1B		9B	EEADR	11B		19B	
1C		9C	EECON1	11C		19C	
1D		9D	EECON2	11D		19D	
1E		9E		11E		19E	
1F	CMCON	9F	VRCON	11F		19F	

Рисунок 3.3 – Регистры специальных функций микроконтроллера PIC16F628

Система команд МК семейства PIC16 не имеет двухадресных команд — команд, имеющих два операнда. Если операция предполагает использование двух операндов (например, сложение), то один из них размещается в аккумуляторе. Аккумулятор не входит в адресное пространство ПД и адресуется только неявно[2], т. е. сам код операции подразумевает использование аккумулятора в качестве одного из операндов или в качестве приемника результата.

3.3 Счетчик команд

Счетчик команд содержит адрес выполняемой команды в памяти программ. Его содержимое используется аппаратурой МК для чтения команд из ПП. Счетчик команд автоматически увеличивается на единицу в начале каждого машинного цикла, т. е. фактически до выполнения команды. Поэтому, если команда производит считывание счетчика команд, то она получает адрес следующей команды.

[2] См раздел 5.

Счетчик команд в МК PIC16 13-разрядный, что позволяет адресовать до 2^{13}=8192 слова памяти программ. Младшие восемь бит счетчика команд доступны через регистр PCL (адрес 02h), старшие пять бит связаны с регистром PCLATH (адрес 0Ah). Регистр PCL полностью отождествляется с младшим байтом счетчика команд, т. е. запись в него приведет к немедленному изменению счетчика команд и, как следствие, к изменению порядка выполнения команд. Возможно также чтение младшего байта счетчика команд через регистр PCL. PCLATH является буфером-защелкой для старшей части счетчика команд. Значение из PCLATH переписывается в счетчик команд лишь при изменении PCL. Причем речь идет не об автоматическом изменении при выборке из ПП очередной команды, а о программном изменении при помощи команд передачи управления или записи в регистр PCL. Чтение старшего байта счетчика команд через PCLATH невозможно. Итак, постоянная связь между PCLATH и старшей частью счетчика команд отсутствует. Эта связь устанавливается лишь при программном изменении PCL и выражается в пересылке содержимого PCLATH в старшую часть счетчика команд.

Рисунок 3.3 – Счетчик команд

Кроме того, существует разница между изменением PCL командами пересылки и непосредственной записью в него значения из АЛУ, например, командой пересылки. В первом случае в счетчик команд переписываются лишь третий и четвертый биты PCLATH, а во втором случае переписываются все пять бит.

Такой сложный, на первый взгляд, механизм изменения счетчика команд совершенно необходим. Разрядность счетчика команд превышает разрядность АЛУ, и загрузка его новым значением невозможна одной командой. Загрузка должна производиться в два этапа: сначала, старшая часть, а затем младшая. Представим, что при изменении старшей части происходит фактическое изменение счетчика команд. Это приведет к немедленному изменению последовательности выполнения команд. Тогда следующая команда, записывающая младшую часть, выполнена не будет. Загрузка счетчика команд нужным значением становится невозможной.

Счетчик команд отображается на все страницы памяти, поэтому регистр PCL доступен по адресам 02h, 82h, 102h, 182h, а PCLATH — по адресам 0Ah, 8Ah, 10Ah, 18Ah.

3.4 Регистр состояния

Регистр состояния STATUS содержит флаги состояния АЛУ, флаги источника сброса и биты выбора станиц памяти. Регистр STATUS доступен на всех четырех страницах и имеет адреса 03h, 83h, 103h, 183h.

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
7	6	5	4	3	2	1	0

Рисунок 3.4 – Регистр состояния STATUS

Перечислим назначение битов регистра STATUS, начиная с младшего:

Бит 0 (C) — флаг переноса для команд сложения и инверсный флаг заема для команд вычитания. Бит устанавливается в «1» командами сложения (ADDWF, ADDLW), если в результате операции произошел перенос из старшего разряда (результат больше

0FFh); иначе сбрасывается в «0». Бит сбрасывается в «0» командами вычитания (SUBWF, SUBLW), если произошел заем; иначе устанавливается в «1». При выполнении операций сдвига операнда влево (RLF) и вправо (RRF) бит загружается старшим и младшим битами операнда, соответственно.

Бит 1 (DC) — флаг вспомогательного переноса[3]. Устанавливается в «1» при переносе из младшей тетрады (четыре бита) в старшую и при отсутствии заема из старшей тетрады. Иначе устанавливается в «0».

Бит 2 (Z) — флаг нулевого результата. Устанавливается в «1», если результат выполнения логической, арифметической операции или операции пересылки равен нулю.

Бит 3 ($\overline{PD}$), бит 4 ($\overline{TO}$) — флаги источника сброса. Позволяют определить условие, вызвавшее сброс микроконтроллера. Подробно рассматриваться не будут.

Бит 5 (RP0), бит 6 (RP1) — биты выбора страницы памяти данных. Таблица 5.1 устанавливает соответствие между значениями этих битов и номером текущей страницы памяти.

Таблица 3.1

RP1	RP2	Страница ПД №	Адреса ПД
0	0	0	000h–07Fh
0	1	1	080h–0FFh
1	0	2	100h–17Fh
1	1	3	180h–1FFh

Бит 7 (IRP) — бит выбора страниц ПД при косвенной адресации. IRP=0 — доступ к страницам 0, 1; IRP=1 — доступ к страницам 2, 3.

3.5 Стек

В общем случае *стек* представляет собой группу последовательно пронумерованных регистров (аппаратный стек) или ячеек памяти (программный стек). В каждый момент времени возможно обращение (запись или считывание информации) только к одной ячейке, называемой *вершиной стека*. Номер этой ячейки хранится в *указателе стека*. При записи указатель стека автоматически увеличивается на единицу, а при считывании — уменьшается. Заносимые в стек данные каждый раз помещается в следующую по порядку свободную ячейку. А извлекается последняя, но еще не считанная, поступившая в него информация. Данные, занесенные в стек последними будут считаны первыми.

Все микроконтроллеры PIC оснащены аппаратным стеком. В PIC16 стек восьмиуровневый, т. е. состоит из восьми ячеек. Стек PIC16 служит только для сохранения адресов возврата при вызове подпрограмм и возникновении прерываний. Стек программно недоступен, т. е. информация в нем не может быть изменена или прочитана какой либо командой.

Стек организован как циклический буфер. Это означает, что после того как будет записано восьмое (последнее) значение, вновь поступившее (девятое) значение будет записано на место первого, десятое на место второго и т. д. Стек не имеет флага переполнения.

4 Система команд PIC16

Команда представляет собой код, определяющий операцию микроконтроллера (в общем случае вычислительной машины) и данные, участвующие в операции. Код команды состоит из операционной и адресной частей. Адресная часть в свою очередь может состоять из нескольких полей.

При программировании на языке ассемблера каждой команде ставится в соответствие определенное *мнемоническое обозначение*. Мнемоническим обозначением называют

[3] В технической документации по семейству микроконтроллеров PIC16 приводится другое название этого бита — «бит десятичного переноса». Здесь употреблено название, принятое в литературе [5], для определения аналогичного бита MCS-51.

символическую запись, основанную на ассоциациях, призванную облегчить запоминание команд. Мнемоническими обозначениями служат сокращения английских слов, описывающих действия, выполняемые командами.

Операционная часть команды содержит код операции, который задает вид операции, действие, выполняемое МК.

Адресная часть команды содержит информацию об адресах операндов и адресе результата операции.

Система команд МК семейства PIC16 включает 37 инструкций. Из них две не рекомендуется использовать фирмой Microchip по соображением совместимости с новыми МК. Все команды PIC16 можно разделить на следующие группы: операции пересылки, арифметические операции, логические операции, битовые операции, команды передачи управления[4].

Система команд PIC-контролеров имеет следующую отличительную особенность. Большинство команд, так или иначе обрабатывающих содержимое регистров ПД, позволяет выбрать один из двух приемников результата. Результат можно направить либо в тот же регистр, который служит одним из операндов (или единственным операндом), либо в аккумулятор. В таких командах один бит отводится под *модификатор*, показывающий, куда следует поместить результат. Если модификатор равен «1», результат помещается в тот же регистр, перезаписывая операнд, если модификатор равен «0», то результат помещается в аккумулятор. На языке ассемблера, реализованном в интегрированной среде разработки MPLAB, допускается вместо нулей и единиц использовать буквенные обозначения «w» и «f» (соответственно). Такая запись боле наглядна.

Обычно результат направляют в аккумулятор в том случае, если он используется как операнд в следующей операции.

Общепринятые мнемонические обозначения команд, их словесные и символические описания приведены в таблице 4.1.

