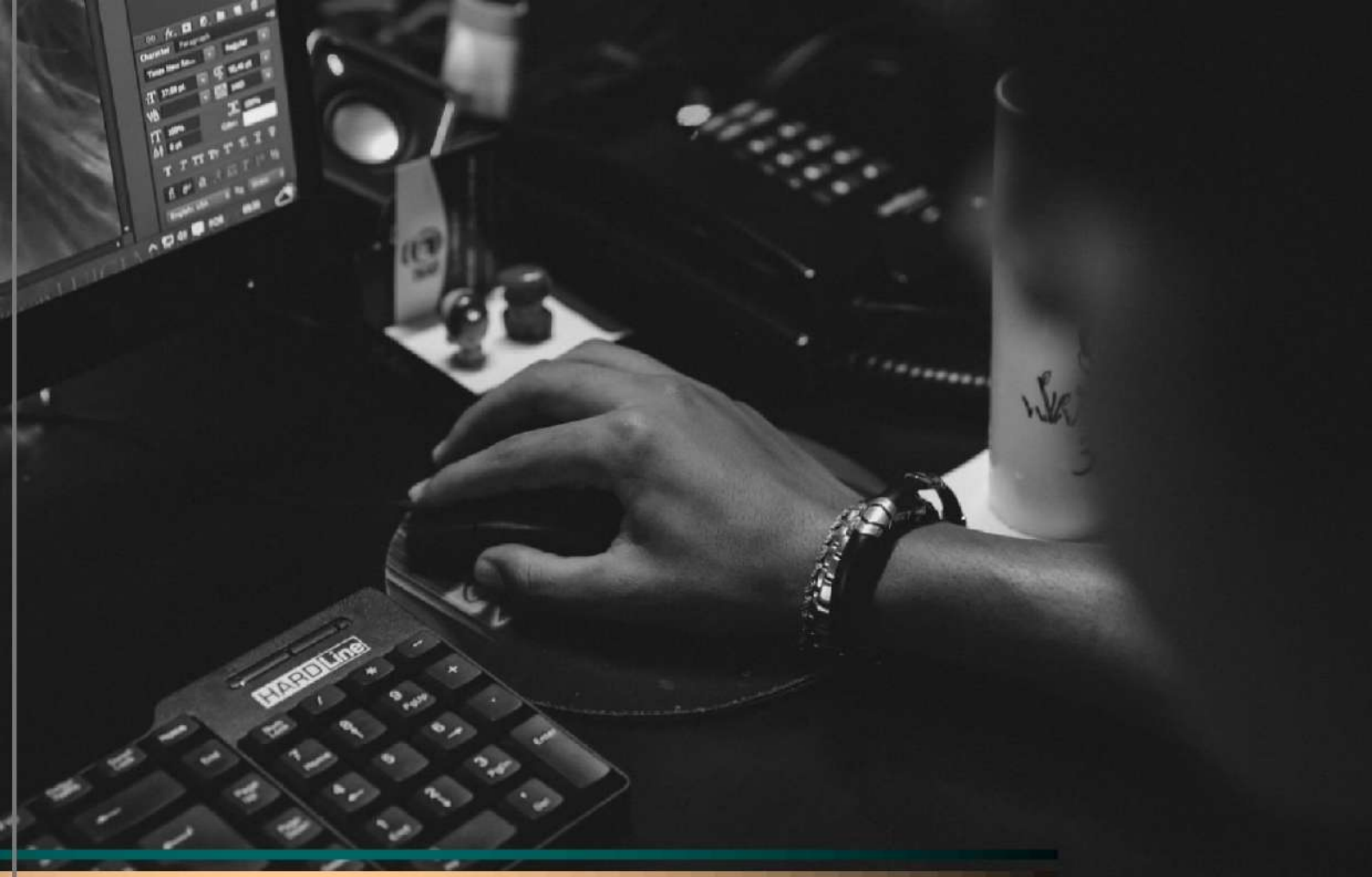

Д.П.Харченко, А.П.Волошин
С.А.Николаенко, Д.С.Цокур

СХЕМОТЕХНИКА

ВНУТРЕННЕЕ УСТРОЙСТВО И ПРОГРАММИРОВАНИЕ PIC-МИКРОКОНТРОЛЛЕРОВ

Учебное пособие

Д.П.Харченко, А.П.Волошин
С.А.Николаенко, Д.С.Цокур

СХЕМОТЕХНИКА

ВНУТРЕННЕЕ УСТРОЙСТВО И ПРОГРАММИРОВАНИЕ PIC-МИКРОКОНТРОЛЛЕРОВ

Учебное пособие

Краснодар
2014

УДК 621.382 (075.8)
ББК 32.852

Рецензент:

О.В. Григораш – профессор кафедры электротехники, теплотехники и ВИЭ Кубанского государственного аграрного университета, д-р тех. наук, профессор

С92 Схемотехника: Внутреннее устройство и программирование PIC-микроконтроллеров: учебное пособие / Д. П. Харченко, С. А. Николаенко, А. П. Волошин, Д. С. Цокур. – Краснодар: КубГАУ, 2014. – 98 с.

Учебное пособие содержит сведения о внутренней архитектуре, системе команд и программировании современных PIC-микроконтроллеров, получивших распространение во встроенных системах контроля и управления. Рассмотренные микроконтроллеры являются практически полностью готовыми вычислительными устройствами, не требующими для своей работы дополнительного оборудования и позволяющие реализовывать достаточно сложные электронные устройства, в которых большая часть функционала реализуется программно.

Учебное пособие предназначено для бакалавров энергетического факультета и факультета заочного обучения по направлению подготовки 110800 «Агроинженерия».

Утверждено и рекомендовано к использованию в учебном процессе методической комиссией энергетического факультета Кубанского госагроуниверситета, протокол №8 от 27.05.14 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Число представленных на рынке разнообразных микроконтроллеров непрерывно растет. Почти еженедельно появляется новое изделие. Наряду с небольшими МК-чипами на рынок выпускается все больше многоядерных устройств. Если в начале появления универсальных устройств основное внимание разработчиков уделялось повышению быстродействия и снижению стоимости МК, то сегодня все больший интерес вызывают контроллеры с расширенными функциональными возможностями, рассчитанные на конкретное применение. Наращиваются средства взаимодействия микроконтроллеров с другими устройствами, например с дешевыми датчиками и Ethernet, а также средства поддержки протоколов беспроводной связи. Непрерывное уменьшение размеров элементов микроконтроллеров, наращивание их интеллекта и повышение быстродействия приводят к расширению их функциональных возможностей и снижению стоимости.

Микроконтроллер - микросхема, предназначенная для управления электронными устройствами, которая сочетает в себе функции процессора и периферийных устройств, а также содержит ОЗУ и ПЗУ. По сути, это однокристальный компьютер, способный выполнять определенные задачи. Большая часть выпускаемых в современном мире процессоров — микроконтроллеры. Первый микроконтроллер появился на свет еще в 1976 году, это была микросхема фирмы Intel, получившая имя 8048. В дальнейшем, Intel продолжала развивать эту категорию устройств, положившую начало целому семейству микроконтроллеров, которые господствовали на рынке вплоть до недавнего времени. Первые значительные перемены произошли с появлением PIC-контроллеров фирмы Microchip. Эти чипы предлагались по рекордно низким ценам, что позволило им в короткий срок захватить значительную часть рынка микроконтроллеров.

Микроконтроллеры семейства PIC построены по гарвардской архитектуре, что подразумевает разделение памяти и шин данных и команд, что позволяет за один такт микроконтроллера выполнять обращение к памяти данных и к памяти команд. В микроконтроллерах PIC реализована двухступенчатая конвейерная обработка команд, что обеспечивает одновременное исполнение текущей команды и вы-

борку из памяти данных следующей. Все команды микроконтроллера, кроме команд безусловного и условного переходов, выполняются за один конвейерный такт. Операции безусловного и условного перехода, включая команды вызова подпрограмм и возврат из них, исполняются за два конвейерных такта.

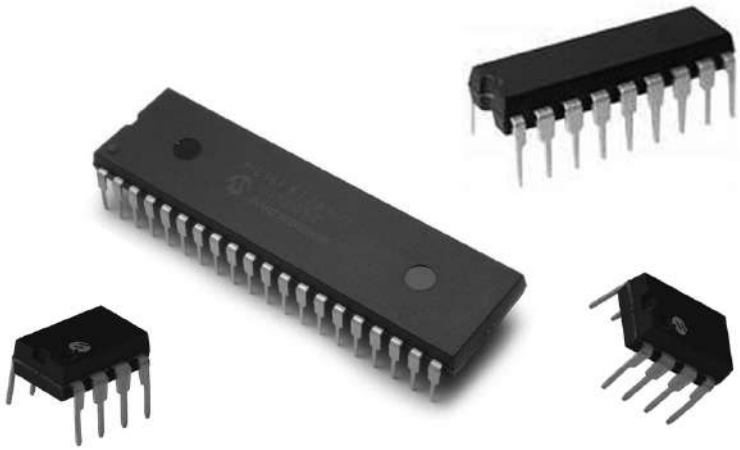

Рисунок 1 - 8-разрядные микроконтроллеры семейства PIC в DIP-корпусах с различным числом выводов

Наибольшее распространение микроконтроллеры получили во встроенных системах контроля и управления. Одной из главных причин популярности микроконтроллеров служит то, что они являются практически полностью готовыми вычислительными устройствами, не требующими для своей работы дополнительного оборудования. Кроме того, возможность программировать работу микроконтроллера позволяет реализовывать достаточно сложные электронные устройства, в которых большая часть функционала реализуется программно.

В настоящее время на рынке микроконтроллеров активно работают более 30 разработчиков и изготовителей. Производители предлагают широкий ассортимент микроконтроллеров, отличающихся как техническими характеристиками, так и перечнем встроенных периферийных устройств, благодаря чему разработчики имеют возможность подобрать микроконтроллер, который наиболее подходит для решения конкретной задачи.

Важной характеристикой, влияющей как на практичность, так и на цену устройства, является способ программирования. Перепрограммируемые микроконтроллеры, являются самыми дорогими, но вместе с тем, и наиболее практичными устройствами для мелкосерийного и экспериментального производства. Однократно-программируемые микроконтроллеры дешевле перепрограммируемых однако, про-

граммирование возможно только один раз. Масочно-программируемые микроконтроллеры – самый дешевый способ изготовления, но программирование осуществляется промышленным способом на заводе изготовителе, что делает возможным применение подобных микроконтроллеров только в крупносерийном производстве, при условии, что программа изменяться не будет. Использование в современном микроконтроллере мощного вычислительного устройства с широкими возможностями, построенного на одной микросхеме вместо целого набора, значительно снижает размеры, энергопотребление и стоимость устройств, построенных на его базе.

С момента появления первого микропроцессора в 1970-х годах бурно развивается область цифровой управляющей электроники, относящаяся к встраиваемым микропроцессорным системам управления реального времени. Речь идет не только о прямом управлении ключами силовых преобразователей, но и о прямом сопряжении с широкой номенклатурой датчиков обратных связей (положения, скорости, ускорения), а также с элементами дискретной автоматики (релейно-контакторной аппаратурой, дискретными датчиками и дискретными исполнительными устройствами). Область управления двигателями и силовыми преобразователями стала ярким примером быстрой адаптации процессорной техники к задачам предметной области.

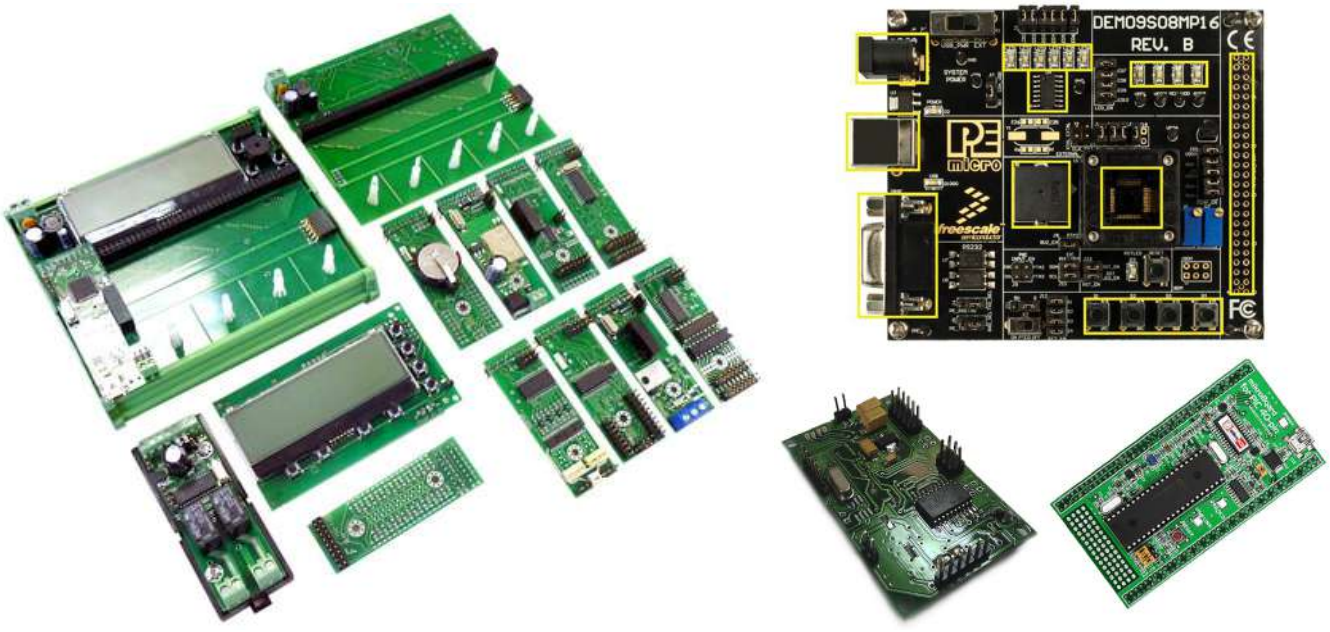

Рисунок 2 - Внешний вид модулей микроконтроллерных систем управления

Функции прямого цифрового управления в современных приводах реализуются за счет использования специализированных периферийных устройств, интегрированных непосредственно на кристалл микроконтроллера, и не требующих до-

полнительных развитых средств сопряжения, а также за счет высокопроизводительной архитектуры и системы команд центрального процессора. Переход к цифровым системам управления приводами на базе специализированных микроконтроллеров позволил обеспечить новый, недостижимый в аналоговых системах, уровень показателей качества:

на порядок меньшие габариты и вес управляющей электроники;

- резкое повышение надежности (фактическое время наработки на отказ достигает 100000 час и выше) и срока службы привода (до 10 лет и более);

- быструю и качественную интеграцию привода в систему комплексной автоматизации производства с помощью унифицированных интерфейсов сопряжения с системами управления более высокого уровня и соответствующих средств программной поддержки (RS-232, RS-485, CAN);

- местное и дистанционное управление;

- интерактивный дружественный интерфейс с человеком-оператором на языке страны использования привода: отображение на встроенном дисплее информации о текущем состоянии привода и значениях наблюдаемых переменных; ввод команд оперативного управления со встроенной клавиатуры; настройка параметров привода и системы управления в процессе пуско-наладочных работ с сохранением значений в энергонезависимой памяти; интерактивная справочная система и система подсказок стратегии управления в реальном времени;

- встроенный и удаленный (по сети) мониторинг состояния привода и раннее предупреждение аварийных ситуаций в технологическом оборудовании, возникающих вследствие срабатывания защит или идентификации отказов в приводе;

- конфигурирование структуры системы управления самим пользователем в процессе запуска привода в эксплуатацию для адаптации к конкретной технологии или специфике применения привода;

- встроенное управление средствами привода сопутствующей дискретной автоматикой без использования дополнительных промышленных программируемых контроллеров и управляющих ЭВМ;

- распределенное мультимикропроцессорное управление многоосевыми приводами роботов, манипуляторов, кабельных линий и т.п. с использованием локаль-

ных промышленных сетей, например, CAN, с широкими возможностями синхронизации, вплоть до систем электрического вала, распределенного позиционного и контурного управления;

- унификацию встроенных средств управления приводами (контроллеров, модулей ввода-вывода, пультов оперативного управления) независимо от типа исполнительного двигателя, структуры силового канала, типов используемых датчиков обратаных связей;

- простую систему наращивания мощности комплектного электропривода за счет использования секционируемых исполнительных двигателей (например, вентильно-индукторных), каждая секция которых управляется от отдельного типового преобразователя с объединением систем управления всеми преобразователями в локальную промышленную сеть;

- возможность использования самых современных структур и алгоритмов управления приводами, которые трудно, а порой и невозможно реализовать на аналоговой элементной базе: векторного управления двигателями переменного тока; прямого управления моментом; прямого частотно-токового управления; управления с элементами фаззи-логики; прямой программной реализации по графам переходов дискретных управляющих автоматов любой сложности и т.п.

В данном пособии рассмотрены основные особенности программирования 8-разрядных PIC микроконтроллеров производства фирмы Microchip. На сегодняшний день данная фирма имеет широчайшую линейку по выпуску 8, 16, и 32-разрядных приборов, отличающихся друг от друга техническими характеристиками, функциональными возможностями и ценой. Выбор микроконтроллера зависит от конкретной инженерной задачи, однако принципы программирования практически всех контроллеров фирмы Microchip остаются схожими, что значительно облегчает переход между контроллерами различных семейств.

1 ОБЩИЕ СВЕДЕНИЯ О PIC-КОНТРОЛЛЕРАХ

Структура микроконтроллеров. Каждая часть микроконтроллера может быть отнесена к одной из трех групп:

1. Ядро микроконтроллера;
2. Периферийные модули;
3. Специальные особенности микроконтроллеров.

Ядро микроконтроллера. Ядро относится к основным особенностям, оно заставляет микроконтроллер работать. В состав этой группы входит:

- Тактовый генератор
- Логика сброса
- Центральный процессор (CPU)
- Арифметико-логическое устройство
- Организация памяти
- Прерывания
- Система команд

Периферийные модули. Периферийные модули - особенности, которые добавляются независимо от центрального процессора. Периферийные модули позволяют организовать интерфейс связи с внешней схемой (например, универсальные порты ввода/вывода, драйверы ЖКИ, входы АЦП, выходы ШИМ) и выполнять отсчет временных интервалов (таймеры). Периферийные модули:

- Универсальные порты ввода/вывода
- Таймер TMR
- Захват/Сравнение/ШИМ (CCP)
- Синхронный последовательный порт (SSP)
- Ведущий синхронный последовательный порт (MSSP)
- USART
- Источник опорного напряжения
- Компараторы
- 10-разрядное АЦП
- Драйвер ЖКИ

- Ведомый параллельный порт (PSP)
- Прочие модули.

Специальные особенности микроконтроллеров. Специальные особенности - уникальные особенности микроконтроллера, позволяющие придать одно или более следующих свойств конечному изделию:

- Уменьшить стоимость устройства;
- Увеличить надежность системы;
- Предоставить дополнительную гибкость разработчикам при проектировании устройства.

Специальные особенности микроконтроллеров PIC:

- Биты конфигурации
- Интегрированная схема сброса по вклю-
- Схема сброса по снижению напряжения
- Сторожевой таймер
- Режим энергосбережения (SLEEP)
- Интегрированный тактовый RC генератор
- Внутрисхемное программирование

После определения функциональных требований к микроконтроллеру необходимо выбрать следующие параметры:

- Технология памяти;
- Рабочий диапазон напряжения питания;
- Рабочий температурный диапазон;
- Тактовая частота;
- Тип корпуса.

Технология памяти. Технология, по которой выполнена память, не влияет на логические операции микроконтроллеров. Из-за различной последовательности изготовления кристалла некоторые электрические параметры могут отличаться для микроконтроллеров с разной технологией памяти. Например, электрический параметр VIL (входное напряжение низкого уровня) может отличаться в типовом микроконтроллере с EPROM памятью и типовым микроконтроллером с ROM памятью.

Каждый микроконтроллер имеет ряд диапазонов тактовой частоты и доступных упаковочных параметров. При выборе функциональных возможностей микроконтроллера технология памяти и диапазон напряжения питания не имеют значения. Microchip предлагает три типа памяти программ. Код типа памяти программ обозначен символами в наименовании микроконтроллера после цифр семейства микроконтроллеров.

1 - C, как в PIC16CXXX - EPROM память программ;

2 - CR, как в PIC16CRXXX - ROM память программ;

3- F, как в PIC16FXXX - FLASH память программ.

Рабочий диапазон напряжения питания. Все микроконтроллеры среднего семейства PICmicro MCU работают в стандартном диапазоне напряжения питания. Некоторые микроконтроллеры могут работать в расширенном диапазоне напряжений питания (с уменьшением тактовой частоты). В таблице 1-1 показаны все возможные типы памяти и рабочий диапазон напряжения питания для PIC16CXXX.

Таблица 1.1 - Тип памяти программ и рабочий диапазон напряжения питания

Тип памяти	**Диапазон напряжения питания**	
	Стандартный	**Расширенный**
EPROM	PIC16**C**XXX	PIC16**LC**XXX
ROM	PIC16**CR**XXX	PIC16**LCR**XXX
FLASH	PIC16**F**XXX	PIC16**LF**XXX

В таблице 1.2 можно увидеть, что если не известны точные параметры устройства, то минимальное напряжение питания для расширенного диапазона несколько ограничено. Для выполнения спецификации необходимо точно определить параметры устройства.

Таблица 1.2 - Диапазон напряжения питания для каждого типа микроконтроллера

Диапазон напряжения питания		**EPROM**		**ROM**		**FLASH**	
Стандартный		**C**	4.5 - 6.0 В	**CR**	4.5 - 6.0 В	**L**	4.5 - 6.0 В
Расширенный	Предварительные параметры	**LC**	3.0 - 6.0 В	**LCR**	3.0 - 6.0 В	**LF**	3.0 - 6.0 В
	Окончательные параметры	**LC**	2.5 - 6.0 В	**LCR**	2.5 - 6.0 В	**LF**	2.0 - 6.0 В

Тип корпуса. В зависимости от стадии проектирования устройства может использоваться микроконтроллер в одном из следующих корпусов:

1. Корпус с окном для стирания памяти. Обычно используется керамический корпус. Микроконтроллеры в таком корпусе как правило используются на этапе проектирования, т.к. память программ может быть стерта и повторно запрограммирована много раз.

2. Недорогой пластмассовый корпус. Этот тип корпуса применяется в готовом устройстве, с целью минимизировать его стоимость.

3. DIE - проверенный, не упакованный кристалл. DIE применяется для недорогих приложений, в которых необходимо минимизировать размер печатной платы.

В таблице 1.3 представлена сводная информация.

Таблица 1.3 - Типовое применение микроконтроллеров в различных корпусах

Тип корпуса	**Типовое применение**
С окном для стирания памяти	Разработка проекта
Пластмассовый	Выпуск продукции
DIE	Специальные приложения, требующие минимальные размеры печатной платы

Микроконтроллеры с ультрафиолетовым стиранием памяти.

Микроконтроллеры с УФ стираемой EPROM памятью программ оптимальны для подготовки опытного образца устройства и экспериментальных программ. Интервал времени, необходимый для стирания памяти, зависит от следующих параметров: длина волны излучаемого УФ света, мощность источника, расстояние от источника до микроконтроллера, технология изготовления кристалла (размер ячейки памяти).

Однократно программируемые микроконтроллеры OTP.

OTP микроконтроллеры выпускаются в пластмассовых корпусах с однократно программируемой EPROM памятью программ. Вместе с памятью программ должны быть запрограммированы биты конфигурации. Эти микроконтроллеры предназначены для изделий, выпускаемых небольшими партиями с возможным изменением текста программы.

FLASH микроконтроллеры.

FLASH микроконтроллеры позволяют выполнять электрическое перепрограммирование памяти. Устройство может быть разработано таким образом, что микроконтроллер программируется после установки его на плату. В корпусе данного вида микроконтроллеров не требуется делать окно для стирания, что позволяет их упаковывать в недорогой пластмассовый корпус.