Рассмотрим обозначения, используемые в таблице.

f — 7-разрядный адрес регистра ПД. $0 \le f \le 127$.

k — 8-разрядная константа (число). $0 \le k \le 255$.

k_{11} — 11-разрядная константа (адрес ПП). $0 \le k_{11} \le 2047$.

b — 3-разрядный номер бита регистра. $0 \le b \le 7$.

d — модификатор команды. Показывает, следует ли результат операции поместить в тот же регистр (d=1) или в аккумулятор (d=0).

f′ — 3-х разрядный адрес TRIS-регистра. $5 \le f' \le 7$.

В столбце «Операция» используются следующие обозначения:

→ — обозначает операцию пересылки.

dest — обозначает регистр-приемник. В зависимости от значения модификатора d, это может быть регистр ПД (f) или аккумулятор (W).

W — аккумулятор.

PC — счетчик команд

OPTION — РСФ OPTION

TOS — указатель стека

GIE — общий флаг разрешения прерываний.

Квадратные скобки показывают, что действие выполняется не над содержимым регистра в целом, а над одним или несколькими его битами. Остальные обозначения являются общепринятыми.

[4] Классификация команд отличается от приведенной в техническом описании PIC16 и соответствует классификации используемой в литературе [5].

Таблица 4.1 – Система команд МК семейства PIC16

Мнемоническое обозначение	Описание	Операция	Число машинных циклов	Изменяемые флаги состояния
		Группа команд пересылки		
MOVF f, d	Пересылка регистра (f) (в аккумулятор)	(f) → (dest)	1	Z
MOVWF f	Пересылка аккумулятора в регистр (f)	(W) → (f)	1	
MOVLW k	Запись константы (k) в аккумулятор	k → (W)	1	
		Группа арифметических команд		
ADDWF f, d	Сложение регистра (f) с аккумулятором	(W)+k → (dest)	1	C, DC, Z
ADDLW k	Сложение константы (k) с аккумулятором	(W)+k → (W)	1	C, DC, Z
SUBWF f, d	Вычитание аккумулятора из регистра (f)	(f)–(W) → (dest)	1	C, DC, Z
SUBLW k	Вычитание аккумулятора из константы (k)	(k)–(W) → (W)	1	C, DC, Z
INCF f, d	Инкремент регистра (f)	(f)+1 → (dest)	1	Z
DECF f, d	Декремент регистра (f)	(f)+1 → (dest)	1	Z
		Группа логических команд		
ANDWF f, d	Побитное лог. И регистра (f) и аккумулятора	$(f) \wedge (W) \rightarrow (dest)$	1	Z
ANDLW k	Побитное лог. И константы (k) и аккумулятора	$(k) \wedge (W) \rightarrow (W)$	1	Z
IORWF f, d	Побитное лог. ИЛИ регистра (f) и аккумулятора	$(f) \vee (W) \rightarrow (dest)$	1	Z
IORLW k	Побитное лог. ИЛИ константы (k) и аккумулятора	$(k) \vee (W) \rightarrow (W)$	1	Z
XORWF f, d	Побитное искл. ИЛИ регистра (f) и аккумулятора	$(f) \oplus (W) \rightarrow (dest)$	1	Z
XORLW k	Побитное искл. ИЛИ константы (k) и аккумулятора	$(k) \oplus (W) \rightarrow (W)$	1	Z
RLF f, d	Циклический сдвиг регистра (f) влево через перенос	См. описание	1	C
RLF f, d	Циклический сдвиг регистра (f) вправо через перенос	См. описание	1	C
CLRW	Очистка аккумулятора	0 → (W)	1	
CLRF f	Очистка регистра (f)	0 → (f)	1	Z
COMF f, d	Побитное лог. инверсия регистра (f)	$\overline{(f)} \rightarrow (f)$	1	Z
SWAPF f, d	Обмен местами тетрад регистра (f)	(f[0..3]) → (f[4..7]), (f[4..7]) → (f[0..3])	1	

Продолжение таблицы 4.1

Мнемоническое обозначение	Описание	Операция	Число машинных циклов	Изменяемые биты состояния
		Группа битовых операций		
BSF f, b	Установить в «1» заданный бит (b) регистра (f)	1 → (f[b])	1	
BCF f, b	Сбросить в «0» заданный бит (b) регистра (f)	0 → (f[b])	1	
		Группа команд управления		
GOTO k_{11}	Безусловный переход по адресу (k_{11})	k_{11} → PC[0..10], PCLATH[3..4] → PC[11..12]	2	
CALL k_{11}	Вызов подпрограммы по адресу (k_{11})	k_{11} → PC[0..10], PCLATH[3..4] → PC[11..12], PC+1 → TOC	2	
RETURN	Возврат из подпрограммы	TOS → PC	2	
RETLW k	Возврат из подпрограммы с загрузкой константы (k) в аккумулятор	TOS → PC, k → (W)	2	
RETFIE	Возврат из процедуры обработки прерываний	TOS → PC, 1 → GIE	2	
INCFSZ f, d	Инкремент регистра (f), пропустить следующую команду, если результат равен нулю	(f)+1 → (dest), Если результат =0, PC+1 → PC	1 или 2	
DECFSZ f, d	Декремент регистра (f), пропустить следующую команду, если результат равен нулю	(f)–1 → (dest), Если результат =0, PC+1 → PC	1 или 2	
BTFSS f, b	Пропустить следующую команду, если заданный бит (b) регистра (f) равен единицы	Если f[b]=1, PC+1 → PC	1 или 2	
BTFSC f, b	Пропустить следующую команду, если заданный бит (b) регистра (f) равен нулю	Если f[b]=0, PC+1 → PC	1 или 2	
SLEEP	Перевод контроллера в ждущий режим	См. описание	1	$\overline{TO}$, $\overline{PD}$
NOP	Холостая операция		1	
CLRWDT	Очистка сторожевой таймер	См. описание	1	$\overline{TO}$, $\overline{PD}$
OPTION	Пересылка аккумулятора в регистр OPTION	(W) → OPTION	1	
TRIS f′	Пересылка аккумулятора в заданный TRIS-регистр (f′)	(W) → (f′)	1	

4.1 Группа команд пересылки

Группа команд пересылки служит для записи константы в аккумулятор и обмена данными между регистром ПД и аккумулятором. Одна из команд — MOVF влияет на флаг нуля (Z). В этом состоит отличие группы команд пересылки PIC-контроллеров, например, от микроконтроллеров семейства MCS-51, в которых команды пересылки не изменяют регистр состояния. С помощью команды MOVF можно протестировать регистр на нулевое содержимое. В других контроллерах для этой цели служит специальная команда.

4.2 Группа арифметических команд

Группа содержит команды двух арифметических действий: сложение и вычитание; и команды выполняющие инкремент и декремент регистров, т. е. увеличение и уменьшение на единицу. Команды, реализующие сложение и вычитание, влияют на все три флага состояния. Команды INCF и DECF, не изменяют битов переноса. Отследить переполнение, вызванное этими командами невозможно.

4.3 Группа логических команд

К этой же группе относятся:

1. Команды, побитно выполняющие логические операции над регистрами, а именно: «И» (конъюнкция), «ИЛИ» (дизъюнкция), «НЕ» (инверсия), «исключающее ИЛИ» (неравнозначность).
2. Команды сброса (очистки) регистра ПД (CLRF) и аккумулятора (CLRW).
3. Команда, осуществляющая обмен местами двух полубайтов (тетрад) регистра ПД (SWAPF).
4. Команды сдвигов. Сдвиг регистра производится через перенос. При сдвиге влево старший (седьмой) бит заносится в бит переноса (C), биты с первого по шестой смещаются влево, а освободившийся младший (нулевой) бит заполняется предыдущим значением бита переноса. Сдвиг вправо происходит аналогичным образом. Большая часть логических команд влияет на флаг нуля (Z). Команды сдвига через перенос, естественно, влияют на флаг переноса (C).

4.4 Группа битовых операций

Содержит лишь две команды, которые позволяют установить один произвольный бит регистра ПД в «1» или сбросить его в «0». Битовые команды не изменяют флагов состояния. Необходимо учитывать, что биты нумеруются от нуля до семи, начиная с младшего (справа на лево).

4.5 Группа команд передачи управления

Группа включает инструкции, предназначенные для изменения порядка выполнения команд управления режимами работы МК.