EEPROM микроконтроллеры.

EEPROM микроконтроллеры позволяют выполнять электрическое стирание памяти. Устройство может быть разработано таким образом, что микроконтроллер программируется после установки его на плату. В корпусе данного вида микроконтроллеров не требуется делать окно для стирания, что позволяет их упаковывать в недорогой пластмассовый корпус.

ROM микроконтроллеры.

Код программы в память программ ROM микроконтроллеров заносится на этапе изготовления кристалла. Память программ этих микроконтроллеров не может быть изменена. ROM микроконтроллеры могут быть упакованы в недорогой пластмассовый корпус.

DIE микроконтроллеры.

Опция DIE позволяет минимизировать размер печатной платы. Использование микроконтроллеров DIE требует определенный уровень технологического оборудования и квалификации специалистов. Это означает, что возможность использования DIE технологии ограничена.

Контрольные вопросы.

1. Какие основные особенности отличают различные типы микроконтроллеров?
2. Что такое периферийные модули контроллера, и каково их назначение при проектировании определенного функционального устройства?
3. В чем заключается отличие FLASH-микроконтроллеров от других приборов, с иным принципом записи исполнительной программы?

2 ТАКТОВЫЙ ГЕНЕРАТОР

Для формирования тактового сигнала микроконтроллера предусмотрен внутренний генератор. Тактовый сигнал необходим для выполнения инструкций микроконтроллера и работы периферийных модулей. Внутренний машинный цикл микроконтроллера (TCY) состоит из четырех периодов тактового сигнала.

Тактовый генератор микроконтроллера может работать в одном из восьми режимов. Существует два режима внутреннего RC генератора, отличающихся между собой режимом работы вывода микроконтроллера (вывод микроконтроллера работает как CLKOUT или как универсальный порт ввода/вывода). Режим работы тактового генератора определяется битами в слове конфигурации, расположенными в энергонезависимой памяти. Настроить биты конфигурации можно только при программировании микроконтроллера. Возможные режимы тактового генератора:

- LP - низкочастотный кварцевый резонатор (пониженное энергопотребление);
- XT - стандартный кварцевый/керамический резонатор;
- HS - высокочастотный кварцевый резонатор;
- RC - внешний резистор/конденсатор (идентичен EXTRC с CLKOUT);
- EXTRC - внешний резистор/конденсатор;
- EXTRC - внешний резистор/конденсатор с CLKOUT;
- INTRC - внутренний резистор/конденсатор (4МГц);
- INTRC - внутренний резистор/конденсатор (4МГц) с CLKOUT;

Различные режимы тактового генератора позволяют использовать один тип микроконтроллеров в приложениях с разными требованиями к генератору. RC режим генератора снижает стоимость устройства, а LP режим генератора имеет меньшее энергопотребление. С помощью битов конфигурации устанавливается требуемый режим тактового генератора.

2.1 Режимы тактового генератора

Среднее семейство микроконтроллеров PICmicro может иметь до восьми

режимов тактового генератора. Для выбора режима тактового генератора пользователь должен запрограммировать до трех битов конфигурации (FOSC2, FOSC1 и FOSC0). Основным отличием между режимами LP, XT и HS является значение коэффициента усиления инвертора внутренней схемы генератора. В таблице 2-1 и 2-2 указан допустимый диапазон частоты тактового генератора в различных режимах работы. Рекомендуется использовать режим тактового генератора с минимальным коэффициентом усиления для выбранной частоты, что позволяет получить меньший динамический ток потребления (I_{DD}). При выборе режима и частоты тактового генератора необходимо учитывать рекомендуемый диапазон частот и выполнение дополнительных требований (напряжение питания, рабочая температура, параметры компонентов (резистор, конденсатор, внутренняя схема генератора микроконтроллера)).

Режимы тактового генератора RC и EXTRC с CLKOUT имеют одинаковые функциональные особенности. Они имеют разные названия, чтобы облегчить описание других режимов генератора.

Таблица 2.1 - Выбор режима тактового генератора для микроконтроллеров с FOSC1:FOSC0

Биты конфигурации FOSC1:FOSC0	**Режим генератора**	**Коэффициент усиления инвертора**	**Описание**
11	RC	-	Минимальная стоимость тактового генератора (требуется только внешний резистор и конденсатор). Большой диапазон частот тактового генератора. Режим работы по умолчанию.
10	HS	Высокий	Для приложений с высокойчастотой тактового генератора. Высокий ток потребления тактового генератора из трех режимов с кварцевым резонатором.
01	XT	Средний	Стандартная частота кварцевого/керамического резонатора. Средний ток потребления тактового генератора из трех режимов с кварцевым резонатором.
00	LP	Низкий	Для приложений с низкой частотой тактового генератора. Низкий ток потребления тактового генератора из трех режимов с кварцевым резонатором.

Таблица 2.2 - Выбор режима тактового генератора для микроконтроллеров с FOSC2:FOSC0

Биты конфигурации FOSC2:FOSC0	**Режим генератора**	**Коэффициент усиления инвер-**	**Описание**
111	EXTRC с CLKOUT	-	Минимальная стоимость тактового генератора. Большой диапазон частот тактового генератора. Функция CLKOUT включена. Режим работы по умолчанию.

110	EXTRC	-	Минимальная стоимость тактового генератора. Большой диапазон частот тактового генератора. Функция CLKOUT выключена (вывод используется как порт ввода/вывода).
101	INTRC с CLKOUT	-	Минимальная стоимость тактового генератора. Настраиваемый генератор 4МГц. Функция CLKOUT включена.
100	INTRC	-	Минимальная стоимость тактового генератора. Настраиваемый генератор 4МГц. Функция CLKOUT выключена (вывод используется как порт ввода/вывода).
011	-	-	Резерв.
010	HS	Высокий	Для приложений с высокой частотой тактового генератора. Высокий ток потребления тактового генератора из трех режимов с кварцевым резонатором.
001	XT	Средний	Стандартная частота кварцевого/керамического резонатора. Средний ток потребления тактового генератора из трех режимов с кварцевым резонатором.
000	LP	Низкий	Для приложений с низкой частотой тактового генератора. Низкий ток потребления тактового генератора из трех режимов с кварцевым резонатором.

2.2 Кварцевый/керамический резонатор

В режимах тактового генератора XT, LP и HS кварцевый или керамический резонатор подключается к выводам OSC1, OSC2 (см. рисунок 2.1). Для микроконтроллеров PICmicro нужно использовать резонаторы с параллельным резонансом. Использование резонаторов с последовательным резонансом может привести к получению тактовой частоты, не соответствующей параметрам резонатора. В режимах XT, LP и HS микроконтроллер может работать от внешнего источника тактового сигнала OSC1 (см. рисунок 2.3).

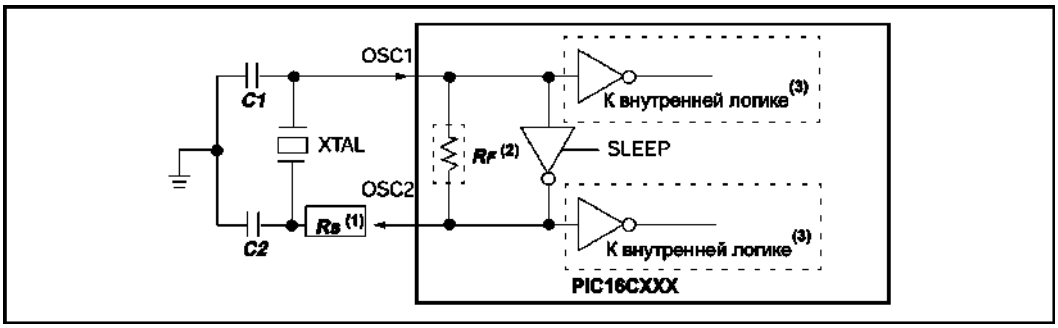

Рисунок 2.1 - Подключение кварцевого/керамического резонатора в HS, XT и LP режиме тактового генератора

Как только напряжение питания микроконтроллера станет выше V_{SS}, тактовый генератор начнет генерацию сигнала. Время, необходимое для запуска генератора, зависит от большого числа факторов:

1. Частота кварцевого керамического резонатора;

2. Емкость конденсаторов C1 и C2 (см. рисунок 2-1);

3. Скорость нарастания напряжения питания VDD;

4. Рабочая температура;

5. Сопротивление резистора Rs, если подключен (см. рисунок 2-1);

6. Режим тактового генератора (коэффициент усиления внутреннего инвертора);

7. Качество резонатора;

8. Размещение компонентов тактового генератора на печатной плате;

9. Помехи.

На рисунке 2.2 показана временная диаграмма запуска тактового генератора с кварцевым/керамическим резонатором. Напряжение питания, при котором форма сигнала тактового генератора удовлетворяет требованиям, может составлять 50% от номинального напряжения питания. Постоянная составляющая тактового сигнала равна $V_{DD/2}$.

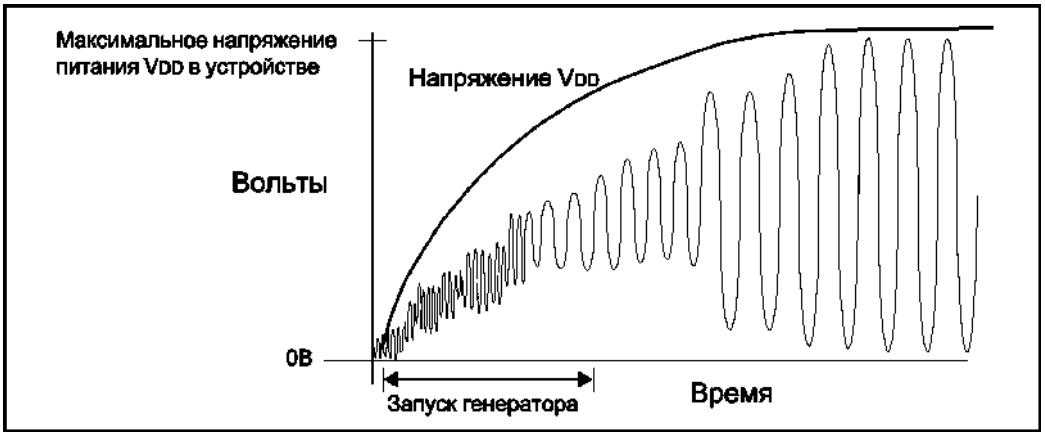

Рисунок 2.2 - Временная диаграмма запуска тактового генератора с кварцевым/керамическим резонатором

2.3 Выбор компонентов

На рисунке 2.1 показана схема подключения кварцевого/керамического резонатора к микроконтроллеру. Значение резистора обратной связи R_F внутреннего инвертора колеблется от 2МОм до 10МОм в зависимости от режима генератора, напряжения питания и температуры. Последовательный резистор Rs может потребоваться для предотвращения возбуждения резонатора на низкой частоте. Убедитесь, что напряжение питания микроконтроллера и режим работы микроконтроллера соответствуют требованиям резистора. Внутренняя логика микроконтроллера может быть подключена к входу или выходу инвертора тактового генератора (см.

рисунок 2.1). Типовые значения емкости конденсаторов C1, C2 для кварцевого/керамического резонатора представлены в таблице 2.3 и 2.4. Точное значение емкости конденсаторов для конкретного микроконтроллера смотрите в соответствующей технической документации.

Таблица 2.3 - Параметры конденсаторов для керамического резонатора

Протестированные диапазоны:		
Режим	**Частота**	**C1 / C2**
XT	455 кГц 2.0 МГц 4.0 МГц	22 - 100 пФ 15 - 68 пФ 15 - 68 пФ
HS	8.0 МГц 16.0 МГц 20.0 МГц	10 - 68 пФ 10 - 22 пФ TBD
Применяемые резонаторы:		
455 кГц	Panasonic EFO-A455K04B	±0.3%
2.0 МГц	Murata Erie CSA2.00MG	±0.5%
4.0 МГц	Murata Erie CSA4.00MG	±0.5%
8.0 МГц	Murata Erie CSA8.00MT	±0.5%
16.0 МГц	Murata Erie CSA16.00MX	±0.5%
20.0 МГц	TBD	TBD

Таблица 2.4 - Параметры конденсаторов для кварцевого резонатора

Протестированные диапазоны:			
Режим	**Частота**	C1	C2
LP	32 кГц 200 кГц	68 - 100 пФ 15 - 30 пФ	68 - 100 пФ 15 - 30 пФ
XT	100 кГц 2.0 МГц 4.0 МГц	68 - 150 пФ 15 - 30 пФ 15 - 30 пФ	68 - 150 пФ 15 - 30 пФ 15 - 30 пФ
HS	8.0 МГц 10.0 МГц 20.0 МГц	15 - 30 пФ 15 - 30 пФ 15 - 30 пФ	15 - 30 пФ 15 - 30 пФ 15 - 30 пФ
Применяемые резонаторы:			
32.768 кГц	Epson C001R32.768-A	±20 PPM	
100 кГц	Epson C-2 100.00 KC-P	±20 PPM	
200 кГц	STD XTL 200.000 kHz	±20 PPM	
2.0 МГц	ECS ECS-20-S-2	±50 PPM	
4.0 МГц	ECS ECS-40-S-4	±50 PPM	
8.0 МГц	ECS ECS-80-S-4	±50 PPM	
10.0 МГц	ECS ECS-100-S-4	±50 PPM	
20.0 МГц	ECS ECS-200-S-4	±50 PPM	

Примечание. Большая емкость увеличивает стабильность генератора, но увеличивается и время запуска. Последовательный резистор Rs может потребоваться в HS и XT режиме для предотвращения возбуждения резонатора на низкой частоте. Значения емкости конденсаторов являются оценочными, т.к. каждый резонатор имеет собственные характеристики.

2.4. Настройка схемы генератора

Существует большое число факторов, влияющих на работу тактового генератора: рабочий диапазон (частота тактового генератора, напряжение питания, рабочая температура и режим работы); внешние компоненты (резонатор, конденсаторы); качество изделия и микроконтроллера. Поэтому необходимо выполнять проверку выбранных параметров, чтобы гарантировать выполнение требований приложения.

При выборе внешних компонентов необходимо учитывать большое число факторов:

- Коэффициент усиления внутреннего инвертора генератора;
- Требуемая частота;
- Частота резонатора;
- Рабочая температура;
- Диапазон напряжения питания;
- Время запуска;
- Стабильность частоты;
- Наработка микроконтроллера;
- Потребляемая мощность;
- Упрощение схемы;
- Использование стандартных компонентов;
- Схема с минимальным числом внешних компонентов.

Рекомендации по выбору кварцевого резонатора, режима тактового генератора, C1, C2 и Rs. Наилучший метод выбора внешних компонентов можно сформулировать так: на основе несложных правил создать схему, провести испытания и тестирование устройства. Кварцевый резонатор необходимо выбирать с параллельным резонансом, но в вашем проекте могут потребоваться и другие параметры резонаторов (например, температурный дрейф или стабильность частоты).

Внутренний тактовый генератор PICmicro выполнен по схеме параллельного генератора, требующей использования кварцевых резонаторов с параллельным резонансом. Типовое значение емкости нагрузочных конденсаторов от 20пФ до 32пФ. Частота тактового сигнала, при указанной емкости нагрузочных конденсаторов будет наиболее близкой к требуемой. Иногда может возникнуть необходимость

изменить частоту генерации в небольших пределах для достижения других целей (эта возможность будет описана позже).

Режим работы тактового генератор выбирается в соответствии с технической документацией на микроконтроллер битами FOSC. Выбор режима тактового генератора (кроме RC) заключается в выборе коэффициента усиления инвертора генератора (малый коэффициент усиления - низкая частота, большой коэффициент усиления - высокая частота тактового генератора). Допускается выбирать высокий коэффициент усиления внутреннего инвертора для реализации определенных требований к схеме тактового генератора.

Первоначально емкость конденсаторов C1 и C2 выбирается в соответствии с требованиями производителя кварцевого резонатора и представленными в технической документации на микроконтроллер таблицами. Значение емкости конденсаторов, указанное в технической документации на микроконтроллер, может использоваться только как отправная точка, т.к. технология изготовление резонатора, напряжение питания и другие уже упомянутые факторы могут изменить параметры работы резонатора в вашей схеме по сравнению с заявленными производителем.

В идеале, значение емкости конденсаторов должно выбираться с учетом максимально возможной рабочей температуры и минимально возможного напряжения питания VDD (в пределах, рекомендуемых изготовителем резонатора). Высокая температура и низкое напряжение питания имеют воздействие на коэффициент усиления инвертора, поэтому если устройство устойчиво работает в этом режиме, то проектировщик может быть более уверен в нормальной работе устройства при других комбинациях рабочей температуры и напряжения питания. Синусоидальный сигнал не должен ограничиваться при самом высоком коэффициенте усиления (самое высокое V_{DD} и самое низкая температура) и амплитуда сигнала на выводе должна быть достаточно большой при минимальном коэффициенте усиления инвертора (самое низкое V_{DD} и самая высокая температура), чтобы удовлетворять требованиям логического входа тактового сигнала микроконтроллера, описанным в технической документации.

Метод улучшенного запуска тактового генератор заключается в том, что значение емкости C2 выбирается больше, чем C1. Это вызывает большой сдвиг фазы

при включении питания, ускоряя запуск генератора.

Помимо нагрузки резонатора для стабилизации частоты генератора конденсаторы C1 и C2 могут иметь эффект понижения коэффициента усиления при увеличении емкости. Значение емкости C2 может быть изменено с целью изменения коэффициента усиления инвертора. Большее значение емкости C2 может понизить коэффициент усиления до полного прекращения генерации. Значение емкости конденсаторов C1 и C2 не должно быть очень большим, поскольку это может привести к значительному снижению тока через резонатор.

Последовательный резистор Rs необходимо использовать, если подбор внешних компонентов не дал удовлетворительной работы тактового генератора. Резистор может быть подключен к выводу OSC2, к которому подключен осциллограф. Подключение осциллографа к выводу OSC1 может являться причиной срыва генерации, поскольку он будет оказывать существенное влияние на отрицательную обратную связь инвертора. Необходимо учитывать то, что подключение измерительного оборудования добавляет собственную емкость к схеме. Например, если тактовый генератор лучше всего работал при емкости 20пФ и был подключен измерительный прибор с емкостью входа 10пФ, то необходимо устанавливать конденсатор в 30пФ. Сигнал с выхода не должен ограничиваться или нагружаться измерительной цепью. Перевозбуждение резонатора может привести к переходу генерации сигнала на более высокой гармонике или повреждению резонатора.

На выводе OSC2 должен присутствовать сигнал в виде чистой синусоиды, размах которой легко достигает минимального и максимального значения сигнала на тактовом входе (хороший тактовый сигнал - уровень сигнала от 4В до 5В при напряжении питания 5В). Необходимо добиться указанных параметров тактового сигнала, а затем проверить схему при минимальной температуре и максимальном V_{DD}, ожидаемом в проекте. В этом режиме будет получена максимальная амплитуда тактового сигнала. Если происходит ограничение амплитуды или постоянная составляющая смещена к V_{DD} или к V_{SS}, а значение емкости конденсаторов значительно превысило рекомендованное производителем резонатора, необходимо подключить переменный резистор между выводом микроконтроллера и конденсатором C2. Переменным резистором добиться "чистого" синусоидального сигнала. При низкой температуре и вы-

соком напряжении питания получается максимальная амплитуда тактового сигнала, что гарантирует предотвращение перевозбуждения. Вместо переменного резистора должен быть установлен постоянный резистор с наиболее близким сопротивлением. Если сопротивление Rs более 20кОм, вход более изолирован от выхода, что делает схему более восприимчивой к шуму. Если принято решение, что необходимо большое сопротивление Rs, то для предотвращения перевозбуждения попробуйте увеличить емкость C2. Попытайтесь получить комбинацию, в которой сопротивление Rs не более 10кОм, а значение нагрузочных конденсаторов не сильно отличается от 20пФ до 32пФ или спецификации изготовителя резонатора.

Запуск генератора. Наиболее сложные условия запуска тактового генератора возникают при выходе из режима SLEEP, потому что нагрузочные конденсаторы заряжены до некоторого постоянного значения и дифференциальная фаза при выходе из SLEEP минимальна. Поэтому требуется больший интервал времени для достижения устойчивого колебания. Необходимо также учитывать, что малое напряжение питания, высокая рабочая температура и низкая тактовая частота накладывают ограничения на коэффициент усиления инвертора генератора, который в свою очередь воздействует на запуск генерации. Каждый из следующих факторов усложняет запуск генератора:

- Низкая частота тактового генератора (низкий коэффициент усиления инвертора генератора);
- Отсутствие шума по цепи питания (устройства с батарейным питанием);
- Эксплуатация устройства в условиях малого электромагнитного шума
- Малое напряжение питания;
- Высокая температура;
- Выход из режима SLEEP.

Электромагнитные помехи или помехи по цепи питания могут служить стартовым импульсом при запуске генератора.

2.5 Внешний тактовый сигнал

Если внутренний тактовый генератор не используется, а тактовый сигнал генерируется внешней схемой, необходимо выбрать один из режимов LP, XT или HS.

Т.е. любой режим, кроме RC, т.к. в RC режиме генератора будет возникать наложение двух сигналов. В идеале нужно выбирать режим тактового генератора, соответствующий частоте внешнего тактового сигнала, но в данном случае это имеет меньшую важность, т.к. внешний тактовый сигнал сразу поступает на внутреннюю логику микроконтроллера без использования цепей внутреннего генератора. Допускается выбирать режим тактового генератора с меньшим диапазоном тактовых частот, чем частота тактового сигнала, с целью снижения потребляемого тока. Необходимо удостоверится, что амплитуда сигнала на выходе OSC2 удовлетворяет требования логических уровней микроконтроллера.

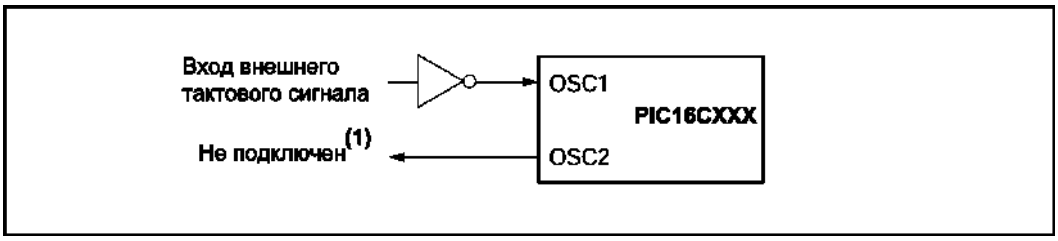

Рисунок 2.3 - Подключение внешнего тактового сигнала в HS, XT и LP режиме тактового генератора

Внешний тактовый генератор. Иногда требуется, чтобы более одного устройства работало от тактового генератора. Фирма Microchip не рекомендует подключать дополнительные цепи к внутренней схеме тактового генератора, поэтому должна использоваться внешняя схема генератора. В этом случае каждое устройство схемы будет иметь внешний генератор тактового сигнала. Число подключаемых устройств зависит от нагрузочной способности выходного буфера тактового генератора. Применение внешнего тактового генератора полезно, когда необходимо синхронизировать работу нескольких устройств.

В качестве внешнего тактового генератора можно использовать готовый генератор, либо собрать простую схему с ТТЛ выходом. Качественный кварцевый резонатор обеспечивает высокую эффективность ТТЛ схемы. Существует две основных схемы включения кварцевых резонаторов: с параллельным резонансом, с последовательным резонансом.

На рисунке 2.4 показана типовая схема генератора с параллельным резонансом, предназначенная для работы на основной частоте кварцевого резонатора. Инвертор 74AS04 производит необходимый для параллельного резонанса сдвиг фазы

на 180^0. Для обеспечения стабильности схемы в отрицательной обратной связи включен резистор 4.7кОм. Потенциометр 10кОМ предназначен для смещения рабочей точки инвертора в линейную область.