1. Команда безусловного перехода (GOTO). Команда передает управление в заданную точку программы. Операндом команды служит 11-разрядный адрес ПП, указывающий на точку перехода. Разрядность операнда обуславливает интервал адресов «охватываемых» командой. Переход возможен только в пределах блока ПП из 2^{11}=2048 слов. Для МК PIC16F628 этого вполне достаточно, т. к. объем ПП составляет именно 2048 слов. В других МК для адресации всех 8129 слов ПП требуется 13 разрядов. В этом случае 11 разрядов, содержащиеся в поле операнда команды GOTO, дополняются двумя разрядами регистра PCLATH — PCLATH[3..4]. Для «длинного» перехода необходимо сперва загрузить в PCLATH старшие биты адреса перехода, а затем воспользоваться команду GOTO. При выполнении команды GOTO младшая часть счетчика команд загружается числом из адресной части команды, а старшие два бита берутся из PCATH.

Иначе говоря, можно представить память программ, разбитой на страницы по 2048 слов. Тогда биты 3, 4 регистра PCLATH показывают номер страницы, а операнд команды GOTO — адрес внутри страницы.

2. Команда вызова подпрограммы (CALL). Команда служит для передачи управления в подпрограмму с возможностью возврата. Работа команды аналогична команде GOTO, только CALL еще сохраняет в аппаратном стеке адрес возврата. Адрес возврата — это адрес следующей команды, равный содержимому счетчика команд, увеличенному на единицу.

3. Команда возврата из подпрограммы (RETURN). Команда завершает работу подпрограммы. RETURN извлекает из стека адрес возврата и помещает его в счетчик команд. Управление передается на команду, следующую за командой CALL, вызвавшей подпрограмму. Аппаратный стек PIC-контроллера 13-разрядный, поэтому адрес возврата помещается в него целиком. Таким образом, возврат возможен в любую точку ПП.

4. Команда возврата с загрузкой константы в аккумулятор (RETLW). Работа команды полностью аналогична команде RETURN, но RETLW еще загружает в аккумулятор константу, указанную в адресной части команды. RETLW применяется для передачи параметров из подпрограмм в основную программу и для организации приема, называемого табличного хранения данных в памяти программ.

5. Команда возврата из процедуры обработки прерываний. Работа команды аналогична команде RETURN, но RETFIE устанавливает глобальный бит разрешения прерываний GIE (бит 7 регистра INTCON). Назначение команды будет разъяснено в разделе 10.

DECFSZ — команда уменьшает на единицу (декрементирует) указанный регистр, а далее сравнивает результат с нулем. Если результат равен нулю, то следующая команда пропускается, иначе порядок выполнения команд не изменяется. Пропуская следующую команду, МК выполняет один холостой цикл. В этом случае команда выполняется за два машинных цикла. INCFSZ — то же, но регистр увеличивается на единицу (инкрементируется). Команды применяются дл организации циклов.

6. BTFSS — анализирует значение заданного бита регистра. Если этот бит равен «1» выполнение следующей команды пропускается. В случае пропуска выполняется один холостой машинный цикл. BTFSC — следующая команда пропускается, если бит равен «0».

7. SLEEP — переводит МК в ждущий режим. В ждущем режиме тактовый генератор остановлен, выполнение программы приостанавливается. В таком состоянии микроконтроллер потребляет ток менее 10 мкА. Состояние всех регистров ПД остается без изменений, сохраняются логические уровни на выводах МК. МК выводится из режима ожидания аппаратным сбросом или по прерыванию от периферийного устройства (см. раздел 10). SLEEP переводит МК в ждущий режим не сразу. Прежде чем приостановить свою работу МК выполняет еще одну команду. Тактовый генератор останавливается через один машинный цикл после команды SLEEP.

8. NOP — холостая операция. Используется для организации задержки длительностью в один машинный цикл.

9. CLRWDT — сброс сторожевого таймера.

10. OPTION — загружает регистр специальных функций OPTION значением из аккумулятора (использовать не рекомендуется).

11. TRIS — загружает указанный TRIS-регистр (см. раздел 8) значением из аккумулятора (использовать не рекомендуется).

5 Методы адресации

Исполнительным адресом называется номер регистра (в общем случае ячейки памяти), к которой производится фактическое обращение.

Методом адресации называется способ формирования исполнительного адреса.

В вычислительной технике находят применение множество методов адресации. В МК семейства PIC16 используются всего четыре метода.

Неявная адресация. В команде не содержится явных указаний на адрес операнда или адрес, по которому помещается результат операции. Операнд или его адрес подразумевается и фактически задается кодом операции.

Непосредственная адресация. В адресной части команды содержится не адрес операнда, а непосредственно сам операнд.

Прямая адресация. Исполнительный адрес совпадает с адресной частью команды.

Косвенная адресация. Адресная часть команды содержит адрес регистра, в котором находится исполнительный адрес операнда.

Приведем несколько примеров. Команда CLRW (очистка аккумулятора) неявно адресует аккумулятор — адресная часть вообще отсутствует. MOVLW использует непосредственную адресацию для задания константы. В команде MOVWF регистр адресуется прямым методом, хотя может адресоваться и косвенно. В то же время аккумулятор адресуется неявно. Косвенная адресация будет рассмотрена позднее.

6 Ассемблер PIC-контроллеров

Язык ассемблер характеризуется алфавитом, словарем и синтаксическими правилами. В алфавит входят буквы латинского алфавита, арабские цифры, знаки препинания и некоторые служебные символы. Различают постоянный и переменный словарь. В постоянный словарь входят мнемонические обозначения команд и директивы ассемблера.

Понятие мнемонического обозначения было разъяснено в разделе 4. Директива ассемблера представляет собой некоторое предписание, указание ассемблеру как транслятору. Директивы управляют процессом преобразования исходного текста программы в объектный код. Они не имеют непосредственной связи с системой команд МК и сами не преобразуются в объектный код, не записываются в память программ контроллера. Различным версиям ассемблера могут быть присущи разные языки директив, даже если они поддерживают один и тот же тип контроллеров. В то время как мнемонические обозначения однозначно соответствуют командам МК и преобразуются ассемблером в объектный код. Набор мнемонических обозначений, распознаваемых ассемблером, определяется системой команд конкретного контроллера.

Переменный словарь составляется программистом и содержит произвольные буквенные имена меток и регистров. Такие символические имена облегчают визуальное восприятие программы.

Программы на ассемблере принято записывать в четыре колонки. Каждая строка состоит из четырех полей: поля меток, операций, операндов и комментариев.

Метка представляет собой символическое обозначение адреса команды. Это поле является необязательным и используется в том случае, если в данную точку программы необходимо переходить командами передачи управления.

Поле операции содержит мнемоническое обозначение операции. Поле соответствует операционной части команды, обозначает код операции.

Поле операндов может содержать один или несколько операндов перечисленных через запятую. Соответствует адресной части команды. Некоторые команды не имеют адресной части. Такие команды называются безадресными.

Комментарий начинается с точки с запятой «;» и содержит произвольный текст, поясняющий работу команды. Поле является необязательным, программа снабжается комментариями по усмотрению программиста.

6.1 Мнемонические обозначения и директивы ассемблера

Мнемонические обозначения команд записываются в точном соответствии с таблицей 4.1. Ассемблер не различает строчных и прописных букв в мнемокодах и директивах. Рассмотрим наиболее часто используемые директивы ассемблера.

EQU — устанавливает соответствие между символическим обозначением и числом. Используется для ввода элементов переменного словаря, например, для объявления имен регистров общего назначения или специальных функций.

Формат: *Имя* **EQU** *Константа*

ORG — задает адрес команды в ПП. Команда, следующая за директивой, размещается в ПП по указанному в директиве адресу. Все последующие команды размещаются за ней по порядку.

Формат: **ORG** *Адрес ПП*

END — обозначает конец программы. Текст поле END игнорируется ассемблером. Обязательно должна присутствовать в конце любой программы.

Формат: **END**

6.2 Запись элементов переменного словаря

Имена, вводимые программистом, могут состоять из букв латинского алфавита, цифр, символа подчеркивания «_» и не должны начинаться с цифры. Ассемблер различает регистр имен переменного словаря, поэтому записи Reg1 и REG1 будут интерпретироваться как различные.

6.3 Запись чисел

В ассемблере PIC-контроллеров возможно использование четырех форм записи чисел:

1. Шестнадцатеричная форма. Используются шестнадцатеричные цифры: 0, 1, 2,…, 9, A, B,…, F. Шестнадцатеричное число должно начинаться только с десятичной цифры 0, 1, 2,…, 9. Если старшим разрядом шестнадцатеричного числа является буква A, B,…, F, то перед числом необходимо поставить цифру 0.

Пример: **1F** — верно, **B4** — неверно (число начинается с буквы B), **0B4** — верно.

Возможен другой способ записи шестнадцатеричного числа. Число начинается с признака системы счисления — в данном случае с латинской буквы «H», за ней записывается шестнадцатеричное число, заключенное в апострофы (одинарные кавычки) «**'**». При этом число может начинаться с любой шестнадцатеричной цифры, начинать число с нуля не обязательно. Записи **0B4** и **H'B4'** абсолютно идентичны.

2. Двоичная форма. При этом запись числа начинается с латинской буквы B, за ней следуют восемь двоичных разрядов в апострофах.

Пример: **B'00101101'**.

3. Десятичная форма. Десятичное число начинается с точки «**.**», после которой следуют десятичные цифры. Возможен и другой вариант: число записывается в апострофах после обозначения системы счисления — буквы «D». Записи **.37** и **D'37'** означают одно и то же.

Восьмеричная форма. Число записывается в апострофах при помощи восьмеричных цифр 0, 1,…, 7. Ему предшествует буква «O». Восьмеричная система счисления практически никогда не используется.

Пример: **O'75'**.

7 Простейшие приемы программирования

7.1 Пример программы

Рассмотрим пример простейшей программы, выполняющей запись константы в регистр и пересылку его содержимого в другой регистр. Здесь все строки программы пронумерованы, чтобы ссылаться на них в тексте. На самом деле при составлении программы номера строк не ставятся.

Пример 1:

```
1.          reg1    equ     20  ;Присвоить регистру 20h имя reg1
2.          reg2    equ     21  ;Присвоить регистру 21h имя reg2

3.          movlw   .40         ;Поместить в аккумулятор (W) число 40
4.          movwf   reg1        ;Переписать содержимое W в регистр reg1
```

```
5.          movf    reg1, w     ;Поместить содержимое reg1 в W
6.          movwf   reg2        ;Переписать содержимое W в регистр reg2

7.          end                 ;Конец программы
```

Программа начинается с объявления имен переменного словаря. Вводятся символические обозначения регистров ПД. Регистру с адресом 20h ставится в соответствие имя reg1. Теперь вместо слова reg1 автоматически будет подставляться адрес 20h. Аналогично регистру 21h ставится в соответствия имя reg2.

Собственно программа начинается с третьей строки. Команда №3 записывает в аккумулятор число 40. Команда №4 пересылает значение аккумулятора в регистр reg1. Таким образом, комбинация команд 3 и 4 осуществляет запись числа 40 в регистр reg1. Команда №5 копирует содержимое регистра reg1 в аккумулятор. Операндом здесь служит имя регистра. После операнда через запятую указан модификатор «w» (или «0») , показывающий, что содержимое регистра необходимо переслать в аккумулятор. Шестая команда аналогична четвертой. Директива END в седьмой строке обозначает конец программы.

7.2 Безусловные переходы

Для обозначения точки, в которую необходимо выполнить переход, используются метки. Метка представляет собой произвольную последовательность символов, подчиняющуюся правилам составления переменного словаря. Метка обычно предшествует команде, т. е. располагается в поле меток, но может размещаться и на отдельной строке. Каждая метка должна быть уникальна, в то же время, в программе допускается любое количество различных меток.

Для выполнения безусловного перехода служит команда GOTO. Команда имеет единственный операнд — имя метки.

Для иллюстрации использования меток воспользуемся тем же примером. Рассмотренная ранее программа будет выполнена практически мгновенно. Чаще всего требуется, заставить программу работать непрерывно (вечно), периодически повторяя определенную последовательность действий. Для этого в начале программы помещается метка, а в конце — команда перехода на эту метку. Таким образом, после выполнения всей последовательности операций программа начнет свою работу заново.

Пример 2:

```
1.          reg1    equ     20  ;Присвоить регистру 20h имя reg1
2.          reg2    equ     21  ;Присвоить регистру 21h имя reg2

3. label    movlw   .40         ;Поместить в аккумулятор (W) число 40
4.          movwf   reg1        ;Переписать содержимое W в регистр reg1

5.          movf    reg1, w     ;Поместить содержимое reg1 в W
6.          movwf   reg2        ;Переписать содержимое W в регистр reg2

7.          goto    label       ;Перейти на команду с меткой «label»

8.          end                 ;Конец программы
```

7.3 Ветвления

При разработке программ часто бывает необходимо организовать ту или иную реакцию программы на какое-либо событие. Таким событием может быть нажатие кнопки, истечение определенного интервала времени и т. п. В любом случае задача сводится к анализу значения регистра, которое изменяется при наступлении события, и принятию решения. Если событие наступило, выполняется один участок (ветвь) программы, если нет, то другая ветвь.

Для проверки условий используются команды BTFSS и BTFSC.

Пример:

```
            ...
1.          btfss   reg, 5
```

```
2.          movwf   reg1
3.          ...
```

При выполнении команды №1 проверяется, равен ли единице бит 5 регистра reg. Если это условие выполняется, то управление передается на команду №3, а команда №2 пропускается. Иначе выполняется команда №2, затем №3 и т. д.

Если необходимо пропустить не одну, а сразу несколько команд, то приходится совмещать команду ветвления и команду безусловного перехода GOTO:

```
            ...
1.          btfss   reg, 5
2.          goto    label1

3.          movwf   reg2
4.          ...
5.          movwf   reg8

6.  label1  ...
```

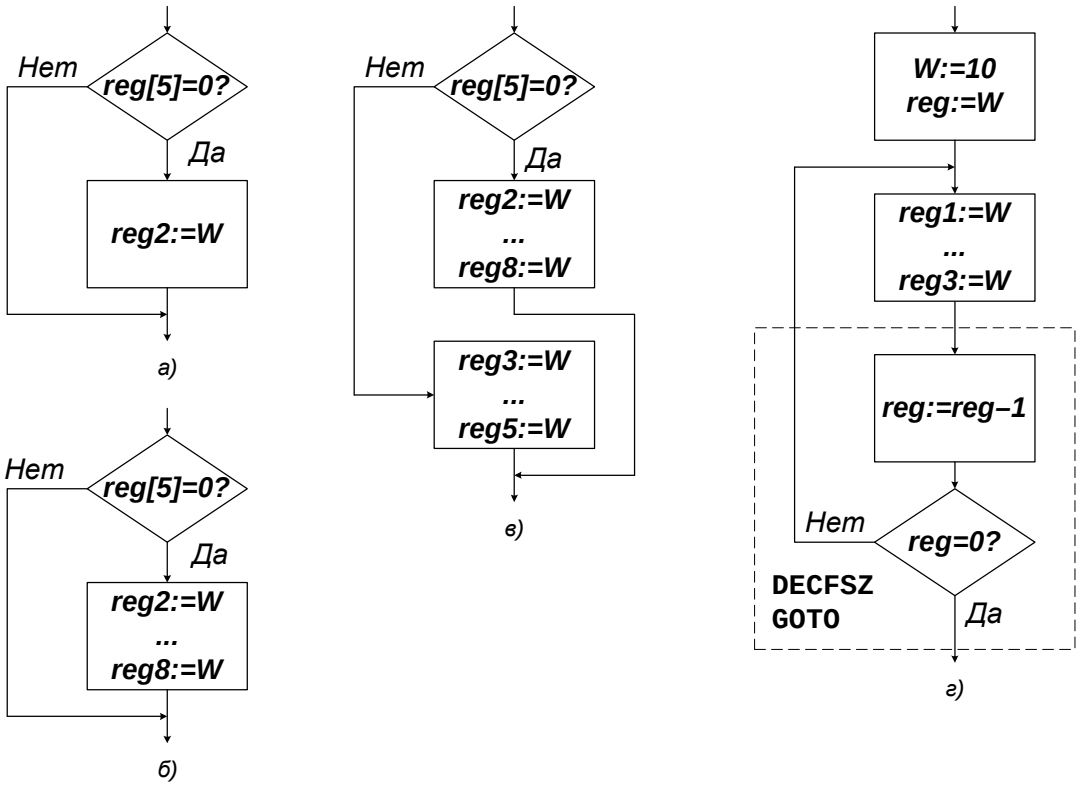

Рисунок 7.1 – Схемы программ организации ветвлений и цикла

Здесь если бит 5 регистра reg равен нулю, то команда №3 пропускается, выполняются команды №3, 4, 5, 6. Иначе, после проверки условия выполняется команда GOTO, происходит переход по метке label1; управление передается на команду №6, а блок команд №4, 5 пропускается.

Можно организовать и более сложную конструкцию с двумя ветвями:

```
            ...
1.          btfss   reg, 5
2.          goto    label1

3.          movwf   reg2
4.          ...
5.          movwf   reg8

6.          goto    label2

7.  label1  movwf   reg3
8.          ...
```

```
9.          movwf   reg5

10. label2  ...
```

7.4 Циклы

Конструкции, обеспечивающие многократный повтор выполнения одной или нескольких команд, называются *циклами*. Фрагмент программы, повторяющийся многократно в цикле, называется *телом цикла*.