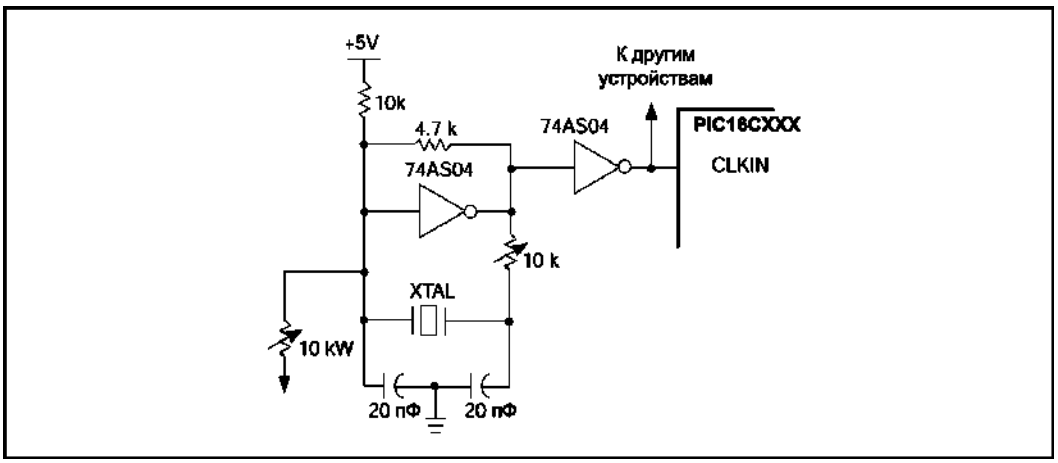

Рисунок 2.4 - Внешний генератор с параллельным резонансом

На рисунке 2-5 показана типовая схема генератора с последовательным резонансом, тоже предназначенная для работы на основной частоте кварцевого резонатора. Инверторы выполняют сдвиг фазы на 180^0. Резисторы 330кОм создают отрицательную обратную связь для смещения рабочих точек инверторов в линейную область.

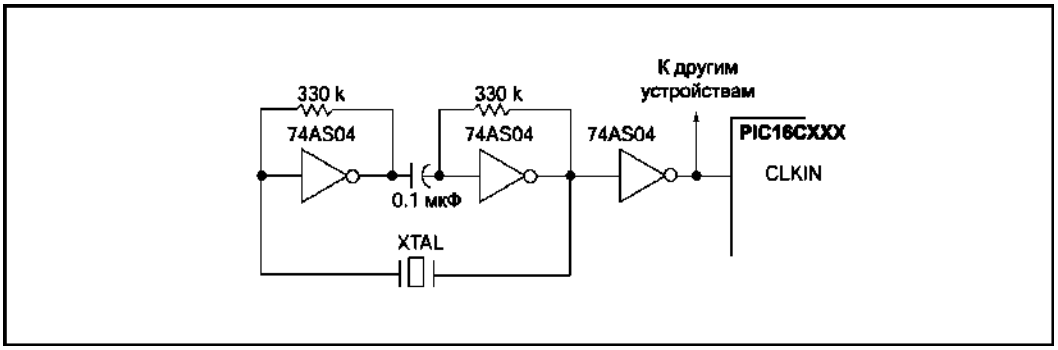

Рисунок 2.5 - Внешний генератор с последовательным резонансом

Когда микроконтроллер работает от внешнего источника тактового сигнала (рисунки 2.4, 2.5), тактовый генератор должен быть настроен в режиме HS, XT или LP (рисунок 2.3).

Внешний RC генератор. В приложениях, не требующей высокостабильной тактовой частоты, возможно использовать RC (EXTRC) режим генератора, уменьшающий стоимость устройства. Частота RC генератора зависит от напряжения пи-

тания, значения сопротивления (REXT), емкости (CEXT) и рабочей температуры. Дополнительно частота будет варьироваться в некоторых пределах из-за технологического разброса параметров кристалла. Различные паразитные емкости также будут влиять на частоту генератора, особенно при малых значениях CEXT. Необходимо учитывать технологический разброс параметров внешних компонентов R и C.

На рисунке 2-6 показана схема подключения RC цепочки к PIC16CXXX. Для сопротивления резистора меньше 2.2кОм частота тактового генератора может быть нестабильна или генерация может прекратиться. Для очень большого сопротивления (больше 1МОм) генератор тактового сигнала становится чувствителен к внешним помехам, токам утечки и влажности. Рекомендуется выбирать сопротивления резисторов от 3кОм до 100кОм.

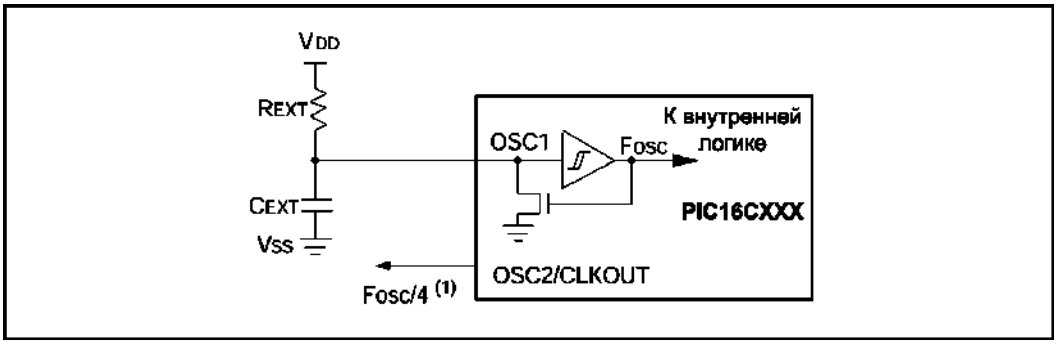

Рисунок 2.6 - EXTRC режим тактового генератора

Тактовый генератор может работать без внешнего конденсатора (CEXT=0пФ), но для стабильной работы генератора рекомендуется подключать конденсатор с емкостью более 20пФ. Без внешнего конденсатора (или конденсатор имеет очень малую емкость) частота тактового генератора может зависеть от емкости проводников печатной платы и выводов компонентов.

В разделе электрических характеристик PIC-микроконтроллеров представлены данные технологического разброса частоты RC генератора. Разброс частоты возрастает с увеличением сопротивления R (т.к. возрастает влияние токов утечки) и уменьшением емкости C (т.к. усиливается влияние паразитной емкости проводников и выводов компонентов). Для испытательных целей или для синхронизации внешней логики на выводе OSC2/CLKOUT присутствует тактовый сигнал с частотой FOSC/4.

Запуск RC генератора. Внешний RC генератор немедленно начнет формировать тактовый сигнал после достижения порогового уровня напряжения на выводах микроконтроллера. Время запуска RC генератора зависит от большого числа факторов, вот основные из них:

- Сопротивление внешнего резистора;
- Емкость внешнего конденсатора;
- Скорость нарастания напряжения питания;
- Температура.

Внутренний RC генератор 4МГц. Внутренний тактовый генератор (не для всех микроконтроллеров) формирует тактовый сигнал с частотой 4МГц (номинальное значение) при напряжении питания VDD=5В и температуре 25^0С. Запись калибровочной константы в регистр OSCCAL позволяет устранить технологический разброс параметров внутреннего RC генератора. Биты CAL3:CAL0 используются для настройки частоты тактового генератора в пределах окна калибровки. Увеличение значения битов CAL3:CAL0 (от 0000 до 1111) приводит к увеличению тактовой частоты.

Если тактовая частота 4МГц внутреннего RC генератора не может быть достигнута изменением битов CAL3:CAL0, то частота тактового генератора может быть смещена битами CALFST и CALSLW. Эти биты дают возможность выполнить положительное или отрицательное смещение окна калибровки частоты тактового генератора. Установка бита CALFST в '1' смещает окно калибровки к более высоким частотам RC генератора, а установка бита CALSLW в '1' смещает окно калибровки к более низкой частоте.

При сбросе микроконтроллера в регистр OSCCAL загружается значение, соответствующее среднему положению калибровки (CAL3:CAL0 = 7h, CALFST и CALSLW не создают смещения).

Регистр OSCCAL

R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	U-0	U-0
CAL3	**CAL2**	**CAL1**	**CAL0**	**CALFST**	**CALSLW**	**-**	**-**
Бит 7							Бит 0

R – чтение бита

W – запись бита

U – не реализовано, читается как 0

–n – значение после POR

–x – неизвестное значение после POR

биты 7-4: CAL3:CAL0: Биты калибровки внутреннего RC генератора. 0000 = наименьшая частота в пределах окна калибровки; 1111 = наивысшая частота в пределах окна калибровки.

бит 3: CALFST: Бит смещения окна калибровки внутреннего RC генератора. 1 = увеличение частоты тактового генератора; 0 = смещения нет

бит 2: CALSLW: Бит смещения окна калибровки внутреннего RC генератора 1 = уменьшение частоты тактового генератора; 0 = смещения нет

биты 1-0: Не реализованы: читаются как '0'

Примечание 1. Если бит CALFST = 1, то значение бита CALSLW игнорируется.

Примечание 2. При записи в регистр OSCAL эти биты всегда должны равняться '0' для совместимости с последующими версиями микроконтроллеров.

Запись калибровочной константы в регистр OSCCAL позволяет устранить технологический разброс параметров внутреннего RC генератора. Калибровочное значение, сохраненное компанией Microchip, не должно изменяться. Все функции отсчета времени должны быть откорректированы программным обеспечением.

На рисунке 2.7 показана возможная частота некалиброванного тактового генератора (VDD = 5В, 25^0C, OSCCAL=07h) и реализуемое смещение частоты генератора за счет изменения значения регистра OSCCAL.

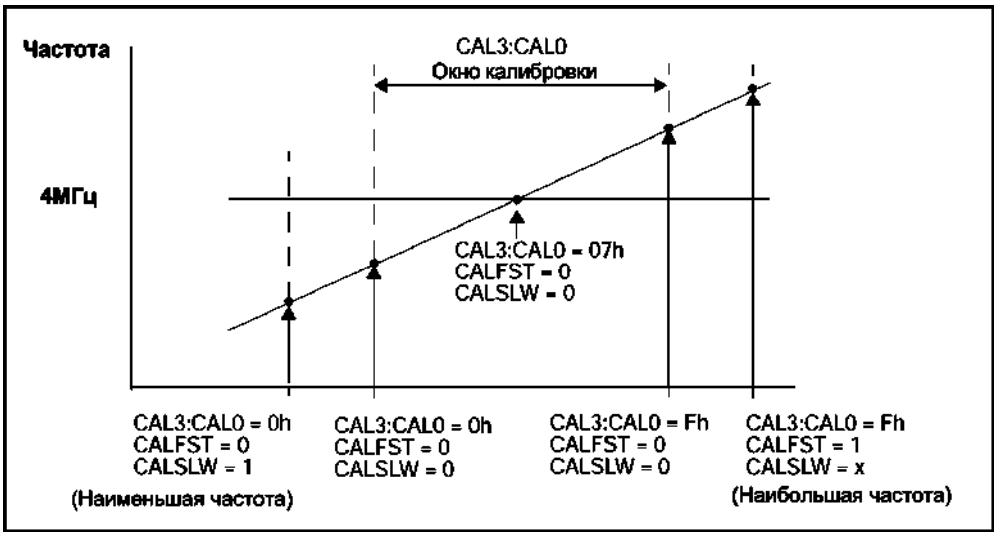

Рисунок 2.7 - Идеальная частота внутреннего RC генератора в зависимости от значения регистра OSCCAL

На рисунке 2.8 показан пример, в котором частота тактового генератора исправляется к 4МГц за счет изменения битов CAL3:CAL0. В данном случае удалось скорректировать частоту тактового генератора, изменяя только биты CAL3:CAL0. Иногда частота внутреннего RC генератора не может быть скорректирована к 4МГц изменением битов CAL3:CAL0. Поэтому были предусмотрены два дополнительных (CALSLW и CALFST), создающих большее смещение частоты генератора. После грубого смещения частоты RC генератора можно выполнить точную подстройку битами CAL3:CAL0. Действие битов CALFST и CALSLW показано на рисунках 2.9, 2.10.

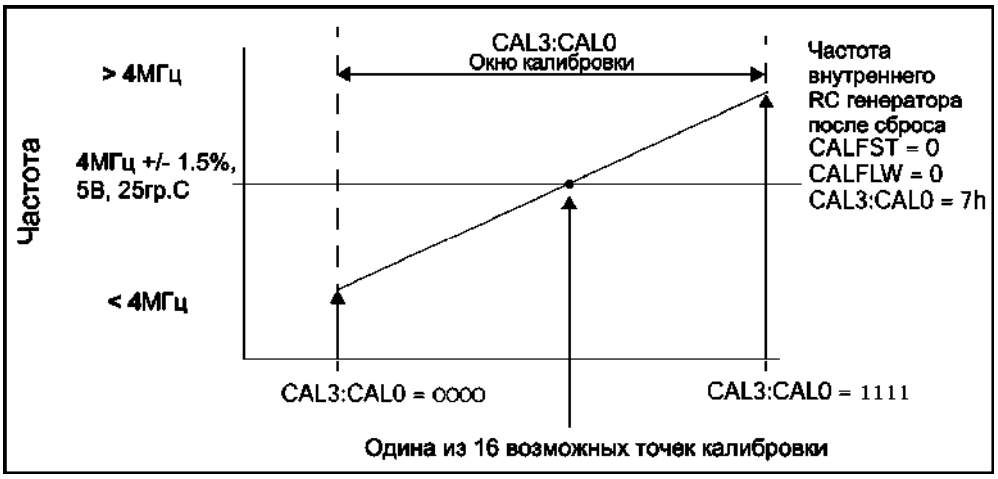

Рисунок 2.8 - Корректировка частоты внутреннего RC генератора с помощью битов CAL3:CAL0

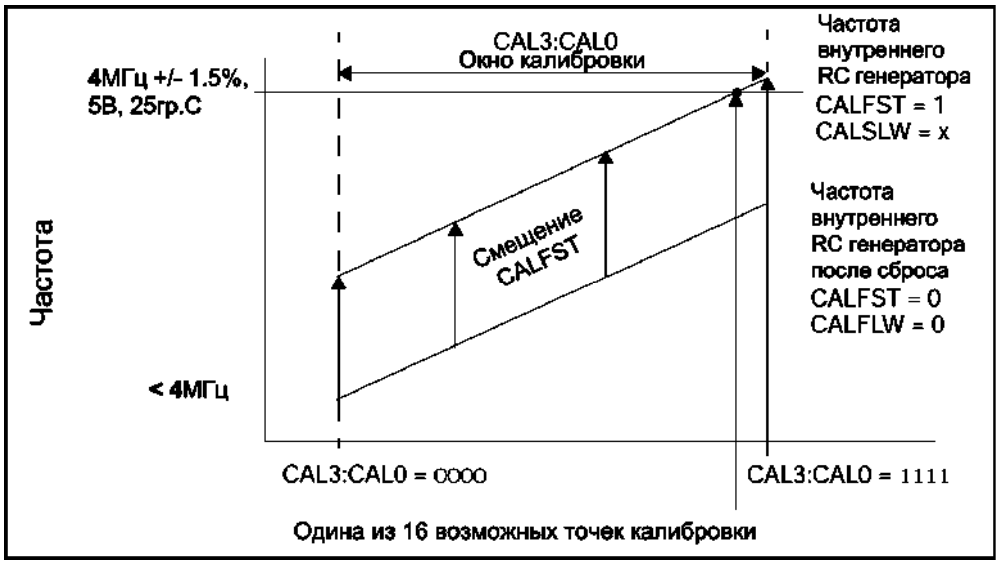

Рисунок 2.9 - CALFST положительное смещение частоты внутреннего RC генератора

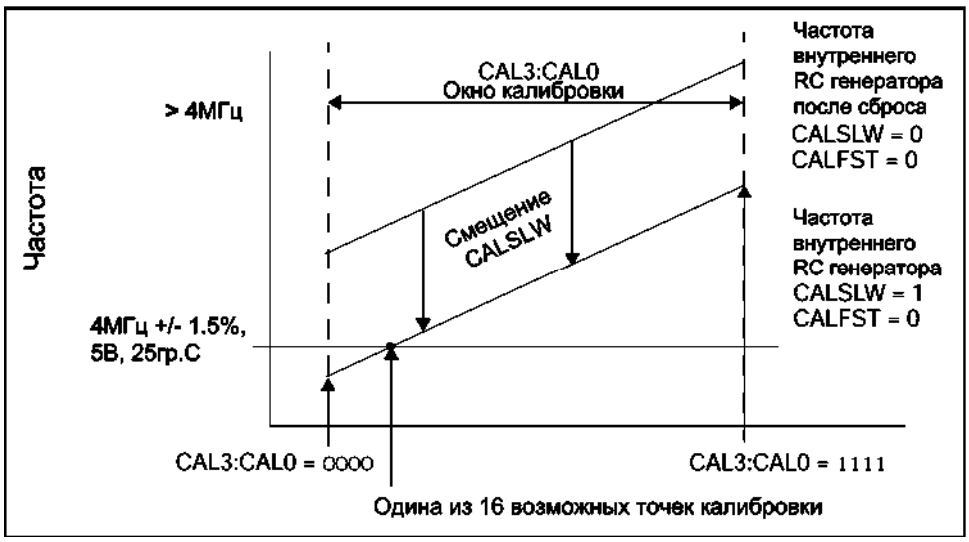

Рисунок 2.10 - CALSLW отрицательное смещение частоты внутреннего RC генератора

В последней ячейки памяти программ сохраняется калибровочная константа для внутреннего RC генератора. Калибровочная константа сохраняется в виде команды RETLW XX, где XX - калибровочное значение. Чтобы загрузить калибровочную константу выполните инструкцию CALL YY, где YY - последняя доступная пользователю ячейка памяти программ. Значение калибровки будет загружено в регистр W. Затем необходимо выполнить инструкцию MOVWF OSCCAL, чтобы загрузить калибровочную константу в регистр калибровки внутреннего RC генератора. В таблице 2.5 показано расположение калибровочной константы в зависимости от объема памяти программ.

Таблица 2.5. Размещение калибровочной константы

Объем памяти программ (слов)	Адрес калибровочной константы
512	1FFh
1к	3FFh
2к	7FFh
4к	FFFh
8к	1FFFh

Стирание памяти микроконтроллера (с УФ стиранием памяти) также сотрет предварительно запрограммированную калибровочную информацию. Для сохранения калибровочной информации ее рекомендуется прочитать перед стиранием памяти микроконтроллера. Калибровочная информация должна быть восстановлена перед программированием микроконтроллера.

Выход тактового сигнала. Внутренний RC генератор может быть настроен для работы в режиме формирования на выводе CLKOUT тактового сигнала с частотой FOSC/4 (биты FOSC2, FOSC1, FOSC0 в слове конфигурации (адрес 2007h) должны равняться '101' для внутреннего RC генератора и '111' для внешнего RC генератора). Выход тактового сигнала может использоваться для измерения частоты генератора или синхронизации внешней логики.

Если калибровочная информация внутреннего RC генератора стерта, входной тактовый сигнал позволяет скорректировать частоту генератора. Это может быть реализовано написанием дополнительной программы, изменяющей значение регистра OSCCAL. Когда на выводе CLKOUT присутствует сигнал с частотой 1 МГц (±1.5%) при VDD = 5В и температуре 25^0C, то значение в регистре OSCCAL правильное. Это значение должно быть передано через порты ввода/вывода для сохранения его в калибровочной ячейки памяти программ.

2.6 Воздействие режима SLEEP на тактовый генератор

При выполнении инструкции SLEEP тактовый генератор выключается, а внутренняя логика микроконтроллера переводится в начало цикла команды (такт Q1). При выключенном тактовом генераторе на выводах OSC1 и OSC2 не будет наблюдаться гармонических колебаний. Т.к. тактовый генератор выключен, то в SLEEP режиме микроконтроллера достигается наименьший ток потребления (только токи утечек). Включение любого периферийного модуля, работающего в SLEEP

режиме микроконтроллера, увеличит суммарный ток потребления. Микроконтроллер может выйти из режима SLEEP по внешнему сбросу, переполнению сторожевого таймера WDT или возникновению прерывания.

Таблица 2.6. Состояние выводов OSC1 и OSC2 в SLEEP режиме.

Режим генератора	Вывод OSC1	Вывод OSC2
EXTRC	Свободное состояние, внешний резистор должен подтянуть к напряжению питания	Низкий логический уровень
INTRC	-	-
LP, XT и HS	Выключена обратная связь инвертора, постоянное напряжение	Выключена обратная связь инвертора, постоянное напряжение

Воздействие сброса микроконтроллера на тактовый генератор. Сброс микроконтроллера не вызывает никакого воздействия на работу тактового генератора. Он продолжает работать, поскольку сброс произошел в нормальном режиме микроконтроллера. Сброс переводит логику микроконтроллера в исходное состояние (в начало цикла команды, такт Q1).

В режиме внешнего RC генератора (EXTRC) при использовании сигнала CLKOUT на выводе OSC2 будет присутствовать низкий логический уровень сигнала, пока на -MCLR активный уровень. Как только на выводе -MCLR появится высокий логический уровень VIH , RC генератор начнет формировать тактовый сигнал.

Задержка сброса микроконтроллера при включении питания. Два дополнительных таймера выполняют задержку старта работы микроконтроллера. Первый, таймер запуска генератора (OST), удерживает микроконтроллер в состоянии сброса пока не стабилизируется частота тактового генератора. Второй - таймер включения питания (PWRT), срабатывает после включения питания и удерживает микроконтроллер в состоянии сброса в течение 72мс (типовое значение), пока не стабилизируется напряжение питания. В большинстве приложений эти функции микроконтроллера позволяют исключить внешние схемы сброса.

Контрольные вопросы.

1. Что такое тактовый генератор микроконтроллера и какова его роль?

2. Какие возможности настройки тактового генератора имеют PIC-микроконтроллеры?

3. Чем отличается работа микроконтроллера в режимах внешнего и внутреннего тактовых сигналов?

4. Какими путями можно снизить энергопотребление электронной схемы на основе программируемого микроконтроллера?

5. Чем отличаются внутренние типы генераторов микроконтроллеров? В каком случае оправдано применение каждого из них?

6. Что такое биты конфигурации и какова их роль в настройке?

3 АРХИТЕКТУРА PIC-МИКРОКОНТРОЛЛЕРОВ

Высокая эффективность микроконтроллеров PICmicro достигается за счет архитектуры ядра, подобная архитектура обычно применяется в RISC микропроцессорах.

Основные особенности архитектуры микроконтроллеров PICmicro:

- Гарвардская архитектура;
- Длинное слово команды;
- Команда состоит из единственного слова;
- Конвейерная обработка команд;
- Команды выполняются за один машинный цикл;
- Небольшое число команд;
- Файловая структура данных;
- Все команды ортогональны (симметричны).

На рисунке 3.2 показана общая структурная схема микроконтроллеров PICmicro среднего семейства.

Гарвардская архитектура. В гарвардской архитектуре разделена память программ и память данных. Обращение к памяти происходит по отдельным шинам адреса и данных, что значительно повышает производительность процессора по сравнению с традиционной архитектурой.

В микроконтроллерах с традиционной архитектурой ядра команды и данные запрашиваются по одной и той же шине. Чтобы выполнить выборку команды необходимо сделать несколько запросов по 8-разрядной (или кратной 8 разрядам) шине. Затем (если необходимо) запросить данные, выполнить команду и сохранить результат. Как может быть замечено, шина с традиционной архитектурой ядра значительно загружена.

В микроконтроллерах с гарвардской архитектурой ядра выборка команды происходит за один цикл (все команды 14 - разрядные). При обращении к памяти программ можно выполнить запись или чтение данных, т.к. память данных подключена к ядру микроконтроллера по отдельной шине. Раздельные шины доступа к памяти программ и к памяти данных позволяют исполнять текущую команду и произ-

водить выборку следующей команды, организуя конвейерную обработку команд. Сравнение гарвардской и традиционной архитектуры показано на рисунке 3.1.