При организации циклов возможны два случая: число повторов известно заранее, и число повторов заранее неизвестно и определяется во время выполнения цикла. В первом случае для подсчета числа повторов используется регистр-счетчик. В качестве счетчика должен использоваться любой РОН.

Для организации цикла больше всего подходит команда DECFSZ. Операнд команды — регистр-счетчик, в который предварительно заносится число повторов.

Пример использования цикла:

```
            ...
1.          movlw   .10
2.          movwf   reg

3.  label1  movwf   reg1
4.          ...
5.          movwf   reg3

6.          decfsz  reg, 1
7.          goto    label1

8.          ...
```

Здесь счетчиком служит регистр reg. Команды №1, 2 присваивают счетчику начальное значение — количество повторов. В данном случае это число 10. Далее следует тело цикла (команды №3, 4, 5), которое начинается с метки. При выполнении команды №6, во-первых, происходит уменьшение содержимого счетчика на единицу, во-вторых, его значение сравнивается с нулем. Если содержимое регистра reg не равно нулю, происходит возврат к началу тела цикла через команду безусловного перехода (команда №7). Выполнение последовательности команд №4–8 повторяется до тех пор, пока счетчик не примет нулевого значения. Когда reg становится равен нулю, управление передается команде №8 (команда №7 пропускается). На этом выполнение цикла завершается и продолжается выполнение оставшейся части программы.

8 Порты ввода-вывода

Обмен данными с универсальными микропроцессорами осуществляется по трем шинам: шине данных, шине адреса и шине управления. Шины связывают между собой элементы микропроцессорной системы и не предназначены для прямого воздействия на объект управления. В отличие от универсальных МП микроконтроллеры для ввода и вывода информации используют параллельные порты. Чаще всего линии портов могут быть подключены к объекту управления посредством лишь усилительных устройств без какой-либо логической обработки.

Порты предназначены для ввода и вывода цифровой информации в параллельном коде. Каждый порт представляет собой набор цифровых, как правило, двунаправленных линий. С точки зрения аппаратуры порт — это входные буферы и регистр-защелка, управляющий выходными транзисторами. Для смены режима работы линии (ввод или вывод) имеется схема, запирающая выходные транзисторы независимо от содержимого регистра-защелки. При этом линия переходит в высокоимпедансное состояние (третье состояние).

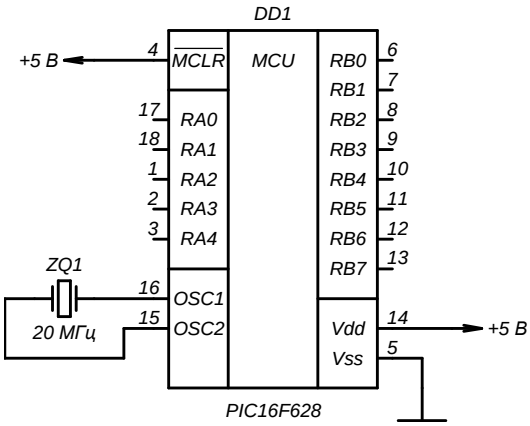

Рисунок 8.1 – Схема подключения источника питания и кварцевого резонатора к МК PIC16F628

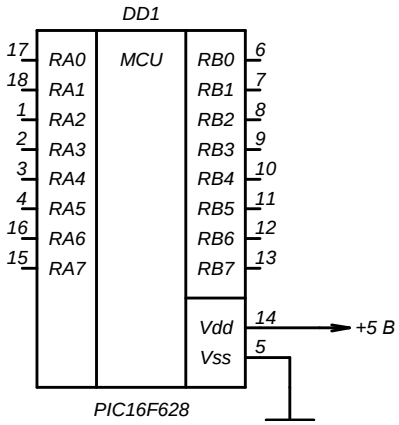

Рисунок 8.2 – Схема подключения МК PIC16F628 в режиме внутреннего резонатора и при отключенном входа сброса

Порты PIC-контроллеров обозначаются латинскими буквами A, B, C и т. д. PIC16F628 имеет два восьмиразрядных порта: порт A и порт B.

Регистры-защелки и входные буферы портов связаны с регистрами специальных функций PORTA и PORTB. Запись в эти регистры приводит к записи в регистр-защелку и соответствующему изменению логических уровней на линиях порта. Результатом считывания регистров PORTA и PORTB является состояние входных буферов порта, соответствующее логическим уровням на его линиях. Например, бит 0 регистра PORTA связан с выводом 0 порта A (обозначается RA0, см таблицу 8.1 и рисунок 8.1). Запись единицы в этот бит приведет к появлению высокого логического уровня на выводе RA0, запись нуля — к появлению низкого логического уровня. Аналогично производится считывание информации с линий порта, настроенных на ввод.

Можно изменять логические уровни на всех выводах порта одновременно или только на одном из них, оставив остальные без изменений. Все зависит от того, какая команда обращается к порту — байтовая или битовая.

Направление передачи информации тоже настраивается побитно, т. е. устанавливается индивидуально для каждой линии порта. Выбор режима работы портовых линий производится через регистры TRISA (85h) и TRISB (86h). Каждый бит TRIS-регистра соответствует линии одноименного порта. Запись нуля в какой-либо бит TRIS-регистра настраивает соответствующую линию на вывод, запись единицы — на ввод.

Входные буферы порта A представляют собой триггеры Шмидта, порта B — ТТЛ-буферы. Поэтому порт A по входам совместим только с уровнями КМОП, а PORTB — с КМОП и ТТЛ.

Ниже приведены электрические характеристики портовых линий в режиме вывода, при напряжении питания 5 В, токе, протекающем через вывод 5 мА и температуре окружающей среды 25ºС:

Минимальный уровень логической единицы, не менее	4,5 В
Максимальный уровень логического нуля, не более	0,6 В

Электрические характеристики выводов, настроенных на ввод, при напряжении питания 5 В и температуре окружающей среды 25ºС:

Минимальное напряжение логической единицы, не менее	
для порта A	4 В
для порта B	2 В
Максимальное напряжение логического нуля, не более	
для порта A	1 В
для порта B	0,8 В

Максимальный ток через каждый вывод составляет 25 мА. При этом выходное напряжение высокого уровня может снижаться примерно до 2,6 В, а низкого уровня может достигать 0,85 В. Такие изменения выходных уровней необходимо учитывать при работе на нагрузку с большим потреблением тока [7].

Потребление тока входами микроконтроллера при воздействии сигнала высокого уровня не превышает 0,5…1 мкА.

Все порты имеют на входах диодные ограничители напряжения, максимальный допустимый ток через ограничительный диод составляет 20 мА. Это позволяет подключать непосредственно (через ограничивающий резистор) к портовым ножкам источники сигнала, с выходным напряжением гораздо больше напряжения питания контроллера, например операционные усилители.

Кроме основной функции (цифровой вход-выход), линии портов могут выполнять одну или несколько альтернативных функций, например, являться входом или выходом какого-либо периферийного устройства МК. Включение альтернативных функций производится через РСФ, управляющие периферийным устройством, с которым данная ножка связана.

В таблице 8.1 приведены все функции, которые могут выполнять линии портов МК PIC16F628.

Таблица 8.1 –

№ вывода	№ линии порта	Обозначение функций	Описание
Порт A			
17	0	RA0	Цифровой вход-выход
		AN0	Инвертирующий вход компаратора 1
18	1	RA1	Цифровой вход-выход
		AN1	Инвертирующий вход компаратора 2
1	2	RA2	Цифровой вход-выход
		AN2	Неинвертирующий вход компаратора 2
		VREF	Выход источника опорного напряжения
2	3	RA3	Цифровой вход-выход
		AN3	Неинвертирующий вход компаратора 1
		CMP1	Выход компаратора 1
3	4	RA4	Цифровой вход, выход с открытым стоком
		T0CKI	Счетный вход таймера-счетчика 0
		CMP2	Выход компаратора 2
4	5	RA5	Цифровой вход
		MCLR	Вход аппаратного сброса
		THV	
15	6	RA6	Цифровой вход-выход
		OSC2	Вывод для подключения кварцевого резонатора
		CLKOUT	Выход тактового генератора
16	7	RA7	Цифровой вход-выход
		OSC1	Вывод для подключения кварцевого резонатора
		CLKIN	Вход для подключения внешнего источника тактовых импульсов
Порт B			
7	0	RB0	Цифровой вход-выход
		INT	Вход запроса прерывания
8	1	RB1	Цифровой вход-выход
		RX	Вход приемника УСАПП в асинхронном режиме
		DT	Вход-выход данных УСАПП в синхронном режиме

9	2	RB2	Цифровой вход-выход
		TX	Выход передатчика УСАПП в асинхронном режиме
		CK	Вход-выход тактовых импульсов УСАПП в синхронном режиме
10	3	RB3	Цифровой вход-выход
		CCP1	Вывод блока захвата-сравнения, выход ШИМ
11	4	RB4	Цифровой вход-выход
		PGM	
12	5	RB5	Цифровой вход-выход
13	6	RB6	Цифровой вход-выход
		T1OSO	Выход тактового генератора таймера 1
		T1CKI	Счетный вход таймера 1
14	7	RB7	Цифровой вход-выход
		T1OSI	Вход тактового генератора таймера 1

9 Блок таймеров-счетчиков

Блок таймеров-счетчиков применяется для слежения за реальным временем либо для подсчета внешних событий. Таймер-счетчик представляет собой программируемый суммирующий счетчик с возможностью выбора источника счетных импульсов. Таймер-счетчик связан с регистрами общего назначения, через которые можно получить доступ к информации, хранимой в нем, и управлять его работой.

В зависимости от источника счетных импульсов различают режим таймера или счетчика. В режиме таймера ведется счет импульсов от внутреннего тактового генератора контроллера. В режиме счетчика подсчитываются импульсы, поступающие извне на одну из портовых ножек, которая в этом режиме служит счетным входом.

В дальнейшем будем называть тот или иной таймер-счетчик либо «таймером», либо «счетчиком» в зависимости от режима работы. Там, где не требуется подчеркивать режим работы, для краткости будем употреблять название «таймер». Следует понимать, что таймер 0 и счетчик 0 это на самом деле один и тот же узел контроллера.

Микроконтроллер PIC16F628 имеет три таймера: два 8-разрядных (таймеры 0 и 2) и один 16-разрядный (таймер 1). Таймеры 0 и 1 могут работать в режиме таймера или счетчика, таймер 2 только в режиме таймера.

Рассматривать работу таймеров будем на примере таймера 1.

9.1 Режим таймера

Таймер 1 доступен для чтения и записи через два регистра специальных функций TMR1H (0Fh) — старший байт и TMR1L (0Eh) — младший байт. Управление таймером производится через регистр специальных функций T1CON (10h).

В режиме таймера в каждом машинном цикле (при предварительном делителе 1:1) происходит увеличение содержимого пары регистров TMR1L, TMR1H на единицу. На аппаратном уровне реализован перенос из младшего байта (TMR1L) в старший (TMR1H), т. е. при переполнении TMR1L автоматически на единицу увеличится TMR1H. В случае переполнения пары регистров таймер сбрасывается в ноль. В связи с этим можно характеризовать таймеры *модулем счета*, т. е. максимальным числом импульсов, после прихода которого таймер-счетчик сбрасывается в ноль. Таймеры 0 и 1 — двоичные. Их модуль счета равен 2^n, где n — разрядность таймера. Таймер 2 — двоично-кодированный — модуль счета может принимать любое значение от 1 до 255. Переполнение таймера (его сброс в ноль) можно отследить через систему прерываний микроконтроллера (будет рассмотрено позднее).

Обычно таймер не может функционировать в режиме пониженного энергосбережения (режим SLEEP), т. к. при этом тактовый генератор остановлен, и поступление счетных импульсов прекращается. Таймер 1 оснащен автономным тактовым генератором, ко-

торый остается включенным в режиме SLEEP. При использовании генератора линии RB6, RB7 (выводы 12 и 13) выполняют альтернативные функции T1OSI, T1OSO. К ним подключается кварцевый резонатор частотой до 200 кГц.

9.2 Режим счетчика

В режиме счетчика ведется счет импульсов, поступающих на ножку RB6. В этом режиме она выполняет альтернативную функцию T1CLI. Линия RB6 обязательно должна быть настроена на ввод, т. е. бит 6 регистра TRISB должен содержать единицу. Переключение счетчика происходит по переднему фронту импульсов.

Различают режим синхронного и асинхронного счетчика. Основное отличие состоит в том, что в синхронном режиме опрос входа происходит один раз в каждом машинном цикле. В этом случае период входных импульсов должен превышать длительность машинного цикла не менее чем на 40 нс[5] [7]. Кроме того, в ждущем режиме, синхронизация прекращается, и счет становится невозможен.

В асинхронном режиме по данным технического описания [7] период счетных импульсов должен быть не менее 60 нс, т. е. допускается частота 16,7 кГц. В ждущем режиме счет не прекращается.

9.3 Предварительный и выходной делители

Таймер 1 имеет предварительный делитель (предделитель), который представляет собой аппаратный (асинхронный) счетчик-делитель. Счетные импульсы поступают на вход предделителя, а его выход соединен с входом таймера. Кроме коэффициента 1:1 (можно сказать, предделитель выключен) возможны три коэффициента деления 1:2, 1:4, 1:8. Коэффициент задается битами 4, 5 в соответствие с таблицей 9.1.

Таймер 2 кроме предделителя имеет еще и выходной делитель, через который сигнал, извещающий о переполнении, поступает в систему прерываний контроллера. Тогда прерывание по переполнению таймера происходит через каждые M переполнений, где M — модуль счета выходного делителя.

9.4 Регистр настройки таймера T1CON

Как уже отмечалось выше, управление работой таймера 1 производится через РСФ T1CON (адрес 10h). Приведем назначение его битов.

		T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
7	6	5	4	3	2	1	0

Рисунок 9.2 – Регистр управления таймером 1 T1CON

Бит 0 (TMR1ON) — бит включения и выключения таймера. Значение «1» включает таймер, «0» — выключает, счет запрещен.

Бит 1 (TMR1CS) — выбор источника счетных импульсов (выбор режима таймер/счетчик). «1» — режим счетчика, счетные импульсы принимаются с линии RB6. «0» — режим таймера, счет импульсов внутренней тактовой частоты.

Бит 2 ($\overline{\text{T1SYNC}}$) — в режиме счетчика выбирает синхронный («1») или асинхронный («0») счет. В режиме таймера значения не имеет.

Бит 3 (T1OSCEN) — значение «1» включает автономный тактовый генератор таймера 1.

Биты 4–5 (T1CKPS0, 1) — выбор предварительного делителя (предделителя) в соответствие с таблицей 9.1.

[5] При условии, предделитель 1:1

Таблица 9.1 – Выбор предделителя через регистр T1CON.

T1CKPS1	T1CKPS0	Значение предделителя
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

Биты 6–7 — не используются.

В заключение приведем сравнительные характеристики всех трех таймеров (см. таблицу 9.2).

Таблица 9.2 – Характеристики таймеров микроконтроллера PIC16F628

	Таймер 0	Таймер 1	Таймер 2
Разрядность	8	16	8
Режимы счетчика	Синхронный	Синхронный, асинхронный	Отсутствует
Предварительный делитель	Выключен (подключен к сторожевому таймеру), 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128, 1:256	1:1, 1:2, 1:4, 1:8	1:1, 1:4, 1:16
Выходной делитель	Отсутствует	Отсутствует	1:1, 1:2, 1:3…1:16
Особенности	Инкремент по фронту или по срезу внешнего сигнала. Предделитель общий со сторожевым таймером.	Автономный кварцевый генератор синхроимпульсов частотой до 200 кГц.	Только режим таймера. Двоично-кодированный (модуль счета произвольный от 1 до 255).
Регистры	TMR0	TMR1H, TMR1L	TMR2
Регистры управления	OPTION	T1CON	T2CON

10 Система прерываний

Во время выполнения программы внутри МК или в связанной с ним внешней среде могут возникать события, требующие немедленной реакции на них со стороны контроллера. Реакция состоит в том, что МК прерывает обработку текущей программы и переходит к выполнению другой программы, специально предназначенной для данного события. По завершении этой программы контроллер возвращается к выполнению прерванной программы.

Рассматриваемый процесс называется *прерыванием* программы. Принципиально важным является то, что моменты возникновения событий, требующих немедленной реакции заранее неизвестны, и поэтому не могут быть учтены при программировании.

Каждое событие, требующее прерывания, сопровождается сигналом, оповещающим контроллер. Эти сигналы называются *запросами прерывания*, а события их порождающие — *источниками прерываний*. Программу, затребованную запросом прерывания, будем называть *процедурой обработки прерывания*, а адрес, по которому она размещается в памяти программ — *вектором прерывания*.

Система прерываний представляет собой совокупность аппаратурных средств, необходимых для реализации прерываний программы.