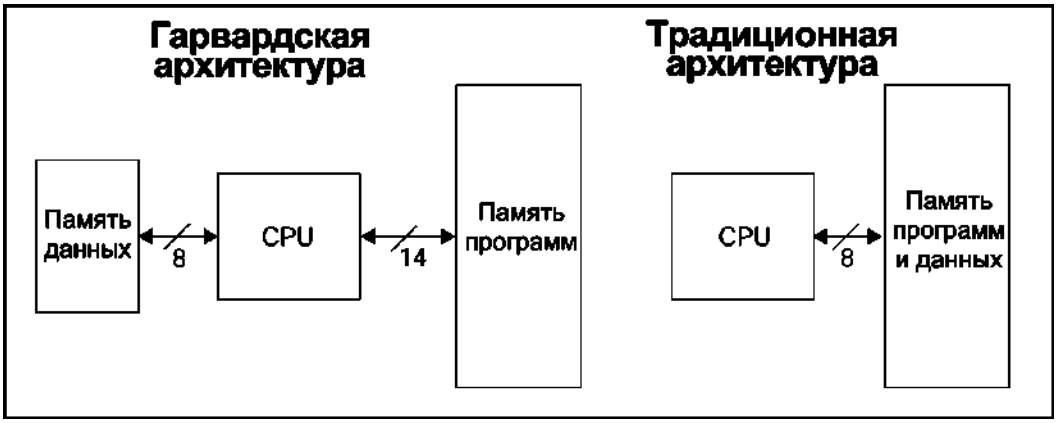

Рисунок 3.1 - Сравнение гарвардской и традиционной архитектуры

Длинное слово команды. Разрядность команд микроконтроллера несколько больше чем 8-разрядная шина памяти данных. Это стало возможным из-за раздельных шин доступа к памяти программ и к памяти данных. Разделение шин доступа к разным видам памяти позволяет произвольно выбирать разрядность команд микроконтроллера (не кратное 8-разрядной шине данных), что в свою очередь дает возможность эффективно использовать память программ и оптимизировать разрядность шины программ к архитектурным требованиям микроконтроллера.

Команда состоит из единственного слова. Все команды микроконтроллеров однословные 14 - разрядные. 14 - разрядная шина доступа к памяти программ позволят выполнить выборку 14 - разрядной команды за один машинный цикл микроконтроллера. При использовании однословных команд число слов в памяти программ равняется максимальному числу команд программы микроконтроллера. Это означает, что все ячейки памяти имеют силу команды.

Как правило, в традиционной архитектуре большинство команд многобайтные. Микроконтроллер, имеющий 4к байт памяти, содержит примерно 2к команд. Коэффициент использования памяти примерно равен 2:1 и зависит от конкретного приложения. Поскольку каждая команда может состоять из нескольких байтов, то нет никакой гарантии, что каждая ячейка памяти программ имеет силу команды.

Конвейерная обработка команд. Конвейерная обработка команд состоит из двух стадий: выборка команды из памяти, выполнение команды. Выборка команды происходит в первый машинный цикл TCY, а выполнятся команда во втором ма-

шинном цикле TCY. Однако из-за одновременной выборки текущей команды и выполнения предыдущей в каждом машинном цикле TCY происходит выборка и выполнение команд.

Команды выполняются за один машинный цикл. Полная выборка команды происходит за один машинный цикл (TCY) из-за того, что шина доступа к памяти программ 14 - разрядная. Каждая команда содержит всю необходимую информацию и выполняется за один машинный цикл. При выполнении команды может возникать задержка в один машинный цикл, если результат команды изменяет содержимое счетчика команд PC. Задержка в один машинный цикл необходима для выборки новой команды, которая должна быть выполнена.

Небольшое число команд. Когда система команд хорошо проработана и команды ортогональны (симметричны), то требуется меньшее число команд для решения всех необходимых задач. С меньшим числом команд изучение микроконтроллера значительно упрощается.

Файловая структура данных. Обращение к регистрам памяти данных можно выполнить прямой или косвенной адресацией. Все регистры специального назначения, включая счетчик команд PC, отображается в памяти данных.

Все команды ортогональны (симметричны). Ортогональная система команд дает возможность выполнить любую операцию с любым регистром памяти данных прямой или косвенной адресацией. В ортогональной системе команд малое количество "специальных команд", что упрощает изучение и программирование микроконтроллеров не теряя эффективности кода программы. В микроконтроллерах среднего семейства используется только две не ортогональные команды, реализующие особенности ядра.

Команда SLEEP - переводит микроконтроллер в режим пониженного энергопотребления.

Команда CLRWDT - подтверждает нормальную работу микроконтроллера, предотвращая сброс по переполнению сторожевого таймера WDT.

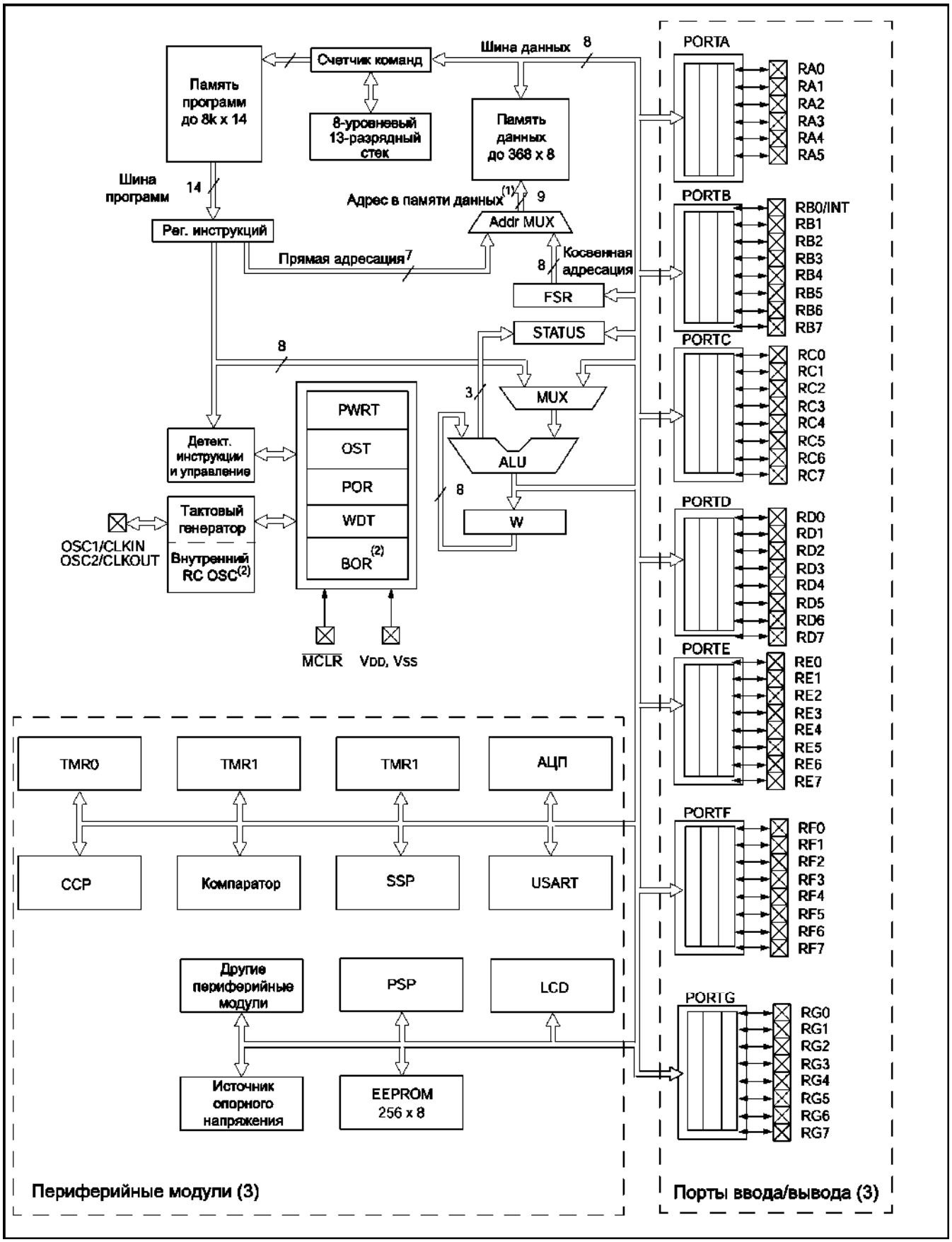

Рисунок 3.2 - Общая структурная схема микроконтроллеров PICmicro среднего семейства

Синхронизация выполнения команд. Входной тактовый сигнал (вывод OSC1) внутренней схемой микроконтроллера разделяется на четыре последовательных неперекрывающихся такта Q1, Q2, Q3 и Q4. Внутренний счетчик команд (PC) увеличивается на единицу в каждом такте Q1, а выборка команды из памяти программ происходит на каждом такте Q4. Декодирование и выполнение команды происходит с такта Q1 по Q4. На рисунке 3.3 показаны циклы выполнения команд.

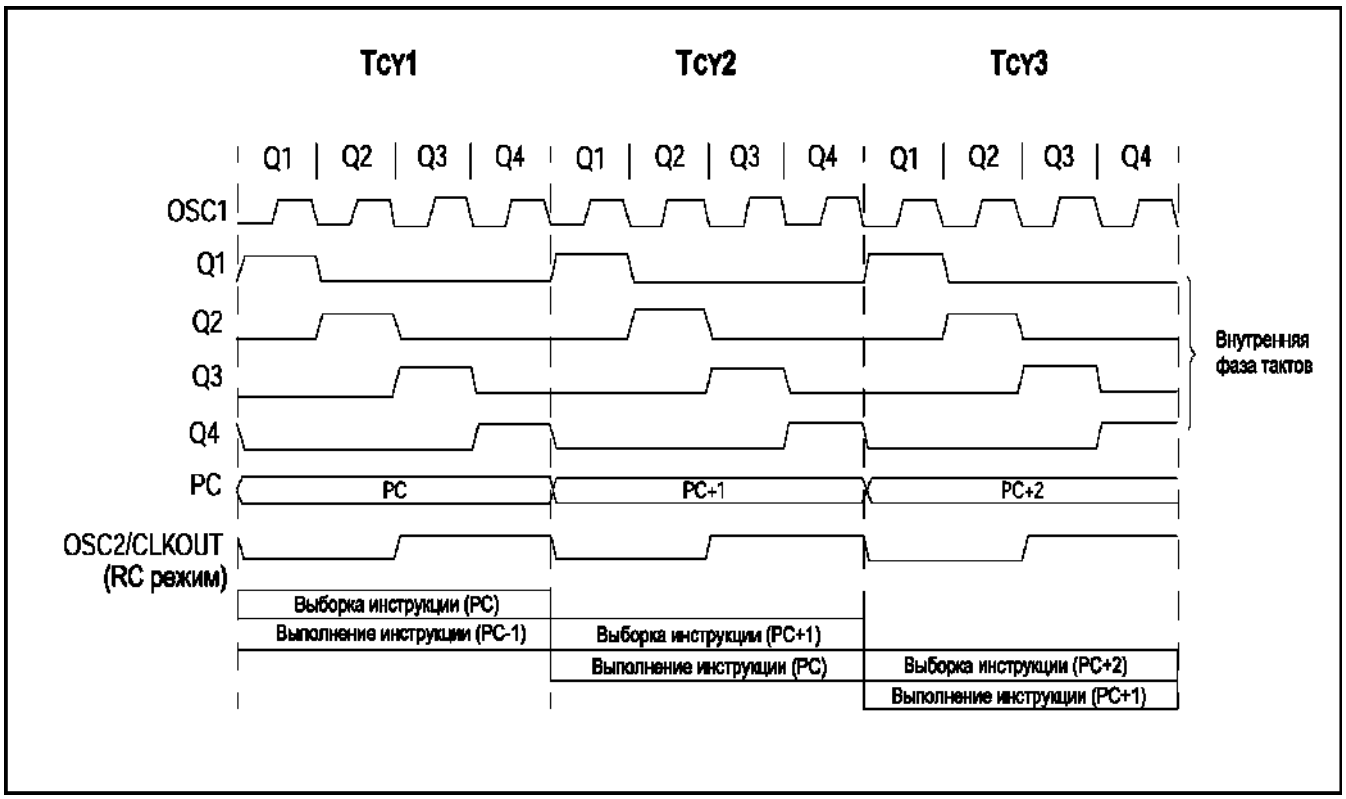

Рисунок 3.3 - Диаграмма циклов выполнения команд

Конвейерная выборка и выполнение команд. Цикл выполнения команды состоит из четырех тактов Q1, Q2, Q3 и Q4. Выборка следующей команды и выполнение текущей совмещены по времени, таким образом, выполнение команды происходит за один цикл. Если команда изменяет счетчик команд PC (команды ветвления, например GOTO), то необходимо два машинных цикла для выполнения команды.

Цикл выборки команды начинается с приращения счетчика команд PC в такте Q1. В цикле выполнения команды, код загруженной команды, помещается в регистр команд IR на такте Q1. Декодирование и выполнение команды происходит в тактах Q2, Q3 и Q4. Операнд из памяти данных читается в такте Q2, а результат выполнения команды записывается в такте Q4.

На рисунке примере 3.4 показаны две стадии конвейерной обработки команд для представленной последовательности. В цикле TCY0 происходит выборка первой команды из памяти программ. На цикле TCY1 первая команда исполняется, а вторая команда выбирается из памяти программ. В течение цикла TCY2 вторая команда исполняется, а третья выбирается из памяти программ. На цикле TCY3 происходит выборка четвертой команды и выполняется команда третья команда (CALL SUB_1). Когда завершается выполнение третьей команды CPU загружает адрес четвертой команды в вершину стека и изменяет счетчик команд PC на адрес SUB_1. Это означает, что команда, загруженная в цикле TCY3, должна быть удалена из конвейера. В течение цикла TCY4 четвертая команда удаляется из конвейера (выполняется пустой цикл NOP) и происходит выборка команды по адресу SUB_1. В цикле TCY5 выполняется команда пять и выбирается из памяти программ команда с адресом SUB_1 + 1.

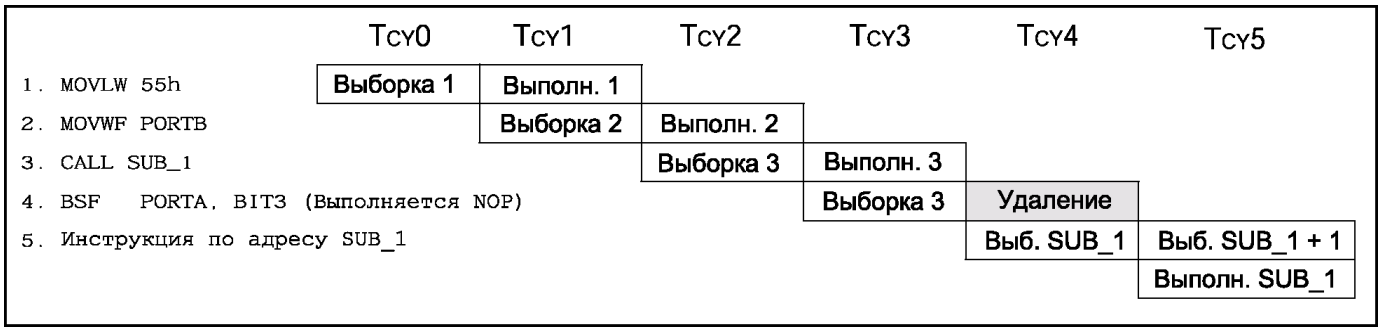

Рисунок 3.4 - Выборка и выполнения команд

Все команды выполняются за один цикл, кроме команд ветвления. Команды ветвления требуют два машинных цикла, т.к. необходимо удалить предварительно выбранную команду из конвейера. Во время удаления выбирается новая команда, а затем она исполняется в следующем машинном цикле.

Описание портов ввода/вывода.

В таблице 3.1 дано краткое описание функций, которые могут быть мультиплицированы к каналам портов ввода/вывода. Возможна ситуация, когда на один вывод мультиплицируется несколько функций. При использовании вывода периферийным модулем действие битов регистра TRIS может быть заблокировано (например для АЦП или LCD модуля).

Таблица 3.1 Описание выводов

Имя вывода	Тип вывода	Тип буфера	Описание
			Аналоговые входы.
AN0	I	Аналоговый	
AN1	I	Аналоговый	
AN2	I	Аналоговый	
AN3	I	Аналоговый	
AN4	I	Аналоговый	
AN5	I	Аналоговый	
AN6	I	Аналоговый	
AN7	I	Аналоговый	
AN8	I	Аналоговый	
AN9	I	Аналоговый	
AN10	I	Аналоговый	
AN11	I	Аналоговый	
AN12	I	Аналоговый	
AN13	I	Аналоговый	
AN14	I	Аналоговый	
AN15	I	Аналоговый	
A_{VDD}	P	P	Аналоговое питание.
A_{VSS}	P	P	Аналоговый общий.
C1	I	Аналоговый	Напряжение LCD.
C2	I	Аналоговый	Напряжение LCD.
CCP1	I/O	ST	Вывод Захват/Сравнение/ШИМ модуля CCP1.
CCP2	I/O	ST	Вывод Захват/Сравнение/ШИМ модуля CCP2.
CDAC	O	Аналоговый	Токовый вывод АЦП. Обычно используется для подключения внешнего конденсатора, чтобы формировать линейное уменьшение напряжения.
CK	I/O	ST	Тактовый сигнал USART в синхронном режиме. Всегда связан с функциями вывода TX (см. TX, RX, DT).
CLKIN	I	ST/CMOS	Вход внешнего тактового сигнала. Всегда связан с функциями вывода OSC1 (см. OSC1/CLKIN, OSC2/CLKOUT).
CLKOUT	O	-	Вывод тактового генератора. Подключается кварцевый/керамический резонатор в HS,XT, LP режиме генератора. В RC режиме генератора на выводе OSC2 присутствует сигнал CLKOUT с 1/4 частоты OSC1, равной циклам выполнения команд. Всегда связан с функциями вывода OSC2 (см. OSC1, OSC2).
CMPA	O	-	Выход компаратора A.
CMPB	O	-	Выход компаратора B.
COM0	L	-	Общий драйвер 0 LCD.
COM1	L	-	Общий драйвер 1 LCD.
COM2	L	-	Общий драйвер 2 LCD.
COM3	L	-	Общий драйвер 3 LCD.
-CS	I	TTL	Вход выбора микросхемы ведомого параллельного порта (см. -RD,-WR)
DT	I/O	ST	Сигнал данных USART в синхронном режиме. Всегда связан с функциями вывода RX (см. TX, RX, CK).
			GP - двунаправленный порт ввода/вывода. На некоторых входах могут быть программно включены внутренние подтягивающие резисторы.
GP0	I/O	TTL/ST	TTL буфер в режиме порта ввода/вывода. Буфер с триггером Шмидта режиме последовательного программирования.
GP1	I/O	TTL/ST	TTL буфер в режиме порта ввода/вывода. Буфер с триггером Шмидта режиме последовательного программирования.
GP2	I/O	ST	
GP3	I	TTL	
GP4	I/O	TTL	
GP5	I/O	TTL	

Таблица 3.1 Описание выводов (продолжение)

Имя вывода	Тип вывода	Тип буфера	Опи-са-
INT	I	ST	Вход внешних прерываний.
-MCLR/V_{PP}	I/P	ST	Вход сброса микроконтроллера (активный низкий логический уровень) или вход напряжения программирования.
NC	-	-	Эти выводы должны быть оставлены не подключенными.
OSC1	I	ST/CMOS	Вход тактового генератора или вход внешнего тактового сигнала. Входной буфер с триггером Шмидта в RC режиме генератора. КМОП буфер в остальных режимах.
OSC2	O	-	Вывод тактового генератора. Подключается кварцевый/керамический резонатор в HS,XT, LP режиме генератора. В RC режиме генератора на выводе OSC2 присутствует сигнал CLKOUT с 1/4 частоты OSC1, равной циклам выполнения команд.
PBTN	I	ST	Вход с внутренним подтягивающим резистором. Может использоваться для генерации прерываний.
			Ведомый параллельный порт для связи с микропроцессором. Выимеют входной буфер TTL, когда модуль PSP включен.
PSP0	I/O	TTL	
PSP1	I/O	TTL	
PSP2	I/O	TTL	
PSP3	I/O	TTL	
PSP4	I/O	TTL	
PSP5	I/O	TTL	
PSP6	I/O	TTL	
PSP7	I/O	TTL	
			PORTA - двунаправленный порт ввода/вывода.
RA0	I/O	TTL	
RA1	I/O	TTL	
RA2	I/O	TTL	
RA3	I/O	TTL	
RA4	I/O	ST	RA4 имеет выход с открытым стоком.
RA5	I/O	TTL	
			PORTB - двунаправленный порт ввода/вывода. На входах порта быть программно включены внутренние подтягивающие резисторы.
RB0	I/O	TTL	
RB1	I/O	TTL	
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	Прерывание по изменению уровня сигнала на входе.
RB5	I/O	TTL	Прерывание по изменению уровня сигнала на входе.
RB6	I/O	TTL/ST	Прерывание по изменению уровня сигнала на входе. Тактовый режиме программирования. Буфер ТТЛ в нормальном режиме. Бутриггером Шмидта в режиме программирования.
RB7	I/O	TTL/ST	Прерывание по изменению уровня сигнала на входе. Вывод данрежиме программирования. Буфер ТТЛ в нормальном режиме. Бутриггером Шмидта в режиме программирования.
			PORTC - двунаправленный порт ввода/вывода.
RC0	I/O	ST	
RC1	I/O	ST	
RC2	I/O	ST	
RC3	I/O	ST	
RC4	I/O	ST	
RC5	I/O	ST	
RC6	I/O	ST	
RC7	I/O	ST	
-RD	I	TTL	Управление чтением ведомого параллельного порта (см. -WR, -CS).

Таблица 3.1 Описание выводов (продолжение)

Имя вывода	Тип вывода	Тип буфера	Описание
			PORTD - двунаправленный порт ввода/вывода.
RD0	I/O	ST	
RD1	I/O	ST	
RD2	I/O	ST	
RD3	I/O	ST	
RD4	I/O	ST	
RD5	I/O	ST	
RD6	I/O	ST	
RD7	I/O	ST	
			PORTE - двунаправленный порт ввода/вывода.
RE0	I/O	ST	
RE1	I/O	ST	
RE2	I/O	ST	
RE3	I/O	ST	
RE4	I/O	ST	
RE5	I/O	ST	
RE6	I/O	ST	
RE7	I/O	ST	
REFA	O	CMOS	Выход A программируемого источника опорного напряже-
REFB	O	CMOS	ния. Выход B программируемого источника опорного на- пряжения.
			PORTF - цифровые входы или порт драйвера сегментов LCD.
RF0	I/O	ST	
RF1	I/O	ST	
RF2	I/O	ST	
RF3	I/O	ST	
RF4	I/O	ST	
RF5	I/O	ST	
RF6	I/O	ST	
RF7	I/O	ST	
			PORTG - цифровые входы или порт драйвера сегментов LCD.
RG0	I/O	ST	
RG1	I/O	ST	
RG2	I/O	ST	
RG3	I/O	ST	
RG4	I/O	ST	
RG5	I/O	ST	
RG6	I/O	ST	
RG7	I/O	ST	
RX	I	ST	Вывод приемника в асинхронном режиме USART.
SCL	I/O	ST	Вывод тактового сигнала в режиме I^2C.
SCLA	I/O	ST	Вывод тактового сигнала интерфейса I^2C.
SCLB	I/O	ST	Вывод тактового сигнала интерфейса I^2C.
SDA	I/O	ST	Вывод данных в режиме I^2C.
SDAA	I/O	ST	Вывод данных интерфейса I^2C.
SDAB	I/O	ST	Вывод данных интерфейса I^2C.
SCK	I/O	ST	Вход/выход тактового сигнала в режиме SPI.
SDI	I		Вывод приемника SPI.
SDO	O		Вывод передатчика SPI.
-SS	I		Вход выбора ведомого SPI.
SEG00:SEG31	I/L	ST	Драйверы сегментов LCD от 00 до 31.
SUM	O	AN	AN1 подтверждение перехода. К выводу может быть под- ключен внешний конденсатор
T0CKI	I	ST	Вход внешнего тактового сигнала для TMR0.
T1CKI	I	ST	Вход внешнего тактового сигнала для TMR1.
T1OSO	O	CMOS	Выход генератора TMR1.
T1OSI	I	CMOS	Вход генератора TMR1.
TX	O	-	Выход передатчика USART в асинхронном режиме (см. RX).