Система прерываний микроконтроллера PIC16F628 способна воспринимать запросы от десяти источников:

1. Переполнение таймера 0.
2. Перепад уровня на входе RB0.
3. Изменение уровня на линиях RB4–RB7.

4. Завершение записи во встроенное ЭППЗУ данных.
5. Переключение компараторов
6. Завершение приема данных через УСАПП.
7. Завершение передачи данных через УСАПП.
8. Прерывание от блока захвата-сравнения.
9. Возврат таймера 2 в исходное состояние.
10. Переполнение таймера 1.

В МК семейства PIC16 все источники прерываний связаны с одним и тем же вектором — 04h. Это означает, что появление любого из перечисленных запросов приведет к передаче управления команде, расположенной по адресу 04h.

С каждым источником прерываний связаны бит разрешения прерывания и флаг прерывания. Все они размещены в трех регистрах специальных функций: INTCON (0Bh, 8Bh, 10Bh, 18Bh), PIE1 (0Ch), PIR1 (8Ch). PIE1 называют регистром управления периферийными прерываниями, а PIR — регистром флагов периферийных прерываний. К периферийным устройствам в PIC16 отнесены встроенное ЭППЗУ данных, компараторы, УСАПП, блок захвата-сравнения, таймеры 1 и 2.

При помощи битов разрешения можно разрешить или запретить каждое прерывание индивидуально. Единица в соответствующем бите означает, что прерывание разрешено, ноль — запрещено. Кроме того, существует общий бит разрешения прерываний и бит разрешения прерываний от периферийных устройств.

Флаг прерывания устанавливается аппаратурой контроллера в единицу при возникновении прерывающего события, причем независимо от того разрешено прерывание или запрещено. Это позволяет отслеживать события, не прибегая к механизму прерываний. Назначения битов трех регистров управления прерыван6ями сведены в таблицу 10.1.

Таблица 10.1

Источник прерывания	Бит разрешения		Флаг прерывания	
	РСФ и номер бита	Обозначение бита	РСФ и номер бита	Обозначение бита
Переполнение таймера 0	INTCON [5]	T0IE	INTCON [2]	T0IF
Изменение уровня на RB0	INTCON [4]	INTE	INTCON [1]	INTF
Изменение уровня на RB4–RB7	INTCON [3]	RBIE	INTCON [0]	RBIF
Завершение записи в ЭППЗУ	PIE1 [7]	EEIE	PIR1 [7]	EEIF
Переключение компараторов	PIE1 [6]	CMIE	PIR1 [6]	CMIF
Завершение приема данных через УСАПП	PIE1 [5]	RCIE	PIR1 [5]	RCIF
Завершение передачи данных через УСАПП.	PIE1 [4]	TXIE	PIR1 [4]	TXIF
Прерывание от блока захвата-сравнения	PIE1 [2]	CCP1IE	PIR1 [2]	CCP1IF
Прерывание от таймера 2	PIE1 [1]	TMR2IE	PIR1 [1]	TMR2IF
Переполнение таймера 1	PIE1 [0]	TMR1IE	PIR1 [0]	TMR1IF
Общий бит разрешения прерываний	INTCON [7]	GIE		
Бит разрешения прерываний от периферийных устройств	INTCON [6]	PEIE		

Рассмотрим подробнее механизм обработки прерываний. Как только аппаратурой контроллера распознается прерывающее событие, устанавливается соответствующий флаг прерывания.

Для запуска механизма обработки прерываний необходимо выполнение условий:

1. данное прерывание разрешено;
2. установлен бит GIE

3. установлен бит PRIE (только для периферийных прерываний)

Если все три условия выполнены, то аппаратно выполняются следующие действия:

1. Бит GIE сбрасывается в ноль. Таким образом, на время выполнения процедуры обработки прерывания все прерывания автоматически запрещаются.
2. В стек заносится содержимое счетчика команд. В этот момент он указывает на следующую команду — это адрес возврата из процедуры обработки прерывания.
3. Управление передается на адрес 04h памяти программ — вектор прерывания.

Процедура обработки прерываний заканчивается командой RETFIE. При этом выполняются следующие действия:

1. Бит GIE устанавливается в единицу. Все прерывания вновь разрешаются.
2. Из стека извлекается адрес возврата и записывается в счетчик команд. Выполнение программы продолжается с того места, на каком было прервано.

При этом следует отметить ряд важных особенностей, которые необходимо учитывать при программировании.

1. Флаг прерывания аппаратно устанавливается в единицу при возникновении прерывающего события, но не сбрасывается в ноль, по завершении обработки прерывания. Сброс необходимо выполнять программно внутри процедуры обработки прерываний, например командой BCF. Иначе после выполнения команды RETFIE вместо ожидаемого возврата снова начнет выполняться процедура обработки прерываний. Команду сброса флага прерываний рекомендуется помещать непосредственно перед командой RETFIE. В некоторых случаях сброс флагов в начале процедуры может привести к тому же зацикливанию.

2. В большинстве случаев в самом начале процедуры обработки прерываний необходимо сохранять состояние контроллера, а именно содержимое регистра состояния STATUS и аккумулятора W. Дело в том, что в процессе выполнения процедуры обработки прерываний эти регистры скорее всего будут изменены, а следовательно их прежние значения, используемые в основной программе, будут потеряны. Пренебрежение этим правилом приводит к возникновению у начинающих программистов большого числа сбоев программы, проявление которых носит случайный характер и трудно диагностируется.

Для сохранения состояния необходимо выделить два регистра общего назначения. Рекомендуется использование следующего кода:

```
1.          movwf     w_temp
2.          swap      status, w
3.          movwf     stat_temp
```

Здесь W_TEMP и STAT_TEMP — заранее объявленные регистры общего назначения. Вполне очевидно, что сохранение аккумулятора выполняется командой MOVWF. В то время как для сохранения STATUS пользоваться командами пересылки нельзя, потому, что они сами влияют на флаги состояния, т. е. на содержимое регистра STATUS. Поэтому возможно применение лишь команд SWAPF и MOVWF, не изменяющих флагов.

Восстановление состояния выполняется в самом конце процедуры в обратной последовательности.

```
1.          swap      status, w
2.          movwf     stat_temp
3.          movf      w_temp, w
```

Еще раз подчеркнем, что команды сохранения и восстановления состояния рекомендуется помещать непосредственно в начале и в конце процедуры обработки прерываний.

11 Применение таймера для формирования временных интервалов

Задача генерирования прямоугольных импульсов сводится к формированию временных интервалов и изменению логических уровней на выводах МК в конце каждого временного интервала. Наиболее гибкое решение получается с использованием прерыва-

ний по переполнению таймера. Тогда команды, изменяющие логические уровни на выводах МК, помещаются внутри процедуры обработки прерываний. В разных случаях может быть удобно формирование интервала, равного периоду, а чаще половине периода. Тогда частота повторения процедуры вдвое превышает генерируемую частоту.

Длительность интервала, по истечении которого будет выработано прерывание, определяется тактовой частотой контроллера и содержимым таймера в начальный момент времени. Чем больше начальное значение таймера, тем быстрее произойдет его переполнение и тем меньше длительность интервала. Если в начальный момент времени таймер содержал нулевое значение, то прерывание произойдет через $65536 \cdot 0{,}2 \cdot 10^{-6}$с ≈ 13,1 мс, где $0{,}2 \cdot 10^{-6}$с — длительность машинного цикла. Предполагается, что тактовая частота равна 20 МГц и предварительный делитель таймера отсутствует. Изменяя начальное значение таймера можно сформировать временной интервал любой длительности с абсолютной погрешностью теоретически не более ±100 нс.

Рассмотрим пример использования прерываний. Программа, приведенная ниже, обеспечивает генерирование на всех линиях порта B прямоугольных импульсов частотой примерно 1 кГц. Для формирования временных интервалов используется прерывание от таймера 1.

```
                                ;Описание РСФ
status     equ      03          ;Регистр состояния
porta      equ      05          ;Порт A
portb      equ      06          ;Порт B
intcon     equ      0B          ;Регистр управления прерываниями
pir1       equ      0C          ;Регистр флагов прерываний
pie1       equ      8C          ;Регистр управления периферийными прерываниями
tmr1l      equ      0E          ;Младший байт таймера 1
tmr1h      equ      0F          ;Старший байт таймера 1
t1con      equ      10          ;Регистр управления таймером 1

                                ;Описание РОН
w_temp    equ       20          ;Регистры для временного...
stat_temp equ       21          ;...хранения состояния

 1.       goto      start

 2.       org       4               ;Процедуры обработки прерываний
 3.       movwf     w_temp          ;Сохранение состояния
 4.       swapf     status, w
 5.       movwf     stat_temp

 6.       comf      portb, f        ;Инверсия порта B

 7.       bcf       t1con, 0        ;Остановить таймер 1
 8.       movlw     0F6
 9.       movwf     tmr1h
10.       movlw     3C
11.       movwf     tmr1l
12.       bsf       t1con, 0        ;Остановить таймер 1

13.       bcf       pir1, 0         ;Сброс флага прерывания

14.       swapf     stat_temp, w    ;Восстановление состояния
15.       movwf     status
16.       movf      w_temp, w

17.       retfie

   start                           ;Основная программа
18.       movlw     b'00111111'     ;Настройка на ввод...
19.       tris      porta           ;...порта A
20.       movlw     b'00000000'     ;Настройка на вывод...
```

```
21.        tris      portb          ;...порта B
22.        clrf      portb          ;Сброс в ноль линий порта B

23.        movlw     b'00000001'    ;Настройка и запуск таймера 1
24.        movwf     t1con
                                    ;Настройка прерываний
25.        movlw     b'11000000'    ;Разрешение периферийных прерываний
26.        movwf     intcon

27.        bsf       status, 5      ;Выбор страницы памяти 1
28.        movlw     b'00000001'    ;Разрешение прерывания от таймера 1
29.        movwf     pie1
30.        bcf       status, 5      ;Возврат на страницу памяти 0

   loop
31.        goto      loop           ;Вечный цикл

           end
```

Первая команда необходима для того, чтобы «обойти» процедуру обработки прерываний и приступить к выполнению основной программы. На начало основной программы указывает метка START (команда №18). Команды №18–22 подготавливают порты к работе. Все линии порта A настраиваются на ввод путем записи единиц в регистр TRISA командой TRIS; порт B настраивается на вывод.

Команды №23, 24 настраивают таймер 1. Здесь используется двоичное представление числа. Такая запись упрощает задание каждого бита РСФ. Настройка производится в соответствие с рисунком 9.2. Таймер-счетчик запускается в режиме таймера, предварительный делитель выключается.

Настройка прерываний (команды №25–30) выполняется в два этапа. Во-первых, устанавливаются общий бит разрешения прерываний и бит разрешения периферийных прерываний в регистре INTCON (см. таблицу 10.1). Во-вторых, разрешается прерывание от таймера 1 через регистр PIE1. Следует учитывать, что PIE1 расположен на первой странице памяти, поэтому прежде чем обратиться к нему необходимо переключить страницу (команда №27), а после обращения выбрать исходную нулевую страницу (команда №30).

Далее контроллер переходит к выполнению вечного цикла. Команда №31 выполняет переход на саму себя. Цикл будет периодически приостанавливаться прерыванием от таймера 1.

Рассмотрим процедуру обработки прерываний (команды №3–17). На второй строке расположена директива ассемблера ORG, задающая адрес следующей команды. Процедура начинается с сохранения состояния (команды №3–5). Эти команды включены в программу лишь для примера. Здесь сохранять состояние не обязательно, потому что основная часть программы представляет собой вечный цикл и не использует ни аккумулятора, ни регистра STATUS.

Изменение уровней на выводах порта B выполняется командой COMF (логическая инверсия). Таким образом, каждый раз при переполнении таймера уровни порта B меняются на противоположные. Очевидно, что интервал времени от одного переполнения до другого соответствует половине периода генерируемых импульсов.

Команды №7–15 служат для присваивания таймеру начального значения, определяющего частоту выходных импульсов. Рассчитаем начальное значение таймера.

Желаемая частота f=1000 Гц, период $T=10^{-3}$ с. Определим количество N машинных циклов контроллера, соответствующих половине периода генерируемого сигнала.

$$N = \frac{T/2}{0{,}2 \cdot 10^{-6}} = 2500,$$

где $0{,}2 \cdot 10^{-6}$ — длительность машинного цикла микроконтроллера при тактовой частоте 20 МГц. Начальное значение таймера определяется как разность модуля счета таймера M

и числа N. Обозначим начальное значение таймера буквой C, тогда $C=M–N=2^{16}–$ $–2500=63036$. Это число необходимо разместить в двух регистрах TMR1H и TMR1L, поэтому переходим к шестнадцатеричной записи числа: $63036_{10}=F63C_{16}$ и выделяем старший и младший байты: F6 и 3C.

Такой расчет является неточным, потому что не учитывает время, затрачиваемое на выполнение команд №3–12. Поэтому при $N=F63C_{16}$ период повтора процедуры составляет не 500 мкс, а 502 мкс. Проводить точный расчет нецелесообразно. Для этого пришлось бы вручную пересчитать все команды с начала процедуры обработки прерываний до запуска таймера, учитывая при этом время их выполнения (1 или 2 машинных цикла). На практике начальное значение таймера рассчитывается приближенно, период повтора определяется экспериментально при помощи отладочных средств, а затем корректируется (подгоняется).

Присваивание таймеру начального значения выполняется в два этапа. Сначала старший байт (команды №8, 9), затем младший (команды №10, 11), хотя можно и наоборот. В этой связи следует отметить важную особенность. Запись в таймер ведется побайтно, поэтому в какой-то момент таймер содержит некоторое промежуточное значение — старший байт обновлен, а младший еще нет. Если в этот момент очередной инкремент таймера вызовет перенос из младшего байта в старший, то результирующее значение таймера будет отличаться от желаемого. Проиллюстрируем такую ситуацию примером. Пусть требуется записать в таймер шестнадцатеричное число 1020_{16}. В таблице приведен фрагмент программы и значения таймера в каждом машинном цикле. Для удобства машинные циклы пронумерованы.

Таблица 11.1 –

№ Маш. цикла	Команда	Содержимое таймера		Примечание
		TMR1H	TMR1L	
1	`movlw 10`	00	FE	Таймер содержит 00FE
2	`movwf tmr1h`	10	FF	Старшему байту присвоено значение 10
3	`movlw 20`	11	00	Младший байт переполняется, вызывая инкремент старшего байта
4	`movwf tmr1l`	11	20	Таймер содержит 1120 вместо 1020

Чтобы предотвратить такую ошибку нужно останавливать таймер, прежде чем обращаться к нему для чтения или записи, а затем снова запускать. В рассматриваемом примере программы для этого служат команды №7, 12. Существует и другой способ — очищать командой CLRF младший байт таймера перед записью старшего байта. Но тогда необходимо соблюдать очередность записи: сначала старший байт, затем младший.

Команда №13 очищает флаг прерывания, предотвращая его повторную обработку. Команды №14–16 восстанавливают состояние контроллера. Выход из процедуры обработки прерываний осуществляется командой RETFIE.

Сокращения, используемые в тексте

АЛУ — арифметико-логическое устройство.
МК — микроконтроллер.
МП — микропроцессор.
ПД — память данных.
ПЗУ — постоянное запоминающее устройство.
ПП — память программ.
РОН — регистр общего назначения.
РСФ — регистр специальных функций.
УСАПП — универсальный синхронно-асинхронный приемопередатчик.
ЦПОС — цифровой процессор обработки сигналов.
ЭППЗУ — электрически перепрограммируемое постоянное запоминающее устройство.

Список литературы

1. Яценков В. С., Микроконтроллеры Microchiop®. Практическое руководство. — М.: Горячая линия-Телеком, 2002. — 296 с.
2. Малафеев С. И., Малафеева А. А., Мамай В. С. и др., Микроконтроллеры PIC16C8X. Архитектура, программирование и применение: Учеб. пособие. — Владимир: ВлГУ, 1999. — 180 с.
3. Каган Б. М., Электронно-вычислительные машины и системы: Учеб. пособие для вузов — 3-е изд. перераб. и доп. — М.: Энергоатомиздат, 1991. — 592 с.
4. Микропроцессоры, в 3-х кн., под ред. Л. Н. Преснухина, кн. 1, Архитектура и проектирование микро-ЭВМ: Учеб. для вузов. — М.: Высш. шк., 1986. — 495 с.
5. Сташин В. В. и др., Проектирование цифровых устройств на однокристальных микроконтроллерах. — М.: Энергоатомиздат, 1990. — 224 с.
6. Опадчий Ю. Ф., Глудкин О. П., Гуров А. И., Аналоговая и цифровая электроника (Полный курс): Учебник для вузов, под. ред. Глудкина О. П. — М.: Горячая Линия – Телеком, 2000. — 768 с.
7. Microchip, PIC16F628 Data Sheet (DS40300B)
8. Microchip, PIC16F87x Data Sheet (DS30292C)
9. Microchip, PIC18Fxx2 Data Sheet (DS39564A)
10. Microchip Technology Inc., Product Line Card, January–March 2003.