Таблица 3.1 Описание выводов (окончание)

Имя вывода	Тип вывода	Тип буфера	Описание
$V_{LCD}1$ $V_{LCD}2$ $V_{LCD}3$	P P P	- - -	Напряжение LCD. Напряжение LCD. Напряжение LCD.
VLCDADJ	I	Аналоговый	Напряжение LCD.
V_{REF}	I	Аналоговый	Вход верхнего опорного напряжения. Вывод входа опорного напряжения присутствует на микроконтроллерах с компараторами.
V_{REF+}	I	Аналоговый	Вход верхнего опорного напряжения. Обычно мультиплицируется на аналоговый вход.
V_{REF-}	I	Аналоговый	Вход нижнего опорного напряжения. Обычно мультиплицируется на аналоговый вход.
VREG	O	-	Вывод управления внешним компонентом N-FET для регулировки напряжения.
V_{SS}	P	-	Общий вывод для внутренней логики и портов ввода/вывода.
V_{DD}	P	-	Положительное напряжение питания для внутренней логики и портов ввода/вывода.
-WR			Управление записью в ведомый параллельный порт (см. -RD, -CS).

Обозначения:

TTL = входной буфер TTL;
ST = входной буфер с триггером Шмидта;
CMOS = КМОП совместимый вход или выход;
NPU = N-канальный подтягивающий элемент;
NO-PD = нет внутр. диода, подключ. к VDD;
PU = внутренний подтягивающий элемент;
I = вход;
O = выход;
L = драйвер LCD;
P = питание;
SM = соответствие спецификации SMBus. Требуется внешний подтягивающий резистор.

Контрольные вопросы.

1. Что такое порты ввода-вывода микроконтроллера, какова их функциональная роль?

2. Какие виды уровней сигналов допускают порты ввода-вывода? Как их можно настроить для решения конкретной задачи?

3. Что такое конвейерная обработка команд?

4. Каковы основные особенности архитектуры PIC-микроконтроллеров?

5. Какие части входят в общую структуру микроконтроллеров PIC среднего семейства и каково их назначение?

4 ОРГАНИЗАЦИЯ ПАМЯТИ

В данном разделе описывается два независимых блока памяти: память программ и память данных. Каждый блок имеет собственную шину данных и шину адреса, позволяя организовать одновременный доступ к обоим типам памяти в течение одного машинного цикла.

Память данных состоит из регистров общего (GPR) и специального (SFR) назначения. Регистры SFR, управляющие ядром микроконтроллера, будут описаны в данном разделе. Описание регистров SFR, управляющие периферийными модулями, смотрите в соответствующем разделе документации.

Организация памяти программ

Микроконтроллеры среднего семейства имеют 13-разрядный счетчик команд, способный адресовать 8К x 14 слов памяти программ, и 14-разрядную шину данных памяти программ. Все команды микроконтроллера состоят из 14-разрядного слова, поэтому микроконтроллер с объемом памяти программ 8К x 14 может содержать 8К команд. Это позволяет легко определить достаточность объема памяти программ для желаемого приложения.

Вся память программ разделена на 4 страницы по 2Кслов каждая (0000h-07FFh, 0800h–0FFFh, 1000h–17FFh, 1800h–1FFFh). На рисунке 4.1 показана карта памяти программ и 8-уровневый аппаратный стек. В зависимости от типа микроконтроллера только некоторая часть доступной памяти программ реализована аппаратно (смотрите техническую документацию на конкретный микроконтроллер).

Для перехода между страницами памяти программ необходимо изменить старшие биты регистра счетчика команд PC, записью в регистр специального назначения PCLATH (старший байт счетчика команд). Изменив значение регистра PCLATH и выполнив команду ветвления, счетчик команд PC пересечет границу страницы памяти программ без дополнительного вмешательства пользователя.

Для микроконтроллеров, имеющих память программ меньше 8Кслов, обращение к памяти программ выше фактически реализованного значения приведет к циклической адресации. Например, в микроконтроллере с памятью программ

4Кслов и попытке перехода по адресу 17FFh переход будет выполнен по адресу 07FFh. В микроконтроллерах с памятью программ 2Кслов управление страницами памяти не требуется.

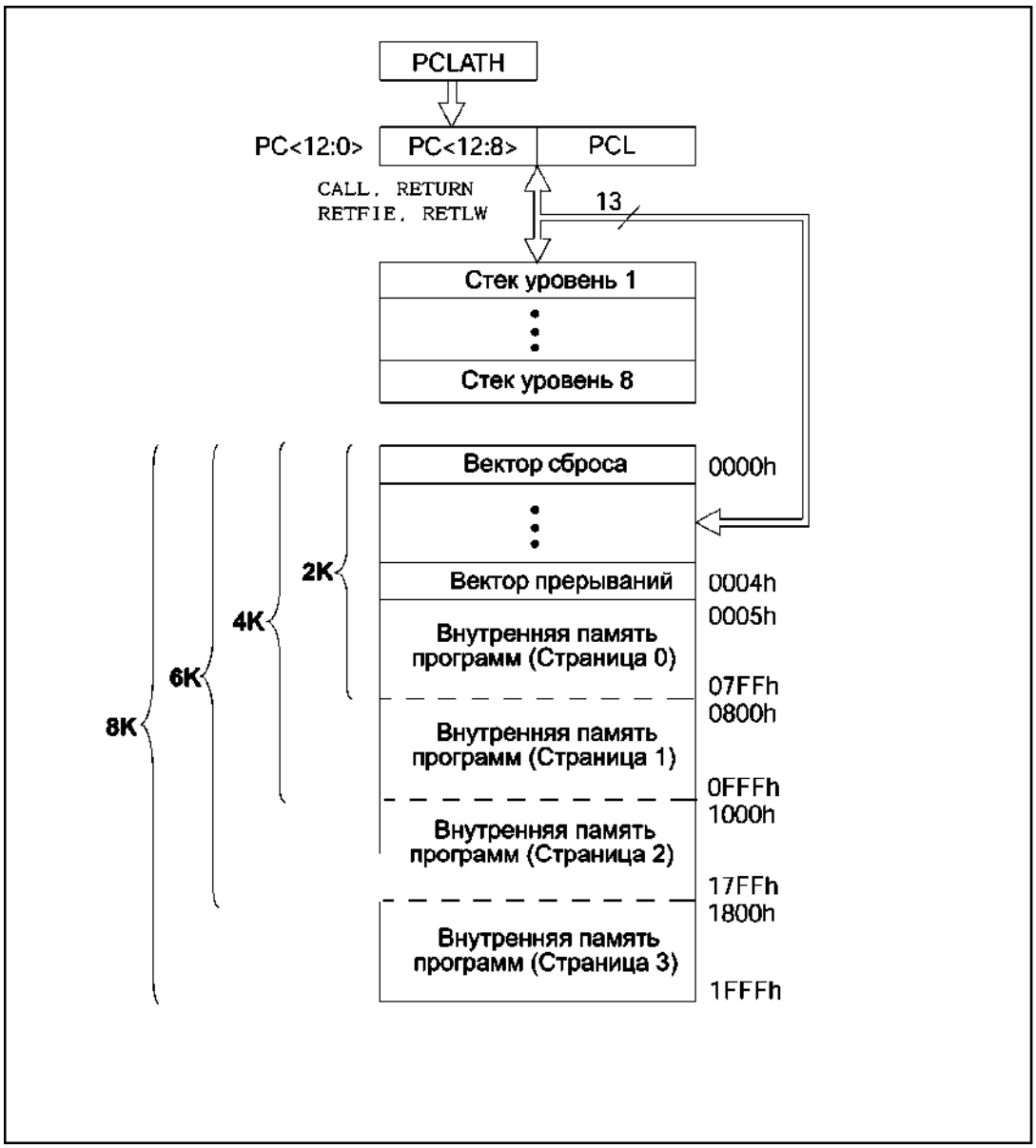

Рисунок 4.1 - Карта памяти программ и 8-уровневый аппаратный стек

Примечания:

1. Не во всех микроконтроллерах полностью реализовано адресное пространство памяти программ.

2. В памяти программ может размещаться калибровочная информация.

Вектор сброса. В любом микроконтроллере PICmicro сброс приведет к очистке счетчика команд (PC), устанавливая адрес 0h. Адрес 0000h называется «адрес вектора сброса», т.к. будет выполнен переход по этому адресу при сбросе микро-

контроллера. Вместе со счетчиком команд (PC) очищается регистр PCLATH, устанавливая рабочую страницу памяти программ 0.

Вектор прерываний. Когда возникает разрешенное прерывание, в счетчик команд PC записывается адрес 0004h, называемый «адрес вектора прерываний», при этом значение регистра PCLATH не изменяется. Если в подпрограмме обработки прерываний требуется выполнять команды ветвления, то необходимо предварительно записать в регистр PCLATH значение, определяющее нужную страницу памяти программ. Прежде чем регистр PCLATH будет изменен, его значение должно быть сохранено в другом регистре памяти данных, а затем восстановлено перед выходом из подпрограммы обработки прерываний.

Калибровочная информация. В некоторых типах микроконтроллеров, во время заключительного заводского испытания в памяти программ сохраняется калибровочная информация. Использование калибровочной информации позволяет приложению получать наилучшие результаты работы. Как правило, калибровочная информация сохраняется в конце памяти программ набором инструкций RETLW.

Примечание. Для микроконтроллеров, в которых перед программированием очищается вся память программ, рекомендуется сначала прочитать калибровочную информацию, а затем запрограммировать микроконтроллер.

Счетчик команд PC. 13-разрядный регистр счетчика команд PC указывает адрес выбираемой команды для выполнения. Младший байт счетчика программ PCL доступен для чтения и записи. Старший байт PCH, содержащий <12:8> биты счетчика команд PC, не доступен для чтения и записи. Все операции с регистром PCH происходят через дополнительный регистр PCLATH. На рисунках 4.2 показано 4 примера изменения значения счетчика команд PC.

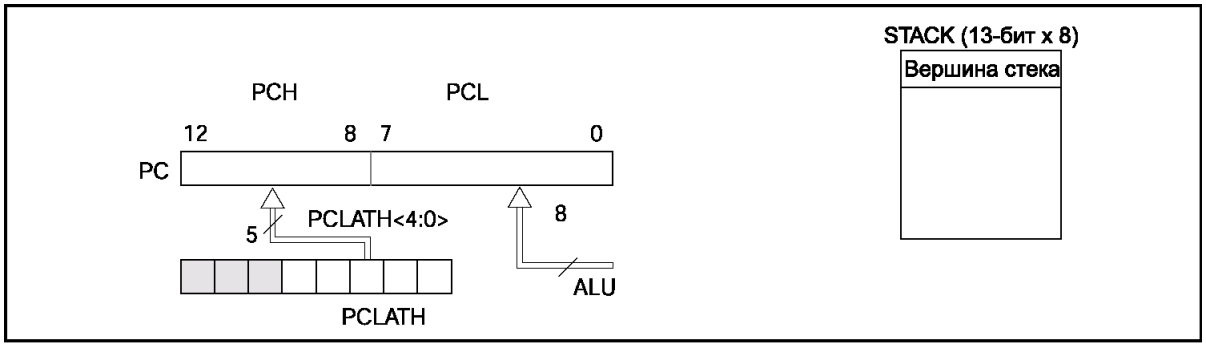

Рисунок 4.2 а) - Непосредственная запись значения в регистр PCL (PCLATH<4:0> →PCH)

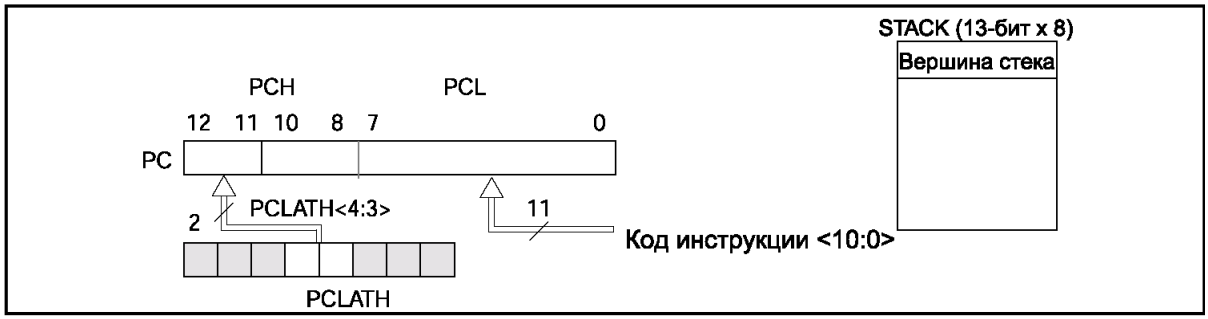

Рисунок 4.2 б) - Изменение значения PC при выполнении инструкции GOTO (PCLATH<4:3> → PCH)

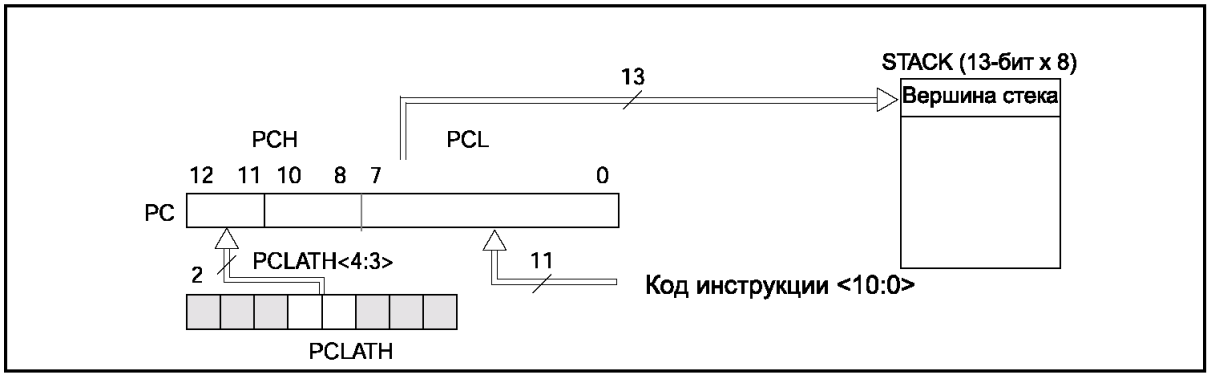

Рисунок 4.2 в) - Изменение значения PC при выполнении перехода к подпрограмме CALL (PCLATH<4:3> → PCH), при этом старое значение PC сохраняется в вершине стека

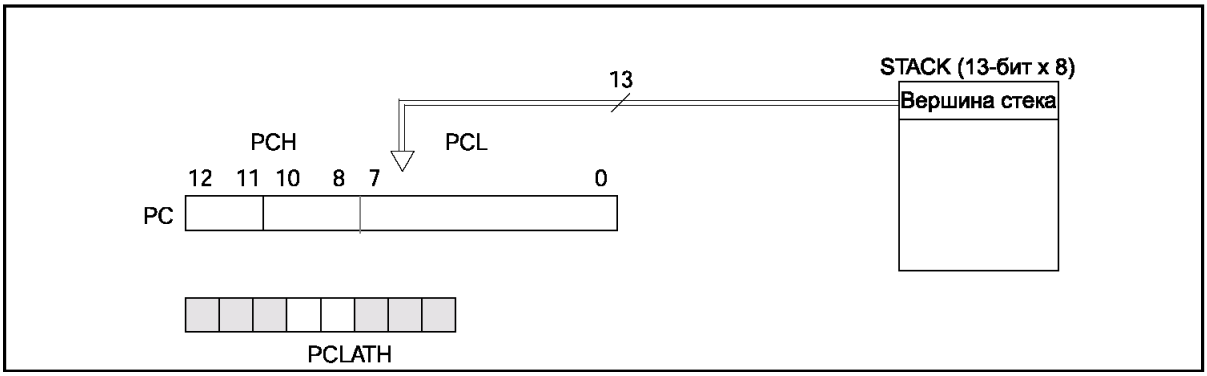

Рисунок 4.2 г) - Возвращение из подпрограммы (RETURN, RETFIE или RETLW), счетчик команд загружается значением с вершины стека

Примечание. Регистр PCLATH не изменяется при изменении PCH.

Вычисляемый переход. Вычисляемый переход может быть выполнен командой приращения к регистру PCL (например, ADDWF PCL). При выполнении вычисляемого перехода следует заботиться о том, чтобы значение PCL не пересекло границу блока памяти (каждый блок 256 байт).

Примечание. При записи значения в регистр PCL, автоматически происходит перезапись 5 младших бит из регистра PCLATH<4:0> в регистр PCH.

Аппаратный стек. Стек поддерживает до 8 уровней вложенности подпрограмм пользователя, включая обработку прерываний. В стеке сохраняется адрес возврата в основную программу.

В микроконтроллерах среднего семейства PICmicro реализован 8-уровневый 13-разрядный аппаратный стек. Стек не имеет отображения на память программ и память данных, нельзя записать или прочитать данные из стека. Значение счетчика команд заносится в вершину стека при выполнении инструкций перехода на подпрограмму (CALL) или обработку прерываний. Чтение из стека и запись в счетчик команд PC происходит при выполнении инструкций возврата из подпрограммы или обработки прерываний (RETURN, RETLW, RETFIE), при этом значение регистра PCLATH не изменяется. После 8 записей в стек, девятая запись запишется на место первой, а десятая запись заменит вторую и т. д. Пример показан на рисунке 4.3.

Рисунок 4.3 - Пример последовательности записи в стек

Примечание 1. В микроконтроллерах не имеется никаких указателей о переполнении стека.

Примечание 2. В микроконтроллерах не предусмотрено команд записи/чтения из стека, кроме команд вызова/возврата из подпрограммы (CALL, RETURN, RETLW и RETFIE) или условий перехода по вектору прерываний.

Страницы памяти программ. Команды переходов (CALL, GOTO) в микроконтроллерах среднего семейства PICmicro имеют 11-разрядное поле для указания адреса, что позволяет непосредственно адресовать 2Кслов памяти программ. Некоторые микроконтроллеры имеют память программ более 2Кслов. Для адресации

верхних страниц памяти программ используются 2 бита в регистре PCLATH<4:3>. Перед выполнением команды перехода (CALL или GOTO) необходимо запрограммировать биты регистра PCLATH<4:3> для адресации требуемой страницы (см. рисунки 4.2-б, 4.2-в).

При выполнении инструкций возврата из подпрограммы 13-разрядное значение для счетчика программ PC берется с вершины стека, поэтому манипуляция битами регистра PCLATH<3:4> не требуется (см. рисунок 4.2-г).

Примечание. В микроконтроллерах с объемом памяти программ до 2Кслов биты регистра PLATH<4:3> игнорируются. Не рекомендуется их использовать как биты общего назначения, т.к. может возникнуть необходимость переноса программы на микроконтроллер с большим объемом памяти программ.

В микроконтроллерах с объемом памяти программ до 4Кслов бит регистра PLATH<4> игнорируется. Не рекомендуется его использовать как бит общего назначения, т.к. может возникнуть необходимость переноса программы на микроконтроллер с большим объемом памяти программ.

В примере 4.1 показан переход со страницы 0 на страницу 1 памяти программ. Этот пример предполагает, что в подпрограмме сохраняется и восстанавливается значение регистра PCLATH.

Пример 4.1. Выполнение перехода со страницы 0 на страницу 1 памяти программ.

```
                ORG     0x500
                BSF     PCLATH,3        ; Выбор страницы 1 (800h-FFFh)
                CALL    SUB1_P1         ; Переход на страницу 1 (800h-FFFh)
                :
                :

                ORG     0x900
SUB1_P1:                                ; Страница 1 (800h-FFFh)
                :
                RETURN                  ; Возврат на страницу 0 (000h-7FFh)
```

Организация памяти данных. Память данных разделяется на регистры двух типов:

- Регистры специального назначения (SFR), управляют работой микроконтроллера;
- Регистры общего назначения (GPR), для хранения данных программы.

Память данных разделена на банки, содержащие регистры общего и специального назначения. Регистры общего назначения размещаются в разных банках памяти данных для того, чтобы была возможность организовать более 96 байт ОЗУ. Регистры специального назначения предназначены для управления периферийными модулями и функциями микроконтроллера. Управление банками памяти выполняется битами в регистре STATUS<7:5>. На рисунке 4.5 представлена одна из разновидностей карты памяти данных. Организация памяти данных зависит от типа микроконтроллера.

Чтобы передать данные из одного регистра в другой, необходимо использовать дополнительный регистр W. Эта операция выполняется двумя командами за два машинных цикла микроконтроллера.

Обращение к всем регистрам памяти данных может быть выполнено прямой или косвенной адресацией:

• Прямая адресация - для указания банка памяти данных необходимо использовать биты PR1:PR0 регистра STATUS;

• Косвенная адресация - адрес регистра сохраняется в FSR, а в бите IRP регистра STATUS указывается к какой паре банков памяти данных выполняется обращение (Банк0/Банк1 или Банк2/Банк3).

Регистры общего назначения (GRP). Регистры общего назначения размещаются в разных банках памяти данных. Эти регистры не инициализируются при сбросе по включению питания и имеют неизвестное значение, а при всех остальных сбросах микроконтроллера не изменяют своего значения.

Обращение к регистрам общего назначения может быть выполнено прямой или косвенной адресацией (через регистры FSR и INDF). В некоторых микроконтроллерах существуют регистры общего назначения, адресуемые к одной и той же ячейке ОЗУ, независимо от текущего банка памяти данных. Обратите внимание на эти регистры, т.к. они расположены в общем ОЗУ.

Регистры специального назначения (SFR). Регистры специального назначения используются для управления ядром и периферийными модулями микроконтроллера. Эти регистры реализованы как статическое ОЗУ. Описание регистров

SFR, управляющих периферийными модулями, смотрите в соответствующем разделе документации.

Регистры специального назначения размещены в различных банках памяти данных, а некоторые из регистров отображаются во всех банках. Переключение рабочего банка памяти выполняется настройкой битов RP1:RP0 регистра STATUS. При сбросе по включению питания и других видах сброса микроконтроллера в некоторые регистры специального назначения записывается определенное значение. Существуют регистры SFR, которые содержат неизвестное значение при сбросе по включению питания, а при других видах сброса не изменяются (см. техническую документацию на соответствующий микроконтроллер). Обращение к регистрам специального назначения может быть выполнено прямой или косвенной адресацией.

Примечание. В области регистров специального назначения могут размещаться регистры общего назначения.

Банки памяти данных. Память данных разделена на 4 банка. Каждый банк содержит регистры специального назначения (в начале адресного пространства банка) и регистры общего назначения. Переключение между банками осуществляется с помощью битов:

- RP1:RP0 в регистре STATUS при прямой адресации;
- IRP в регистре STATUS при косвенной адресации.

Таблица 4.1 - Выбор банка памяти данных при прямой и косвенной адресации

Доступ к банку	Прямая адресация (RP1:RP0)	Косвенная адресация (IRP)
0	0 0	0
1	0 1	
2	1 0	1
3	1 1	

Вся память данных выполнена по технологии статического ОЗУ с максимальным размером банка памяти 7Fh (128 байт). В начале банка располагаются регистры специального назначения, за ними регистры общего назначения. Некоторые, часто используемые регистры специального назначения банка 0 отображаются в других банках памяти для упрощения программы микроконтроллера и получения быстрого доступа к ним.

В процессе разработки новых микроконтроллеров, адреса размещения регистров специального назначения в памяти данных претерпели некоторые изменения. Организация памяти данных, показанная на рисунке 4.5, является стандартом для всех новых микроконтроллеров PIC среднего семейства. Последние 16 байт всех банков памяти данных, показанных на карте, отображаются на банк 0, что должно упростить программу, работающую с банками памяти данных. Регистры, имена которых выделены полужирным текстом, присутствуют во всех микроконтроллерах.

Состав регистров специального назначения, их адреса в памяти данных и объем регистров общего назначения для конкретного типа микроконтроллера смотрите в технической документации на соответствующий микроконтроллер.

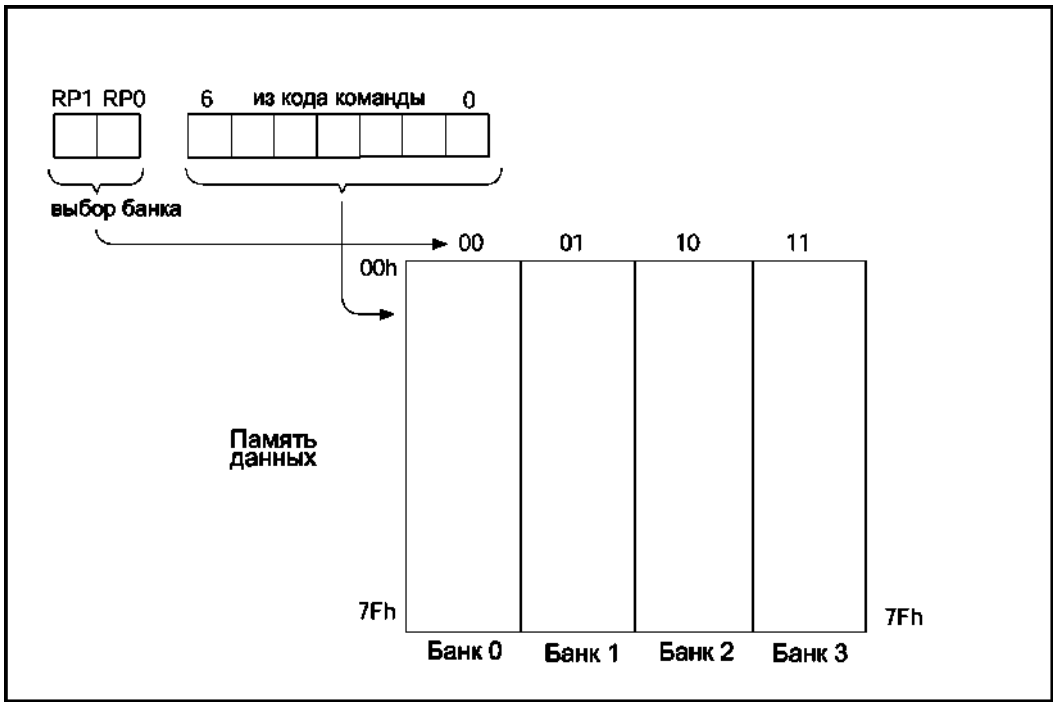

Рисунок 4.4 - Механизм прямой адресации памяти данных

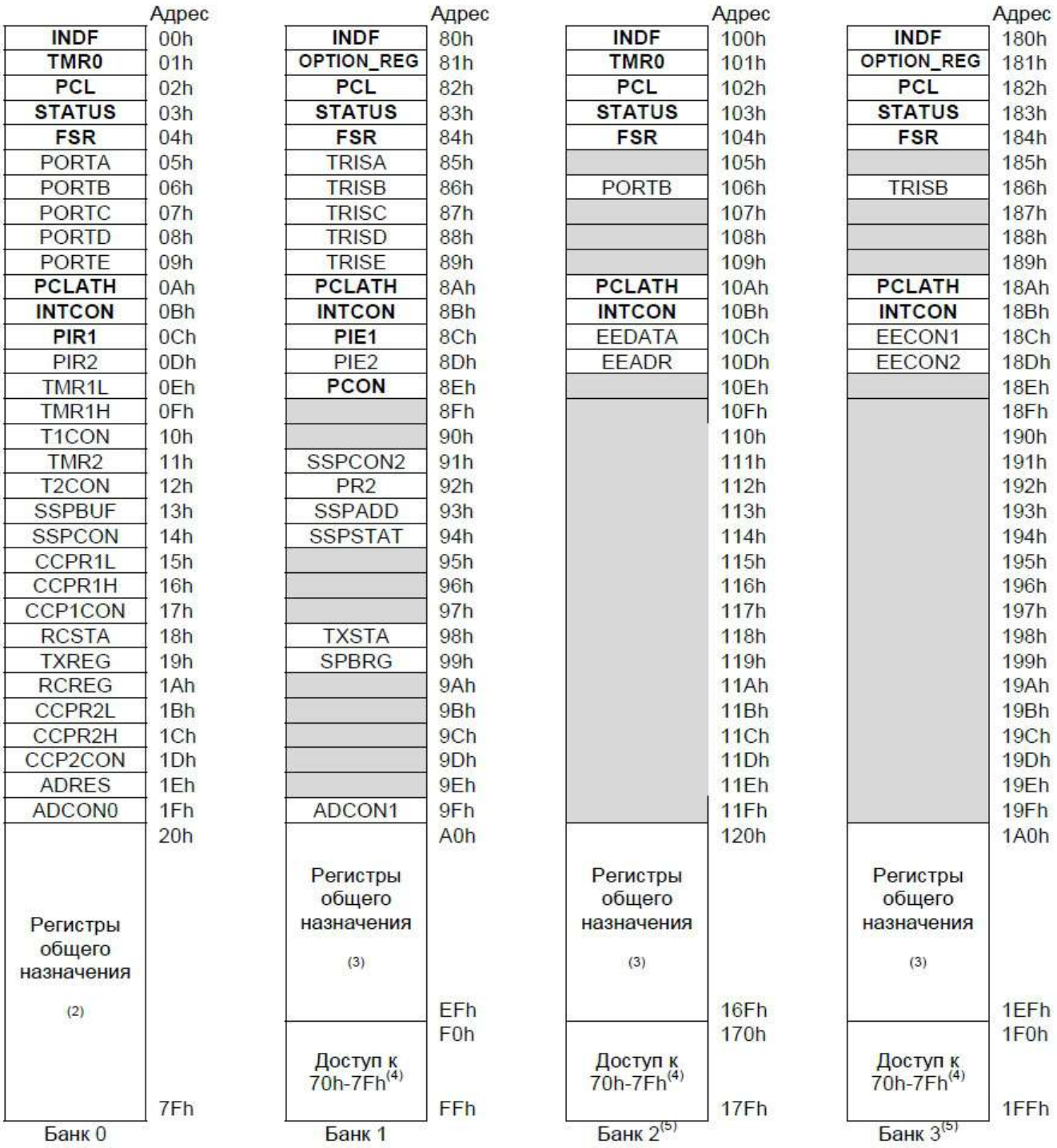

Рисунок 4.5 - Карта памяти данных

Примечания:

1. Регистры, названия которых выделены жирным текстом, будут присутствовать в каждом микроконтроллере.

2. В некоторых микроконтроллерах часть банка памяти данных может быть не реализована. Чтение по адресу несуществующего ОЗУ будет давать результат 00h.

3. Эта часть ОЗУ может быть не реализована. Чтение по адресу несуществующего ОЗУ будет давать непредсказуемый результат.

4. В некоторых микроконтроллерах эти регистры общего назначения отображается на банк 0.

5. Банк может быть не реализован. Чтение по адресу несуществующего ОЗУ будет давать результат 00h.

6. Регистры общего назначения могут быть расположены в области регистров специального назначения.

На рисунке 4.6 показана карта памяти данных некоторых 18-выводных микроконтроллеров. Чтение не реализованных регистров будет давать результат '0'.

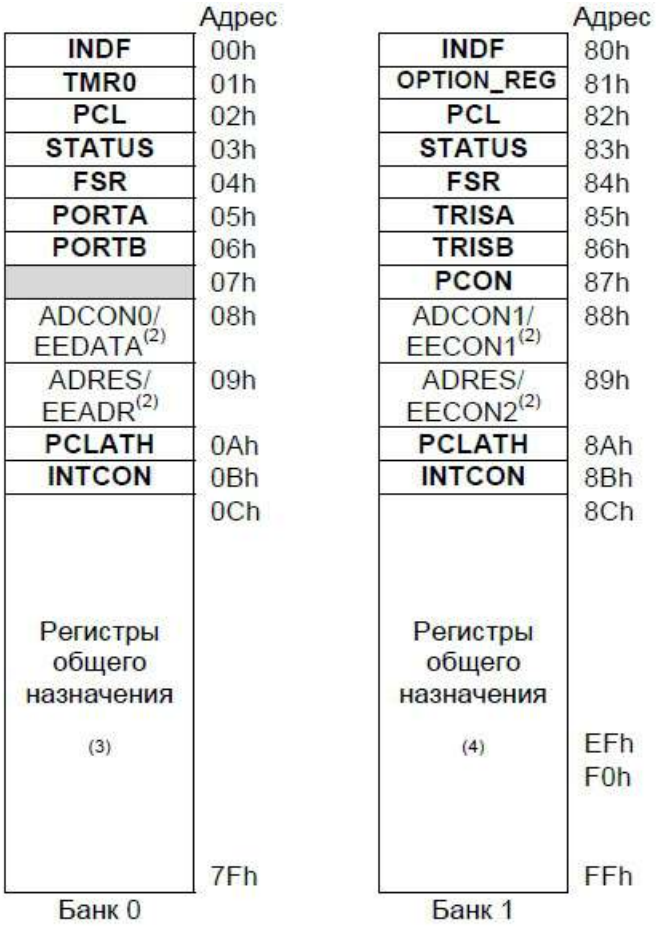

Рисунок 4.6 - Карта памяти данных некоторых 18 - выводных микроконтроллеров

Косвенная адресация, регистры INDF и FSR. Косвенная адресация – такой режим адресации регистров, при котором в команде не указывается адрес памяти данных. Регистр специального назначения FSR, доступный для записи/чтения, используется в качестве указателя адреса в памяти данных. Этот режим адресации может быть полезен при обращении к таблицам данных.

На рисунке 4.7 показана операция косвенной адресации (запись значения в регистр памяти данных с адресом, указанном в регистре FSR). Для выполнения косвенной адресации необходимо обратиться к регистру INDF. Обращение к регистру INDF фактически вызовет действие с регистром, адрес которого указан в FSR. Кос-

венное чтение регистра INDF (FSR=0) даст результат 00h. Косвенная запись в регистр INDF не вызовет никаких действий (вызывает воздействия на флаги АЛУ в регистре STATUS). 9-бит косвенного адреса IRP сохраняется в регистре STATUS<7>. Пример 9-разрядной косвенной адресации показан на рисунке 4.8.

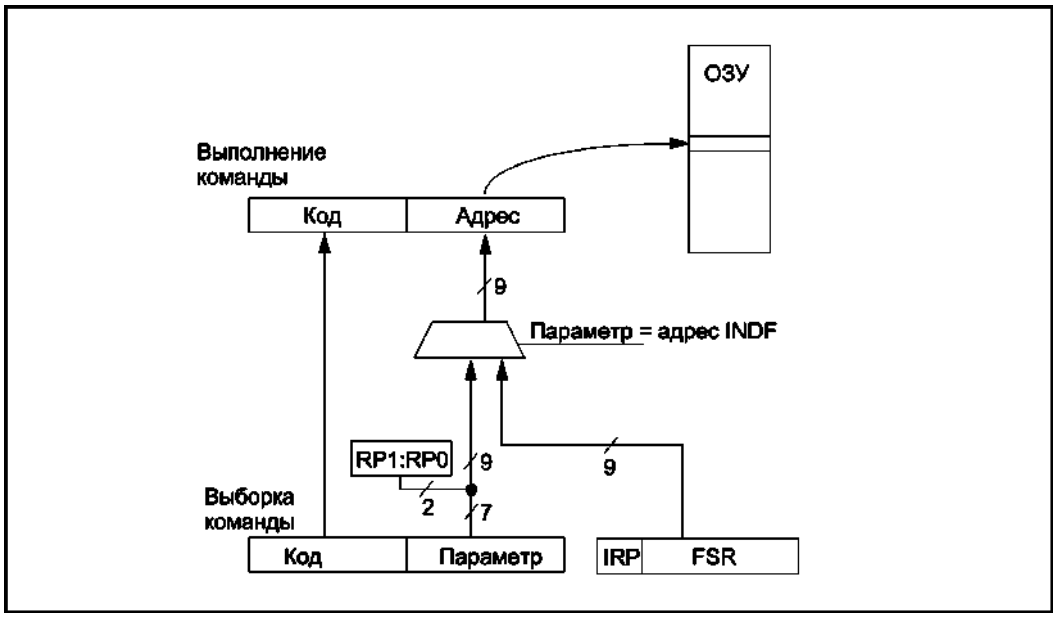

Рисунок 4.7 - Косвенная адресация

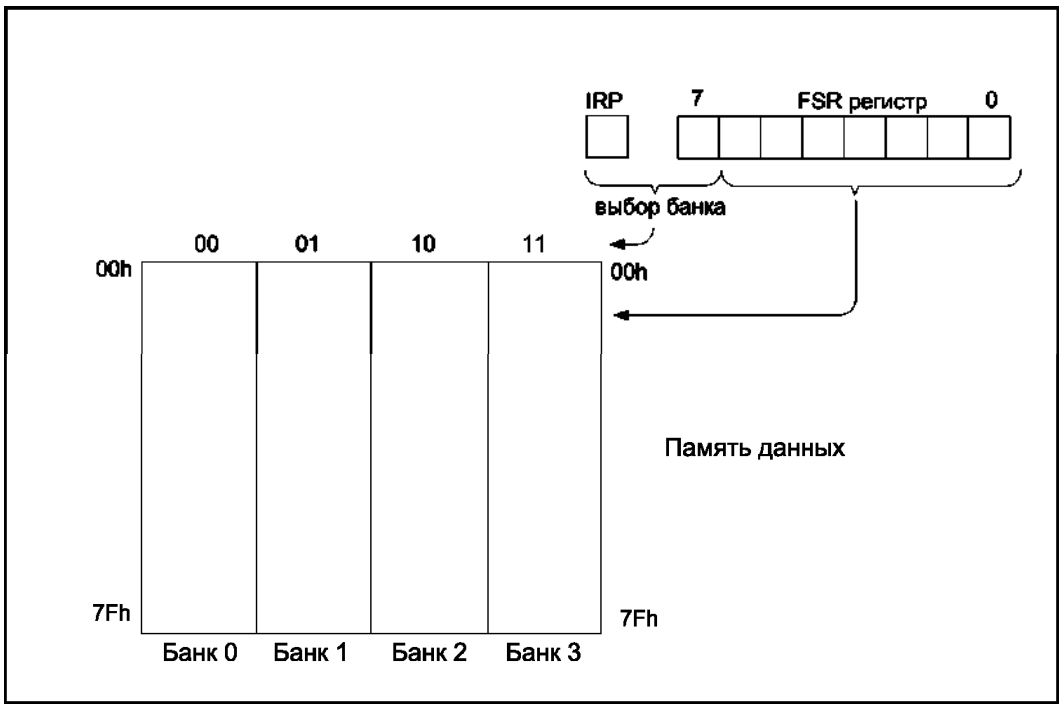

Рисунок 4.8 - Механизм косвенной адресации

В примере 4.2 показано использование косвенной адресации для очистки памяти данных (адреса 20h – 2Fh) минимальным числом команд микроконтроллера. Подобный метод может использоваться для передачи блока данных в регистр

TXREG передатчика USART. Начальный адрес блока данных, подготовленного для передачи, может быть легко изменен в соответствии с требованиями программы.

Пример 4.2. Очистка регистров памяти данных с адреса 20h по 2Fh

```
        BCF         STATUS, IRP       ; Установить банк 0,1
        MOVLW       0x20              ; Указать первый регистр в ОЗУ
        MOVWF       FSR
NEXT:
        CLRF        INDF              ; Очистить регистр
        INCF        FSR,F             ; Увеличить адрес
        BTFSS       FSR,4             ; Завершить?
        GOTO        NEXT              ; Нет, продолжить очистку
CONTINUE:                             ; Да
```

Инициализация. В примере 4.3 показано переключение между банками памяти данных для прямой адресации, а в примере 4.4 представлен код программы, выполняющей инициализацию (очистку) регистров общего назначения.

Пример 4.3 Переключение банков памяти данных

```
        CLRF    STATUS        ; Очистка регистра STATUS (Банк 0)
        :
        BSF     STATUS, RP0   ; Банк 1
        :
        BCF     STATUS, RP0   ; Банк 0
        :
        MOVLW 0x60            ; Установить RP0 и RP1 в STATUS регистре
        XORWF STATUS, F       ; (Банк 3)
        :
        BCF     STATUS, RP0   ; Банк 2
        :
        BCF     STATUS, RP1   ; Банк 0
```

Пример 4.4 Инициализация регистров общего назначения

```
        CLRF    STATUS        ; Очистить регистр STATUS (Банк 0)
        MOVLW   0x20          ; 1-й адрес регистра GPR
        MOVWF   FSR           ; записать в регистр косвенного адреса
Bank0_LP
        CLRF    INDF0         ; Очистить регистр GPR с адресом в регистре FSR
        INCF    FSR           ; Следующий регистр GPR
        BTFSS   FSR, 7        ; Очистка регистров GPR в этом банке завершена?
                              ; (FSR = 80h, C = 0)
        GOTO    Bank0_LP      ; НЕТ, продолжать очистку
                              ;
                              ; Следующий банк (Банк 1)
                              ; (** Только для микроконтроллеров с банком 1 **)
                              ;
        MOVLW   0xA0          ; 1-й адрес регистра GPR
        MOVWF   FSR           ; записать в регистр косвенного адреса
Bank1_LP
        CLRF    INDF0         ; Очистить регистр GPR с адресом в регистре FSR
        INCF    FSR           ; Следующий регистр GPR
        BTFSS   STATUS, C     ; Очистка регистров GPR в этом банке завершена?
                              ; (FSR = 00h, C = 1)
        GOTO    Bank1_LP      ; НЕТ, продолжать очистку
                              ;
                              ; Следующий банк (Банк 2)
                              ; (** Только для микроконтроллеров с банком 2 **)
                              ;
        BSF     STATUS, IRP   ; Выбор банков 2 и 3  для косвенной адресации
        MOVLW   0x20          ; 1-й адрес регистра GPR
```

```
            MOVWF  FSR          ; записать в регистр косвенного адреса
Bank2_LP
            CLRF   INDF0        ; Очистить регистр GPR с адресом в регистре FSR
            INCF   FSR          ; Следующий регистр GPR
            BTFSS  FSR, 7       ; Очистка регистров GPR в этом банке завершена?
                                ; (FSR = 80h, C = 0)
            GOTO   Bank2_LP     ; НЕТ, продолжать очистку
                                ;
                                ; Следующий банк (Банк 3)
                                ; (** Только для микроконтроллеров с банком 3 **)
                                ;
            MOVLW  0xA0         ; 1-й адрес регистра GPR
            MOVWF  FSR          ; записать в регистр косвенного адреса
Bank3_LP
            CLRF   INDF0        ; Очистить регистр GPR с адресом в регистре FSR
            INCF   FSR          ; Следующий регистр GPR
            BTFSS  STATUS, C    ; Очистка регистров GPR в этом банке завершена?
                                ; (FSR = 00h, C = 1)
            GOTO   Bank3_LP     ; НЕТ, продолжать очистку
            :                   ; ДА, все регистры GPR очищены
```

Контрольные вопросы.

1. В чем отличие памяти программ микроконтроллера от памяти данных?

2. Что представляет собой карта памяти программы микроконтроллера?

3. Что такое аппаратный стек и вычисляемый переход?

4. В чем отличие регистров специального назначения от регистров общего назначения?

5. Для чего применяется калибровочная информация микроконтроллера, можно ли ее изменить?

6. Что такое банк памяти, каково его назначение?

7. Что такое косвенная адресация, для чего она применяется?

5. СИСТЕМА КОМАНД

Каждая команда состоит из одного 14 - разрядного слова, разделенного на код операции (OPCODE), определяющий тип команды и один или несколько операндов, определяющие операцию команды. Полный список команд показан в таблице 5.1.

Система команд аккумуляторного типа, ортогональна и разделена на три основных группы:

• Байт ориентированные команды;

• Бит ориентированные команды;

• Команды управления и операций с константами.

Для байт ориентированных команд 'f' является указателем регистра, а 'd' указателем адресата результата. Указатель регистра определяет, какой регистр должен использоваться в команде. Указатель адресата определяет, где будет сохранен результат. Если 'd'=0, результат сохраняется в регистре W. Если 'd'=1, результат сохраняется в регистре, который используется в команде.

В бит ориентированных командах 'b' определяет номер бита участвующего в операции, а 'f' - указатель регистра, который содержит этот бит.

В командах управления или операциях с константами 'k' представляет восемь или одиннадцать бит константы или значения литералов.

Все команды выполняются за один машинный цикл, кроме команд условия, в которых получен истинный результат и инструкций изменяющих значение счетчика команд PC. В случае выполнения команды за два машинных цикла, во втором цикле выполняется инструкция NOP. Один машинный цикл состоит из четырех тактов генератора. Для тактового генератора с частотой 4 МГц все команды выполняются за 1мкс, если условие истинно или изменяется счетчик команд PC, команда выполняется за 2мкс.

Таблица 5.1 - Список команд микроконтроллеров среднего семейства

Мнемоника команды		Описание	Циклов	14-разрядный код		Изм. флаги	Прим.
				Бит 13	Бит 0		
Байт ориентированные ко-							
ADDWF	**f,d**	Сложение W и f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	**f,d**	Побитное 'И' W и f	1	00 0101 dfff ffff		Z	1,2
CLRF	**f**	Очистить f	1	00 0001 1fff ffff		Z	2
CLRW	**-**	Очистить W	1	00 0001 0xxx xxxx		Z	
COMF	**f,d**	Инвертировать f	1	00 1001 dfff ffff		Z	1,2
DECF	**f,d**	Вычесть 1 из f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	**f,d**	Вычесть 1 из f и пропустить если 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	**f,d**	Прибавить 1 к f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	**f,d**	Прибавить 1 к f и пропустить если 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	**f,d**	Побитное 'ИЛИ' W и f	1	00 0100 dfff ffff		Z	1,2
MOVF	**f,d**	Переслать f	1	00 1000 dfff ffff		Z	1,2
MOVWF	**f**	Переслать W в f	1	00 0000 1fff ffff			
NOP	**-**	Нет операции	1	00 0000 0xx0 0000			
RLF	**f,d**	Циклический сдвиг f влево через перенос	1	00 1101 dfff ffff		C	1,2
RRF	**f,d**	Циклический сдвиг f вправо через перенос	1	00 1100 dfff ffff		C	1,2
SUBWF	**f,d**	Вычесть W из f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	**f,d**	Поменять местами полубайты в регистре f	1	00 1110 dfff ffff			1,2
XORWF	**f,d**	Побитное 'исключающее ИЛИ' W и f	1	00 0110 dfff ffff		Z	1,2
Бит ориентированные ко-							
BCF	**f,b**	Очистить бит b в регистре f	1	01 00bb bfff ffff			1,2
BSF	**f,b**	Установить бит b в регистре f	1	01 01bb bfff ffff			1,2
BTFSC	**f,b**	Проверить бит b в регистре f, пропустить если 0	1(2)	01 10bb bfff ffff			3
BTFSS	**f,b**	Проверить бит b в регистре f, пропустить если 1	1(2)	01 11bb bfff ffff			3
Команды управления и операций с константами							
ADDLW	**k**	Сложить константу с W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	**k**	Побитное 'И' константы и W	1	11 1001 kkkk kkkk		Z	
CALL	**k**	Вызов подпрограммы	2	10 0kkk kkkk kkkk			
CLRWDT	**-**	Очистить WDT	1	00 0000 0110 0100		-TO,-PD	
GOTO	**k**	Безусловный переход	2	10 1kkk kkkk kkkk			
IORLW	**k**	Побитное 'ИЛИ' константы и W	1	11 1000 kkkk kkkk		Z	
MOVLW	**k**	Переслать константу в W	1	11 00xx kkkk kkkk			
RETFIE	**-**	Возврат из подпрограммы с разрешением прерываний	2	00 0000 0000 1001			
RETLW	**k**	Возврат из подпрограммы с загрузкой константы в W	2	11 01xx kkkk kkkk			
RETURN	**-**	Возврат из подпрограммы	2	00 0000 0000 1000			
SLEEP	**-**	Перейти в режим SLEEP	1	00 0000 0110 0011		-TO,-PD	
SUBLW	**k**	Вычесть W из константы	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	**k**	Побитное 'исключающее ИЛИ' константы и W	1	11 1010 kkkk kkkk		Z	

Примечания:

1. При выполнении операции "чтение - модификация - запись" с портом ввода/вывода (например MOVF PORTB,1) исходные значения считываются с выводов порта, а не из выходных защелок. Например, если в выходной защелке было записана '1', а на соответствующем выходе низкий уровень сигнала, то обратно будет записано значение '0'.

2. При выполнении записи в TMR0 (и d=1) предделитель TMR0 сбрасывается, если он подключен к модулю TMR0.

3. Если условие истинно или изменяется значение счетчика команд PC, то инструкция выполняется за два цикла. Во втором цикле выполняется команда NOP.

Формат команд. На рисунке 5.1 показан формат трех групп команд. Код команды может быть от 3 до 6 бит, что позволяет реализовать 35 команд.

Примечание 1. Любой не реализованный код операции сохранен для последующих разработок. Использование недокументированного кода операции может привести к непредсказуемым последствиям.

Во всех примерах используется следующий формат шестнадцатеричных чисел: 0xhh, где h - шестнадцатеричная цифра.

Представление двоичного числа: 00000100b, где b - указатель двоичного числа.

Рисунок 5.1 - Общий формат команд микроконтроллеров среднего семейства

Таблица 5.2 Описание полей кода операции

Поле	Описание
f	Адрес регистра (от 0x00 до 0x7F)
w	Рабочий регистр (аккумулятор)
b	Номер бита в 8-разрядном регистре
k	Константа (данные или метка)
x	Не имеет значения (0 или 1). Ассемблер генерирует x=0 для совместимости программы микроконтроллера с инструментальными средствами
d	Указатель адресата результата операции: d = 0 - результат сохраняется в регистре w d = 1 - результат сохраняется в регистре f По умолчанию d = 1
label	Имя метки

TOS	Вершина стека
PC	Счетчик команд
PCLATH	Буфер старшего байта счетчика команд
GIE	Бит глобального разрешения прерываний
WDT	Сторожевой таймер
-TO	Флаг переполнения WDT
-PD	Флаг сброса по включению питания
dest	Приемник, регистр w или регистр памяти
[ ]	Дополнительные параметры
( )	Содержимое
□	Присвоение
< >	Битовое поле
□	Из набора
Курсив	Термин, определяемый пользователем

Обращение к регистрам специального назначения. Система команд микроконтроллеров PICmicro среднего семейства позволяет напрямую обращаться ко всем регистрам, включая регистры общего назначения. Существуют нюансы обращения к некоторым регистрам, которые должен учитывать разработчик программного обеспечения.

STATUS как регистр назначения при выполнении команды. Если обращение к регистру STATUS выполняется командой, которая воздействует на флаги Z, DC и C, то изменение этих трех битов командой заблокирована. Эти биты сбрасываются или устанавливаются согласно логике ядра микроконтроллера. Команды изменения регистра STATUS также не воздействуют на биты -TO и -PD. Поэтому результат выполнения команды с регистром STATUS может отличаться от ожидаемого. Например, команда CLRF STATUS сбросит три старших бита и установит бит Z (состояние регистра STATUS после выполнения команды 000uu1uu, где u - не изменяемый бит).

PCL как источник данных и регистр назначения при выполнении команды. Команды чтения, записи и "чтение - модификация - запись" регистра PCL могут иметь следующие результаты:

Чтение PCL: PCL → dest; PCLATH не изменяется.

Запись PCL: PCLATH → PCH;

8 - разрядное значение → PCL.

"Чтение - модификация - запись": PCL → операнд АЛУ;

PCLATH → PCH;

8 - разрядный результат → PCL.

Где PCH - старший байт счетчика команд (не адресуемый регистр), PCLATH - буфер старшего байта счетчика команд, dest - регистр назначения.

Битовые операции. При изменении любого бита регистра сначала регистр полностью читается из памяти данных, изменяется бит, обратно данные записываются в регистр ("чтение - модификация - запись"). Необходимо учитывать этот факт при обращении к некоторым регистрам специального назначения (например, регистры портов).

Примечание. Изменение состояния управляющих битов (включая флаги прерываний) выполняется в такте Q1, поэтому не возникает проблем при обращении командой "чтение - модификация - запись" к регистрам, содержащим эти биты.

Такты выполнения команд. Каждый цикл команды (TCY) состоит из четырех тактов (Q1-Q4). Такт Q равен по длительности периоду тактового генератора (TOSC). Такты Q обеспечивают жесткую синхронизацию декодирования, чтения данных, обработки данных, записи результата для каждого цикла команды. На рисунке 5.2 показано соотношение тактов Q к циклу команды.

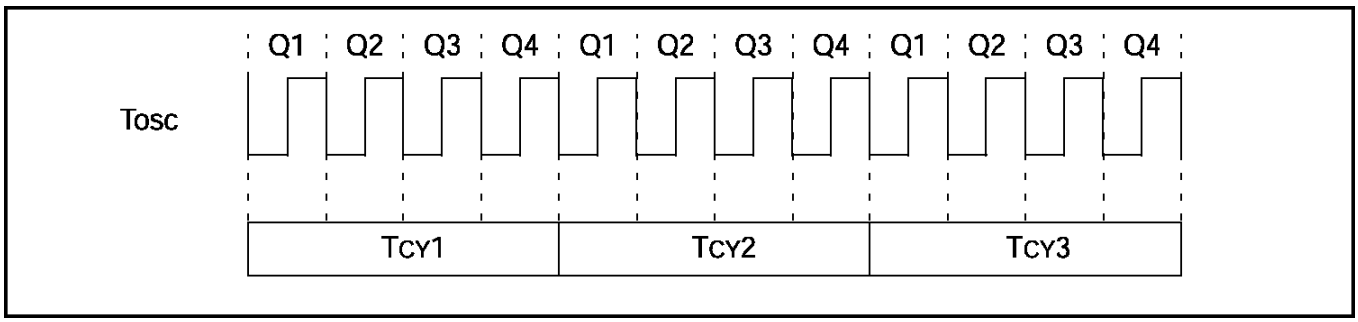

Рисунок 5.2 - Временная диаграмма циклического повторения тактов Q

Цикл команды (TCY), состоящий из 4-х тактов, обобщенно выглядит следующим образом:

Q1: Детектирование команды или принудительной пустой операции (NOP)

Q2: Операция чтения данных или отсутствие операции

Q3: Обработка данных

Q4: Операция записи данных или отсутствие операции.

Описание команд. Команды делятся на 2 вида: бит-ориентированные и байт-ориентированные. Ниже приводится их описание.

МНЕМОНИКА	*ОПИСАНИЕ*	*ПРИМЕРЫ*	Цикл	Флаги
Байт-ориентированные команды				
ADDWF f, d **Сложение W и f**	**Сложить содержимое регистров W и f.** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f.** *Косвенная адресация: для ее выполнения необходимо обратиться к регистру INDF.* *Оно вызовет действие с регистром, адрес которого указан в FSR.* *Косвенная запись в регистр INDF не вызовет никаких действий (кроме воздействия на флаги в регистре STATUS)* *Косвенное чтение INDF (FSR=0) даст результат 00h.* *9-й бит косвенного адреса (IRP) сохраняется в регистре STATUS<7>.* *Изменение адреса счетчика команд PC (вычисляемый переход) выполняется командой приращения к регистру PCL (ADDWF PCL).* *При этом необходимо следить, чтобы значение PCL не пересекало границу блока памяти данных (256 байт, иначе - работа по кольцу).* *PCL – младший байт (8 бит <7:0>) счетчикакоманд (PC), доступен для чтения и записи.* *PCH – старший байт (5 бит <12:8>) счетчика команд PC, не доступен для чтения и записи). Все операции с PCH происходят через дополнительный регистр PCLATH.* *В случае вычисляемого перехода, при переполнении PCL, инкремента PCH не происходит (флаг C поднимается).*	**ADDWF ABC,0** **До выполнения** **W=0x17** **ABC=0xC2** **После выполнения** **W=0xD9** **ABC=0xC2** *Косвенная адресация* **ADDWF INDF,1** **До выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x20)** **После выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x37)** *Вычисляемый переход* **ADDWF PCL,1** **До выполнения** **W=0x10** **PCL=0x37** **C=x** **После выполнения** **PCL=0x47** **C=0** **ADDWF PCL,1** **До выполнения** **W=0x10** **PCL=0xF7** **PCH=0x08** **C=x** **После выполнения** **PCL=0x07** **PCH=0x08** **C=1**	**1**	**C,DC, Z**
ANDWF f, d **Побитное И W и f**	**Выполняется побитное “И” содержимого регистров W и f.** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f.**	**ANDWF ABC,1** **До выполнения** **W=0x17 (00010111)** **ABC=0xC2(11000010)** **После выполнения** **W=0x17** **ABC=0x02 (00000010)** **ANDWF ABC,0** **До выполнения**	**1**	**Z**

		W=0x17 (00010111) **ABC=0xC2(11000010)** **После выполнения** W=0x02 (00000010) **ABC=0xC2** *Косвенная адресация* **ANDWF INDF,1** **До выполнения** **W=0x17** **FSR=0xC2 (по этому адресу "лежит" число 0x5A)** **После выполнения** **W=0x17** **FSR=0xC2 (по этому адресу "лежит" число 0x15)**		
CLRF f **Очистить f**	**Очистить содержимое регистра f и установить флаг Z**	**CLRF FLAG_REG** **До выполнения** **FLAG_REG=0x5A** **После выполнения** **FLAG_REG=0x00** **Z=1** *Косвенная адресация* **CLRF INDF** **До выполнения** **FSR=0xC2 (по этому адресу "лежит" число 0xAA)** **После выполнения** **FSR=0xC2 (по этому адресу "лежит" число 0x00)** **Z=1**	**1**	**Z**
CLRW **Очистить W**	**Очистить содержимое регистра W и установить флаг Z**	**CLRW** **До выполнения** **W=0x5A** **После выполнения** **W=0x00** **Z=1**	**1**	**Z**
COMF f, d **Инвертировать f**	**Инвертировать все биты в регистре f** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f**	**COMF REG1,0** **До выполнения** **REG1=0x13** **После выполнения** **REG1=0x13** **W=0xEC** **COMF REG1,1** **До выполнения** **REG1=0xFF** **После выполнения** **REG1=0x00**	**1**	**Z**

		Z=1 *Косвенная адресация* **COMF INDF,1** **До выполнения** **FSR=0xC2 (по этому адресу "лежит" число 0xAA)** **После выполнения** **FSR=0xC2 (по этому адресу "лежит" число 0x55)**		
DECF f, d **Вычесть 1 из f**	**Декремент содержимого регистра f** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f.**	**DECF CNT,1** **До выполнения** **CNT=0x01** **Z=0** **После выполнения** **CNT=0x00** **Z=1** **DECF CNT,0** **До выполнения** **CNT=0x10** **W=x** **Z=0** **После выполнения** **CNT=0x10** **W=0x0F** **Z=0** *Косвенная адресация* **DECF INDF,1** **До выполнения** **FSR=0xC2 (по этому адресу "лежит" число 0x01)** **После выполнения** **FSR=0xC2 (по этому адресу "лежит" число 0x00)** **Z=1**	**1**	**Z**
DECFSZ f, d **Вычесть 1 из f и пропустить если 0**	**Декремент содержимого регистра f с пропуском, если 0** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f.** **Если результат не равен 0 – исполняется следующая инструкция.** **Если результат = 0, то следующая инструкция не выполняется (пропускается, вместо нее выполняется "виртуальный" NOP), а команда выполняется за 2 цикла.**	**HERE DECFSZ CNT,1** **GOTO LOOP** **CONTINUE .** **.** **1) До выполнения** **CNT=0x01** **PC=адрес HERE** **После выполнения** **CNT=0x00** **PC=адрес CONTINUE** **2) До выполнения** **CNT=0x02** **PC=адрес HERE** **После выполнения** **CNT=0x01** **PC=адрес HERE+1**	**1(2)**	

INCF f, d **Прибавить 1 к f**	**Инкремент содержимого регистра f** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f.**	**1) INCF CNT,1** **До выполнения** **CNT=0xFF** **Z=0** **После выполнения** **CNT=0x00** **Z=1** **2) INCF CNT,0** **До выполнения** **CNT=0x10** **W=x** **Z=0** **После выполнения** **CNT=0x10** **W=0x11** **Z=0** *Косвенная адресация* **INCF INDF,1** **До выполнения** **FSR=0xC2 (по этому адресу “лежит” число 0xFF)** **Z=0** **После выполнения** **FSR=0xC2 (по этому адресу “лежит” число 0x00)** **Z=1**	**1**	**Z**
INCFSZ f, d **Прибавить 1 к f и пропустить если 0**	**Инкремент содержимого регистра f с пропуском, если 0** **Если d=0, результат сохраняется в регистре W.** **Если d=1, результат сохраняется в регистре f.** **Если результат не равен 0 – исполняется следующая инструкция.** **Если результат = 0, то следующая инструкция не выполняется (пропускается, вместо нее выполняется “виртуальный” NOP), а команда выполняется за 2 цикла.**	**HERE INCFSZ CNT,1** **GOTO LOOP** **CONTINUE .** **.** **1) До выполнения** **PC=адрес HERE** **CNT=0xFF** **После выполнения** **CNT=0x00** **PC=адрес CONTINUE** **2) До выполнения** **PC=адрес HERE** **CNT=0x00** **После выполнения** **CNT=0x01** **PC=адрес HERE+1**	**1(2)**	
		1) IORWF RES,0 **До выполнения** **RES=0x13** **W=0x91**		

IORWF f,d **Побитное “ИЛИ” W и f**	**Побитное “ИЛИ” содержимого регистров W и f** **Если d=0 – результат сохраняется в регистре W** **Если d=1 – результат сохраняется в регистре f.**	**После выполнения** **RES=0x13** **W=0x93** **Z=0** **2) IORWF RES,1** **До выполнения** **RES=0x13** **W=0x91** **После выполнения** **RES=0x93** **W=0x91** **Z=0** **3) IORWF RES,1** **До выполнения** **RES=0x00** **W=0x00** **После выполнения** **RES=0x00** **W=0x00** **Z=1** *Косвенная адресация* **IORWF INDF,1** **До выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x30)** **После выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x37)** **Z=0**	**1**	**Z**
MOVF f,d **Переслать f**	**Содержимое регистра f пересылается в регистр адресата** **Если d=0 – значение сохраняется в регистре W.** **Если d=1 – значение сохраняется в регистре f.** **d=1 используется для проверки содержимого f на ноль.**	**MOVF REG,0** **До выполнения** **W=0x00** **REG=0xC2** **После выполнения** **W=0xC2** **REG=0xC2** **Z=0** **MOVF REG,1** **1) До выполнения** **REG=0x43** **После выполнения** **REG=0x43** **Z=0** **2) До выполнения** **REG=0x00** **После выполнения** **REG=0x00**	**1**	**Z**

		Z=1 *Косвенная адресация* **MOVF INDF,1** **До выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x00)** **После выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x00)** **Z=1**		
MOVWF f **Переслать W в f**	**Переслать содержимое W в f**	**MOVWF OPTION** **До выполнения** **OPTION=0xFF** **W=0x4F** **После выполнения** **OPTION=0x4F** **W=0x4F** *Косвенная адресация* **MOVWF INDF** **До выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x00)** **После выполнения** **W=0x17** **FSR=0xC2 (по этому адресу “лежит” число 0x17)**	**1**	
NOP	**Нет операции**	**HERE NOP** **До выполнения** **PC=адрес HERE** **После выполнения** **PC=адрес HERE+1**	**1**	
RLF f, d **Циклический сдвиг f влево через перенос**	**Выполняется циклический сдвиг влево содержимого регистра f через бит C регистра STATUS** **Если d=0 – результат сохраняется в регистре W.** **Если d=1 – результат сохраняется в регистре f.**	**RLF REG,0** **До выполнения** **REG=11100110** **W=xxxxxxxx** **C=0** **После выполнения** **REG=11100110** **W=11001100** **C=1** *Косвенная адресация* **RLF INDF,1** **До выполнения** **FSR=0xC2 (по этому адресу “лежит” число 0x3A - 00111010)**	**1**	**C**

		**C=1** **После выполнения FSR=0xC2 (по этому адресу “лежит” число 0x75 - 01110101)** **C=0** **RLF INDF,1** **До выполнения FSR=0xC2 (по этому адресу “лежит” число 0xB9 - 10111001)** **C=0** **После выполнения FSR=0xC2 (по этому адресу “лежит” число 0x72 - 01110010)** **C=1**		
RRF f, d **Циклический сдвиг f вправо через перенос**	**Выполняется циклический сдвиг вправо содержимого регистра f через бит C регистра STATUS** **Если d=0 – результат сохраняется в регистре W.** **Если d=1 – результат сохраняется в регистре f.** 	**RRF REG,0** **До выполнения REG=11100110** **W=xxxxxxxx** **C=0** **После выполнения REG=11100110** **W=01110011** **C=0** *Косвенная адресация* **RRF INDF,1** **До выполнения FSR=0xC2 (по этому адресу “лежит число 0x3A – 00111010)** **C=1** **После выполнения FSR=0xC2 (по этому адресу “лежит” число 0x9D – 10011101)** **C=0** **RRF INDF,1** **До выполнения FSR=0xC2 (по этому адресу “лежит” число 0x39-00111001)** **C=0** **После выполнения FSR=0xC2 (по этому адресу “лежит” число 0x1C – 00011100)** **C=1**	**1**	**C**

SUBWF f, d **Вычесть W из f**	**Вычетание содержимого регистра W из регистра f.** **Если d=0 – результат сохраняется в регистре W.** **Если d=1 – результат сохраняется в регистре f.**	**SUBWF REG,1** **До выполнения** **REG=0x03** **W=0x02** **C=x** **Z=x** **После выполнения** **REG=0x01** **W=0x02** **C=1, Z=0** **(“+” результат)** **SUBWF REG,1** **До выполнения** **REG=0x02** **W=0x02** **C=x** **Z=x** **После выполнения** **REG=0x00** **W=0x02** **C=1, Z=1** **(“0” результат)** **SUBWF REG,1** **До выполнения** **REG=0x01** **W=0x02** **C=x** **Z=x** **После выполнения** **REG=0xFF** **W=0x02** **C=0, Z=0** **(“-“ результат)**	**1**	**C,DC, Z**
SWAPF f, d **Поменять местами полубайты в f**	**Поменять местами старший и младший полубайты регистра f.** **Если d=0 – результат сохраняется в регистре W.** **Если d=1 – результат сохраняется в регистре f.**	**SWAPF REG,0** **До выполнения** **REG=0xA5** **(1010 0101)** **W=x** **После выполнения** **REG=0xA5** **W=0x5A** (0101 1010) **SWAPF REG,1** **До выполнения** **REG=0xA5** **После выполнения** **REG=0x5A**	**1**	

		Косвенная адресация **SWAPF INDF,1** **До выполнения FSR=0xC2 (по этому адресу “лежит” число 0x20 – 0010 0000)** **После выполнения FSR=0xC2 (по этому адресу “лежит” число 0x02 – 0000 0010)**		
XORWF f, d **Побитное “исключающее ИЛИ” W и f**	*Сравнение содержимого регистров W и f (проверка на “одинаковость”)* **Побитное “Исключающее “ИЛИ” содержимого регистров W и f.** **Если d=0 – результат сохраняется в регистре W.** **Если d=1 – результат сохраняется в регистре f.**	**XORWF REG,1** **До выполнения REG=0xAF W=0xB5** **После выполнения REG=0x1A W=0xB5** **XORWF REG,0** **До выполнения REG=0xAF W=0xB5** **После выполнения REG=0xAF W=0x1A** *Косвенная адресация* **XORWF INDF,1** **До выполнения W=0xB5 FSR=0xC2 (по этому адресу “лежит” число 0xAF)** **После выполнения W=0xB5 FSR=0xC2 (по этому адресу “лежит” число 0x1A)**	**1**	**Z**

Бит - ориентированные команды (b-от 0 до 7)				
BCF f, b **Установить в 0 бит b регистра f**	**Установить в 0 бит b регистра f**	**BCF REG,7** **До выполнения REG=0xC7-11000111** **После выполнения REG=0x47- 01000111** *Косвенная адресация* **BCF INDF,3** **До выполнения FSR=0xC2 (по этому адресу "лежит" число 0x2F – 0010 1111)** **После выполнения FSR=0xC2 (по этому адресу "лежит" число 0x27 – 0010 0111)**	**1**	
BSF f, b **Установить в 1 бит b регистра f**	**Установить в 1 бит b регистра f**	**BSF REG,7** **До выполнения REG=0x0A- 00001010** **После выполнения REG=0x8A- 10001010** *Косвенная адресация* **BSF INDF,3** **До выполнения FSR=0xC2 (по этому адресу "лежит" число 0x20 – 0010 0000)** **После выполнения FSR=0xC2 (по этому адресу "лежит" число 0x28 – 0010 1000)**	**1**	
BTFSC f, b **Проверить бит b в регистре f, если b=0, то пропустить следующую инструкцию**	**Если бит b в регистре f =1, то исполняется следующая инструкция** **Если бит b в регистре f =0, то следующая инструкция не выполняется (пропускается, вместо нее выполняется "виртуальный" NOP), а команда выполняется за 2 цикла.**	**HERE BTFSC FLAG,4** **FALSE GOTO ABC** **TRUE .** **.** **1) До выполнения PC=адрес HERE FLAG=xxx0xxxx** **После выполнения Т.к. FLAG<4>=0, PC=адрес TRUE** **2) До выполнения**	**1(2)**	

		PC=адрес HERE FLAG=xxx1xxxx После выполнения Т.к. FLAG<4>=1, PC=адрес FALSE (исполняется GOTO ABC)		
BTFSS f, b **Проверить бит b в регистре f, если b=1, то пропустить следующую инструкцию**	**Если бит b в регистре f=0, исполняется следующая инструкция Если бит b в регистре f=1, то следующая инструкция не выполняется (пропускается, вместо нее выполняется "виртуальный" NOP), а команда выполняется за 2 цикла.**	**HERE BTFSS FLAG,4 FALSE GOTO ABC TRUE . . 1) До выполнения PC=адрес HERE FLAG=xxx0xxxx После выполнения Т.к. FLAG<4>=0, PC=адрес FALSE (исполняется GOTO ABC) 2) До выполнения PC=адрес HERE FLAG=xxx1xxxx После выполнения Т.к. FLAG<4>=1, PC=адрес TRUE**	**1(2)**	
Команды операций с константами		**(k – от 0 до 255)**		
ADDLW k **Сложить константу с W**	**Содержимое регистра W складывается с 8 – разрядной константой k. Результат сохраняется в регистре W**	**ADDLW 0x15 До выполнения W=0x10 После выполнения W=0x25 ADDLW REG До выполнения W=0x10 REG=0x37 (адрес регистра, а не его содержимое) После выполнения W=0x47 ADDLW CONST До выполнения**	**1**	**C,DC, Z**

		“Прописка” в “шапке” программы: **CONST EQU 0x37** **W=0x10** **После выполнения** **W=0x47**		
SUBLW k **Вычесть W из константы**	**Вычесть содержимое регистра W из 8 – разрядной константы k. Результат сохраняется в регистре W.**	**SUBLW 0x02** **До выполнения** **W=0x01** **C=?** **Z=?** **После выполнения** **W=0x01** **C=1, Z=0** **(“+” результат)** **SUBLW 0x02** **До выполнения** **W=0x03** **C=?** **Z=?** **После выполнения** **W=0xFF** **C=0, Z=0** **(“-“ результат)** **SUBLW 0x02** **До выполнения** **W=0x02** **C=?** **Z=?** **После выполнения** **W=0x00** **C=1, Z=1** **(“0” результат)** **SUBLW REG** **До выполнения** **W=0x10** **REG=0x37 (адрес** регистра, а не его содержимое) **После выполнения** **W=0x27** **C=1**	**1**	**C,DC, Z**

MOVLW k **Переслать константу в W**	**Переслать константу k в регистр W** **В неиспользуемых битах ассемблер устанавливает 0**	**MOVLW 0x5A** **До выполнения** **W=x** **После выполнения** **W=0x5A** **MOVLW REG** **До выполнения** **W=x** **REG=0x37 (адрес регистра, а не его содержимое)** **После выполнения** **W=0x37** **Z=0** **MOVLW CONST** **До выполнения** **“Прописка” в “шапке” программы:** **CONST EQU 0x37** **W=x** **После выполнения** **W=0x37**	**1**	
ANDLW k **Побитное “И” константы и W**	**Выполняется побитное “И” содержимого регистра W и 8 – разрядной константы k.** **Результат сохраняется в регистре W**	**ANDLW 0x5F (01011111)** **До выполнения** **W=0xA3 (10100011)** **После выполнения** **W=0x03 (00000011)** **ANDLW REG** **До выполнения** **W=0xA3** **REG=0x37 (адрес регистра, а не его содержимое)** **После выполнения** **W=0x23** **ANDLW CONST** **До выполнения** **“Прописка” в “шапке” программы:** **CONST EQU 0x37** **W=0xA3** **После выполнения** **W=0x23**	**1**	**Z**
		IORLW 0x35		

IORLW k **Побитное “ИЛИ” константы и W**	**Выполняется побитное “ИЛИ” содержимого регистра W и 8 – разрядной константы k.** **Результат сохраняется в регистре W.**	**(0x35 – 00110101)** **До выполнения** **W=0x9A (10011010)** **После выполнения** **W=0xBF (10111111)** **Z=0** **IORLW REG** **До выполнения** **W=0x9A** **REG=0x37 (адрес регистра, а не его содержимое)** **После выполнения** **W=0x9F** **Z=0** **IORLW CONST** **До выполнения** **W=0x9A** **“Прописка” в “шапке” программы:** **CONST EQU 0x37** **После выполнения** **W=0x9F** **Z=0** **IORLW 0x00** **До выполнения** **W=0x00** **После выполнения** **W=0x00** **Z=1**	**1**	**Z**

XORLW k **Побитное “Исключающее ИЛИ” константы и W**	*Сравнение содержимого регистра W и константы (проверка на “одинаковость”)* **Выполняется побитное “Исключающее ИЛИ” содержимого регистра W и 8 – разрядной константы k.** **Результат сохраняется в регистре W.**	**XORLW 0xAF(10101111)** **До выполнения W=0xB5 (10110101)** **После выполнения W=0x1A (00011010) Z=0** **XORLW REG** **До выполнения W=0xAF REG=0x37 (адрес регистра, а не его содержимое)** **После выполнения W=0x18 Z=0** **XORLW CONST** **До выполнения W=0xAF “Прописка” в “шапке” программы: CONST EQU 0x37** **После выполнения W=0x18 Z=0**	**1**	**Z**
Команды управления				
CALL **Условный переход (переход по стеку)**	**Выполнить условный переход.** **Адрес следующей инструкции (PC+1) “загружается” в вершину стека (TOS).** **11 бит адреса “загружаются” из кода команды в счетчик команд PC<10:0>.** **2 старших бита “загружаются” в счетчик команд PC<12:11> из регистра PCLATH.**	**HERE CALL ABC** **До выполнения PC=адрес HERE** **После выполнения PC=адрес ABC TOS=адрес HERE+1**	**2**	

GOTO k **Безусловный переход (стек не** задейст-вован)	**Выполнить безусловный переход. 11 бит адреса "загружаются" из кода команды в счетчик команд PC<10:0>. 2 старших бита "загружаются" в счетчик команд PC<12:11> из регистра PCLATH.**	**GOTO ABC** **После выполнения PC= адрес ABC**	**2**	
RETURN **Возврат по стеку**	**Возврат из подпрограммы. Вершина стека (TOS) "выгружается" в счетчик команд PC.**	**RETURN** **После выполнения PC=TOS (адрес, "выгруженный" из TOS)**	**2**	
RETLW k **Возврат по стеку с загрузкой константы в W**	**Возврат из подпрограммы. В регистр W загружается 8-разряд-ная константа. Вершина стека (TOS) "выгружается" в счетчик команд PC.**	**HERE CALL TABLE** **.** **.** **TABLE ADDWF PCL,1** **RETLW k1** **RETLW k2** **.** **.** **RETLW kn** **До выполнения W=0x07** **После выполнения W=значение k8 PC=TOS (адрес HERE+1)**	**2**	
RETFIE **Возврат по стеку из ПП обработки прерываний**	**Возврат из подпрограммы обработки прерываний. Вершина стека (TOS) загружается в счетчик команд PC. Осуществляется предварительное разрешение прерываний (бит №7 регистра INTCON {GIE} ус-танавливается в 1).**	**RETFIE** **После выполнения PC=TOS GIE=1**	**2**	
CLRWDT **Сброс WDT (сторожевого таймера)**	**Сброс WDT и предделителя** (если он подключен к WDT). **В регистре STATUS, биты -TO и -PD устанавливаются в 1.** **Коэффициент деления предделителя (если он подключен к WDT) не меняется.**	**CLRWDT** **До выполнения WDT и предделитель не сброшены Кдел. предделителя: 1:128** **После выполнения WDT и предделитель сброшены**	**1**	**-TO -PD**

		Кдел. предделителя: 1:128 -TO=1 -PD=1		
SLEEP **Переход в режим SLEEP**	**Переход в “спящий режим” Сброс флага включения питания (-PD) в 0. Установка флага переполнения WDT (-TO) в 1.** **Сброс WDT и его предделителя. Перевод м/контроллера в режим SLEEP и выключение тактового генератора.**	**SLEEP**	**1**	**-TO -PD**

*Контрольные вопросы***.**

1. Что такое бит-ориентированные команды микроконтроллера? Приведите примеры их применения.

2. Что такое байт-ориентированные команды микроконтроллера? Приведите примеры их применения.

3. Что представляют собой команды управления микроконтроллером, в каких случаях они могут применяться?

4. Приведите примеры программных операций с константами.

5. Какова роль регистра STATUS микроконтроллера?

6. Что такое такты выполнения команд микроконтроллера?

ГЛОССАРИЙ

A

A/D

АЦП

См. Analog to Digital.

Acquisition Time (TAQC)

Длительность заряда конденсатора

Этот параметр связан с модулем АЦП. TAQC - интервал времени, в течение которого внутренний конденсатор АЦП заряжается до напряжения подключенного входного канала. Когда бит GO установлен в '1', то аналоговый вход отсоединен от внутреннего конденсатора, выполняется преобразование.

ALU

АЛУ

Арифметико-логическое устройство. Модуль ядра микроконтроллера, отвечающий за математические (сложение, вычитание и др.), логические ("и", "или" и др.) и операции сдвига.

Analog to Digital (A/D)

Аналого-цифровое преобразование

Входной аналоговый сигнал преобразуется в эквивалентный цифровой код.

Assembly Language

Ассемблер

Символический язык программирования, с помощью которого машинные коды представляются в удобно читаемой форме.

B

Bank

Банк

Метод адресации памяти данных. Команды среднего семейства микроконтроллеров PICmicro имеют 7 бит для прямой адресации памяти данных (максимум 128 байт), включая регистры специального назначения. Для того, чтобы была возможность реализовать больший объем памяти данных, она была разбита на банки по 128 байт. Выбрать требуемый банк можно с помощью битов RP1:RP0. Максимум может быть реализовано 4 банка памяти данных (два управляющих бита).

Baud

Бод

Скорость передачи данных по последовательным интерфейсам (эквивалентно бит/с).

BCD

См. Binary Coded Decimal (BCD).

Binary Coded Decimal (BCD)

Двоично-десятичное кодирование чисел

Каждые 4 бита определяют цифру от 0 до 9. Как правило, один байт содержит две цифры (диапазон чисел от 0 до 99).

BOR

См. Brown-out Reset.

Brown-out

Снижение напряжение питания

Условие, при котором напряжение питания опускается ниже определенного значения. Это может происходить при коммутации мощной нагрузки.

Brown-out Reset (BOR)

Сброс по снижению напряжения питания

Схема, которая переводит микроконтроллер в состояние сброса, если напряжение питания стало ниже установленного значения. Некоторые микроконтроллеры имеют интегрированную схему BOR (если нет внутренней схемы BOR, то может возникнуть необходимость в построении внешней схемы).

Bus width

Разрядность шины

Число бит данных передаваемых по шине. Разрядность шины данных - 8 бит. Разрядность шины программ для микроконтроллеров среднего семейства - 14 бит.

C

Capture

Захват

Функция CCP модуля, в которой значение таймера записывается в регистры захвата при возникновении условия захвата.

CCP

Захват, сравнение, широтно-импульсный модулятор (PWM). Этот модуль может быть настроен для работы в одном из режимов: захват данных, сравнение или ШИМ.

Common RAM

Общее ОЗУ

Область памяти данных, которая доступна во всех банках памяти данных. Как правило, эта область имеет адреса от 70h до 7Fh (включительно). В этой области удобно сохранять часто меняющиеся переменные и контекст программы при обработке прерываний.

Compare

Сравнение

Функция модуля CCP, при которой выполнится указанное действие, когда значение таймера соответствует значению в регистрах сравнения.

Compare Register

Регистр сравнения

16 - разрядный регистр, в котором хранится значение, сравниваемое с 16 - разрядным значением таймера. Однократно выполняется указанное действие, когда значение таймера становится равным значению регистра сравнения.

Capture Register

Регистр захвата

16 - разрядный регистр, в который загружается 16 - разрядное значение таймера TMR1, когда выполняется условие захвата.

Configuration Word

Слово конфигурации

В слове конфигурации определяются параметры работы микроконтроллера (режим работы тактового генератора, включение WDT, включение таймера PWRT и др.). Эти параметры определяются во время программирования микроконтроллера. Для микроконтроллеров с EPROM памятью программ значение бита '1' может быть изменено на '0'. Память программ должна быть стерта, чтобы восстановить значение '1'.

Conversion Time (Tconv)

Время преобразования

Параметр связан с модулем АЦП. Интервал времени, необходимый для нормального преобразования входного аналогового сигнала в соответствующий цифровой код.

CPU

ЦПУ

Центральное процессорное устройство. Выполняет декодирование команд, определяет необходимые операнды и требуемую операцию. Управляет работой АЛУ для выполнения логических, арифметических и других операций.

D

D/A

ЦАП

См. Digital to analog.

Data Bus

Шина данных

Шина, необходимая для передачи данных из/в память данных.

Data EEPROM

EEPROM память данных

Электрически перепрограммируемая память данных. Эта память данных может быть запрограммирована командами ЦПУ для сохранения необходимых приложению данных при выключении питания (энергонезависимая память).

Data Memory

Память данных

Память, подключенная к шине данных, выполненная как статическое ОЗУ. В памяти данных размещаются регистры общего и специального назначения.

Direct Addressing

Прямая адресация

Адрес памяти данных содержится в команде микроконтроллера. Обращение будет выполняться к регистру с указанным адресом.

Digital to Analog

Цифро-аналоговое преобразование

Цифровой код преобразуется в соответствующее аналоговое напряжение (ток).

E

EEPROM

Электрически стираемое постоянно запоминающее устройство. Микросхемы, с данным типом памяти программ, могут быть внутрисхемно стерты и повторно запрограммированы.

EPROM

Электрически программируемое постоянное запоминающее устройство. Микросхемы, с данным типом памяти программ, могут быть внутрисхемно запрограммированы. Стирание EPROM памяти выполняется под действием УФ излучения.

EXTRC

Внешняя RC цепочка. Некоторые микроконтроллеры имеют режим тактового генератора с внешней RC цепочкой. Эквивалентно RC режиму тактового генератора.

F

Flash Memory

Flash память

Микросхемы, с данным типом памяти программ могут быть внутрисхемно стерты и повторно запрограммированы. Flash технология памяти программ функционально эквивалентна EEPROM памяти.

FOSC

Тактовая частота микроконтроллера.

G

GIO

Общий порт ввода/вывода.

GPIO

Универсальный порт ввода/вывода

GPR

Регистры общего назначения (ОЗУ). Эти регистры могут использоваться для хранения переменных программы пользователя.

H

Harvard Architecture

Гарвардская архитектура

В данной архитектуре микроконтроллеров шины памяти данных и памяти программ разделены между собой. Это позволяет выполнять одновременный доступ к памяти программ и памяти данных, что увеличивает производительность ядра микроконтроллера.

Holding Capacitor

Удерживающий конденсатор

Конденсатор расположен в модуле АЦП. Этот конденсатор должен заряжаться до напряжения на аналоговом входе перед началом преобразования. Как только начато преобразование, конденсатор отсоединяется от аналогового входа. Напряжение на конденсаторе используется для преобразования.

HS

Высокоскоростной режим генератора

Один из режимов тактового генератора. Тактовый генератор настроен таким образом, чтобы поддерживать высокую тактовую частоту микроконтроллера (от 4МГц до 20МГц).

I

I2C

Inter-Integrated Circuit. Двухпроводный интерфейс связи. Один из режимов SSP модуля.

Indirect Addressing

Косвенная адресация

Случай, когда адрес регистра памяти данных не содержится в команде. Обращение выполняется к регистру INDF, а операция выполняется с регистром, адрес которого указан в FSR. Всегда будет выполняться обращение к регистру с адресом, который записан в регистр FSR.

Instruction Bus

Шина команд

Шина для передачи кода команды из памяти программ в ЦПУ.

Instruction Fetch

Выборка команды

Поскольку реализована гарвардская архитектура, то одновременно происходит выполнение текущей команды и выборка следующей. Как только будет выполнена текущая команда, следующая команда подготовлена к детектированию.

Instruction cycle

Цикл команды

Выполнение каждой команды состоит из нескольких действий: декодирование, чтение данных, выполнение, запись данных. Некоторые команды могут содер-

жать не все действия (см. описание конкретной команды). Цикл команды (TCY) состоит из четырех тактов генератора (TOSC).

Interrupt

Прерывания

Событие, по которому ЦПУ вынужден перевести выполнение программы по адресу вектора прерываний (0004h). Перед изменением значение счетчика команд PC текущее значение сохраняется в вершине стека, чтобы была возможность продолжить выполнение программы.

INTRC

Внутренняя RC цепочка. Некоторые микроконтроллеры имеют режим тактового генератора с внутренней RC

цепочкой.

L

LCD

ЖКИ

Жидкокристаллический дисплей. Используется для визуального контроля работы устройства.

LED

Светодиод

Используется для визуального контроля работы устройства.

Literal

Константа

Неизменяемое значение, которое входит в состав команды.

Long Word Instruction

Длинное слово команды

В слово команды входит вся необходимая информация для выполнения операции (код операции и данные). Выполнение и выборка команды происходит за один машинный цикл, т.к. все команды однословные.

LP

Низкоскоростной режим генератора

Один из режимов тактового генератора. Тактовый генератор настроен таким образом, чтобы поддерживать низкую тактовую частоту микроконтроллера (до 200кГц).

LSb

Самый младший бит.

LSB

Самый младший байт.

M

Machine cycle

Машинный цикл

Единица времени выполнения программы микроконтроллера. Для PICmicro эта единица времени равна 4 тактам тактового генератора (4 TOSC). Обозначается как TCY.

MSb

Самый старший бит.

MSB

Самый старший байт.

N

Non-Return to Zero

Без возвращения к нулю

Метод кодирования данных при передаче по каналам связи. Логическая '1' передается как высокий уровень сигнала, логический '0' - как низкий уровень сигнала. Уровень сигнала в линии по умолчанию - высокий.

NRZ

Смотрите Non-Return to Zero.

O

Opcode

Код операции

Часть 14-разрядного слова команды, определяющая выполняемую операцию. Код операции может иметь разную длину в зависимости от типа команды (от 4 бит). В остальной части слова команды содержится аргумент.

Oscillator Start-up Timer (OST)

Таймер запуска генератора

Таймер отсчитывает 1024 такта генератора пред отпусканием внутреннего сигнала сброса микроконтроллера.

OST

Смотрите Oscillator Start-up Timer.

P

Pages

Страницы

Метод адресации памяти программ. Микроконтроллеры среднего семейство имеют в слове команд CALL и GOTO 11 - разрядное поле для адресации памяти программ, что позволяет непосредственно адресовать 2кслов памяти. Для адреса-

ции большего объема памяти вся память программ была разделена на страницы по 2кслова. Выбрать нужную страницу можно настройкой битов в регистре PCLATH<5:4>. Всего может быть реализовано 4 страницы памяти программ (два управляющих бита).

Parallel Slave Port (PSP)

Ведомый параллельный порт

Параллельный коммуникационный 8 - разрядный порт для подключения к шине микропроцессора.

POP

Термин, обозначающий восстановление информации из стека (программными или аппаратными средствами). Смотрите PUSH.

Postscaler

Выходной делитель

Схема, замедляющая возникновение прерывания (или сброс WDT) от таймера/счетчика.

Power-on Reset (POR)

Сброс по включению питания

Схема, обнаруживающая повышение напряжения питания от уровня 0В. Если напряжение повышается с 0В, то происходит сброс по включению питания и запускается таймер PWRT.

Power-up Timer (PWRT)

Таймер включения питания

Таймер, удерживающий микроконтроллер в состоянии сброса после выполнения сброса POR, чтобы позволить напряжению питания достигнуть номинального уровня. После завершения отсчета таймера PWRT, запускается таймер OST, если он

включен (таймер OST включен для любого режима тактового генератора с кварцевым или керамическим резонатором).

Prescaler

Предделитель

Схема, уменьшающая частоту входного тактового сигнала для таймера/счетчика.

Program Bus

Шина программ

Шина предназначенная для передачи кода команды из памяти программ в ЦПУ.

Program Counter

Счетчик команд

Регистр счетчика команд, в котором хранится адрес следующей выполняемой команды.

Program Memory

Память программ

Любая память, подключенная к шине памяти программ. Статические данные могут сохраняться в памяти программ (например в виде таблиц).

PSP

Смотрите Parallel Slave Port.

Pulse Width Modulation (PWM)

Широтно-импульсная модуляция (ШИМ)

Последовательный сигнал, информация в котором представлена как длительность импульса высокого уровня с постоянной частотой. Вывод PWM модуля CCP требует минимального программного обеспечения для генерации ШИМ сигнала.

PUSH

Термин, обозначающий сохранение информации в стеке (программными или аппаратными средствами). Смотрите POP.

PWM

ШИМ

Смотрите Pulse Width Modulation.

Q

Q - cycles

Q - циклы

Тоже самое, что и цикл тактового генератора. 4 Q - цикла равно циклу команд TCY.

R

RC

Резистор-конденсатор

Заданный по умолчанию режим тактового генератора микроконтроллера. Наиболее дешевый (менее точный) режим тактового генератора. Максимальная рекомендованная частота 4МГц. (см. EXTRC).

Read-Modify-Write

Чтение - Модификация - Запись

Обозначение операции - чтение данных из регистра, изменение значения, запись нового значения в регистр. Это может быть выполнено в одном или нескольких циклах команды.

Register File

Файл регистров

Память данных с регистрами общего и специального назначения.

ROM

ПЗУ

Постоянное запоминающее устройство. Память, которая запрограммирована и не может быть изменена.

S

Sampling Time

Время выборки

Интервал времени, необходимый для получения одного результат преобразования АЦП. Он включает время заряда конденсатора и время преобразования.

Serial Peripheral Interface (SPI)

Последовательный периферийный интерфейс

Один из режимов модуля SSP. Как правило 3-х проводной интерфейс: линия входящих данных, линия исходящих данных, линия синхронизации. Это синхронный интерфейс, т.к. присутствует сигнал синхронизации.

SFR

Регистры специального назначения, содержащие биты управления ядром микроконтроллера и периферийными модулями.

Single cycle instruction

Одно-цикловые команды

Команды, которые выполняются за один машинный цикл (TCY).

Sleep

Режим пониженного энергопотребления с выключенным тактовым генератором. В этом режиме микроконтроллер потребляет минимальный ток. Некоторые периферийные модули могут продолжать работать в Sleep режиме.

Special Function Registers (SFR)

Регистры специального назначения.

Содержат биты управления ядром микроконтроллера и периферийными модулями.

SPI

Смотрите Serial Peripheral Interface.

Stack

Стек

Часть ЦПУ, в которой сохраняется адрес возврата для продолжения выполнения программы. Стек загружается из счетчика команд при выполнении команды CALL или возникновении прерывания.

T

TAD

Время получения одного бита результата при выполнении аналого-цифрового преобразования.

TCY

Длительность выполнения одно-цикловой команды микроконтроллера (4 TOSC).

TOSC

Период тактового генератора микроконтроллера.

U

USART

Универсальный синхронно-асинхронный приемопередатчик. Этот периферийный модуль может использоваться как полнодуплексный последовательный интерфейс связи или полудуплексный синхронный интерфейс. В асинхронном режиме может использоваться для связи с персональным компьютером.

V

Voltage Reference (VREF)

Источник опорного напряжения

Уровень напряжения, который может использоваться как опорный для модуля АЦП или модуля компараторов.

Von Neumann Architecture

Традиционная архитектура

Память программ и память данных находятся в одной и той же области. Это означает, что обращение к памяти программ и памяти данных выполняется последовательно (меньшая производительность ядра).

W

W Register

Смотрите Working Register.

Watchdog Timer (WDT)

Сторожевой таймер

Применяется для улучшения помехоустойчивости устройства. WDT выполняет сброс микроконтроллера, если не был вовремя очищен, что позволяет предотвратить "зависание" программы. Источником тактового сигнала для WDT является отдельный внутренний RC генератор.

WDT

Смотрите Watchdog Timer.

Working Register (W)

Рабочий регистр (W)

Этот регистр можно рассматривать как аккумулятор микроконтроллера. Используется как операнд в АЛУ при выполнении команд с двумя операндами.

X

XT

Один из режимов тактового генератора. Применяется при тактовой частоте от 100кГц до 4МГц.

СПИСОК ЛИТЕРАТУРЫ.

1. Microchip.com [электронный ресурс]. – Режим доступа: http://microchip.com/.

2. Microchip.ru [электронный ресурс]. – Режим доступа: http://microchip.ru/.

3. Белоус А. И. Основы схемотехники микроэлектронных устройств / А. И. Белоус, В. А. Емельянов, А. С. Турцевич. – М. : Техносфера, 2012. – 472 с.

4. Новиков Ю. В. Основы микропроцессорной техники: уч. пособие / Ю. В. Новиков, П. К. Скоробогатов. – 4-е изд., испр. – М. : Интернет-Университет Информационных Технологий : БИНОМ, Лаборатория знаний, 2009. – 357 с.

5. Чернышев А. Ю. Электронная и микропроцессорная техника: уч. пособие / А. Ю. Чернышев, Е. А. Шутов. – Томск : Изд-во Томского политехнического университета, 2010. – 135 с.

Харченко Дмитрий Павлович, **Николаенко** Сергей Анатольевич,
Волошин Александр Петрович, **Цокур** Дмитрий Сергеевич

СХЕМОТЕХНИКА:
Внутреннее устройство и программирование PIC-микроконтроллеров

Учебное пособие

В авторской редакции

Дизайн обложки – Д.П. Харченко

Подписано в печать 15.12.2014. Формат 60 x 84 $^{1}/_{16}$
Усл. печ. л. – 5,7. Уч.-изд. л. – 4,5.
Тираж 100 экз. Заказ № 14.

Типография Кубанского государственного аграрного университета.
350044, г. Краснодар, ул. Калинина, 13