Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра радиотехнических систем

В.Н. Левкович

АРХИТЕКТУРА И ОСНОВЫ ПРОГРАММИРОВАНИЯ ОДНОКРИСТАЛЬНЫХ МИКРОКОНТРОЛЛЕРОВ PIC16F84

Методическое пособие
к лабораторным работам и курсовому проектированию
по дисциплине “Вычислительные и микропроцессорные устройства»
для студентов специальностей 39 01 02 «Радиоэлектронные системы» и
39 01 01 «Радиотехника» БГУИР всех форм обучения

Минск 2002

УДК 681.325.5-181.48(075.8)
ББК 32.973.26-04 я 73
Л 37

Левкович В.Н.

Л 37 Архитектура и основы программирования однокристальных микроконтроллеров семейства PIC16F84: Метод. пособие к лаб. работам и курсовому проектированию по дисциплине “Вычислительные и микропроцессорные устройства» для студентов спец. 39 01 02 «Радиоэлектронные системы» и 39 01 01 «Радиотехника» БГУИР всех форм обучения /В.Н. Левкович. – Мн.: БГУИР, 2002.- 67 с.: ил.
ISBN 985-444-423-6.

В методическом пособии рассмотрены структурная схема, организация памяти программ и памяти данных, порты ввода/вывода, специальные функции, а также система команд однокристальных микроконтроллеров семейства PIC16F84 и основные правила составления программ на языке Ассемблер. Пособие предназначено для начинающих изучение технологии проектирования устройств на микроконтроллерах и поэтому содержит лишь основные сведения по указанным вопросам.

УДК 681.325.5-181.48(075.8)
ББК 32.973.26-04 я 73

ISBN 985-444-423-6

ВВЕДЕНИЕ

Однокристальные микроконтроллеры (ОМК) содержат в одном корпусе интегральной микросхемы все функциональные блоки электронной вычислительной машины: арифметико-логическое устройство (АЛУ), устройство управления (УУ), операционные регистры, управляющие регистры, память программ, память данных, порты ввода/вывода. Кроме этого, на кристалле микросхемы часто размещают и периферийные устройства, такие, как аналого-цифровой преобразователь (АЦП), цифроаналоговый преобразователь (ЦАП), компараторы, таймеры, энергонезависимая память, драйверы стандартных последовательных интерфейсов и др. Однокристальные микроконтроллеры предназначены для работы по одной программе, записываемой в энергонезависимую память программ и не меняемой в течение всего цикла эксплуатации. Они, как правило, являются встроенными в аппаратуру вычислительными машинами и выполняют функции управления, а также формирования и обработки сигналов и информации.

В настоящее время ОМК производят несколько десятков фирм. Среди них микроконтроллеры PICmicro американской фирмы Microchip Technology Inc. отличаются высокой производительностью, низким энергопотреблением, гибкой и развитой архитектурой, широкими функциональными возможностями, развитой периферией, простотой в освоении, низкой стоимостью. В зависимости от производительности и функциональных возможностей PICmicro подразделяются на семейства: PIC12, PIC16, PIC17 и PIC18. Самые простые модели выпускаются в 8-выводных, а сложные – в 80-выводных корпусах. При этом все семейства имеют общую базовую архитектуру и систему команд, что упрощает их изучение. Такое сочетание потребительских свойств сделало PICmicro одними из самых популярных ОМК среди разработчиков радиоэлектронной аппаратуры.

Модель PIC16F84 является почти идеальной для учебных целей. Она проста, но при этом несет в себе важнейшие черты всех семейств, что облегчает освоение новых моделей микроконтроллеров. Наличие встроенной электрически перепрограммируемой памяти программ позволяет легко тестировать разрабатываемые схемы и программы. Обучаемому может быть предоставлена возможность пройти полный путь от постановки задачи до работающей конструкции.

1. **АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРА PIC16F84**

1.1. Общие сведения

PIC16F84 разработан в соответствии с модифицированной гарвардской архитектурой и изготавливается по КМОП-технологии. Он имеет внутреннюю энергонезависимую электрически перепрограммируемую память программ (EEPROM) емкостью 1024 14-разрядных слов, 8-битную длину машинного слова и 64-байтовую внутреннюю память данных. Система команд включает 35 инструкций. Все команды имеют длину в одно слово шириной 14 бит и исполняются за один цикл (400 нс при тактовой частоте 10 МГц), кроме команд перехода, которые выполняются за два цикла (800 нс). PIC16F84 имеет прерывание, срабатывающее от четырех источников, и восьмиуровневый аппаратный стек. Периферия включает в себя 8-битный таймер/счетчик с 8-битным программируемым предварительным делителем (фактически 16-битный таймер), 64 8-битных энергонезависимых ячеек памяти данных и 13 линий двунаправленного ввода/вывода. Высокая нагрузочная способность (25 мА максимальный втекающий ток, 20 мА максимальный вытекающий ток) линий ввода/вывода упрощает внешние драйверы, за счет чего уменьшается общая стоимость разрабатываемой системы.

Микроконтроллер способен работать в широком диапазоне тактовых частот – от 0 до 10 МГц, широком диапазоне питающих напряжений – от 2 до 6 В, в широком температурном диапазоне – от –40 до +125 °С. Он отличается также низким энергопотреблением: 3 мА при напряжении питания 5 В и тактовой частоте 4 МГц, 50 мкА при питании 2 В и тактовой частоте 32 кГц.

Разработки на базе PIC-контроллеров поддерживаются ассемблером, программным симулятором, внутрисхемным эмулятором и программатором.

Микроконтроллер PIC16F84 подходит для широкого спектра применений: от схем высокоскоростного управления электрическими

двигателями до экономичных удаленных приемопередатчиков, показывающих приборов и связных процессоров. Наличие энергонезависимой памяти данных позволяет подстраивать параметры в прикладных программах (коды передатчика, скорости двигателя, частоты приемника и т.д.) в процессе эксплуатации.

Малые размеры корпусов, как для обычного, так и для поверхностного монтажа, делают эту серию микроконтроллеров пригодной для портативных приложений. Низкая цена, экономичность, быстродействие, простота использования и гибкость ввода/вывода делают PIC1684 привлекательным даже в тех областях, где ранее не применялись микроконтроллеры.

Программа, записанная в ПЗУ, может быть защищена от считывания при помощи установки бита защиты в слове конфигурации. В режиме защиты программы содержимое памяти программы не может быть прочитано в исходном виде, тем самым невозможно реконструировать записанную программу. Кроме того, при установленном бите защиты невозможно допрограммировать контроллер.

1.2. Структурная организация

Структурная схема PIC16F84 показана на рис. 1.1. Архитектура микроконтроллера основана на концепции раздельных шин и областей памяти для данных и для программ (гарвардская архитектура). Это увеличивает скорость обмена по сравнению с традиционной принстонской архитектурой, в которой команды и данные передаются по одной и той же шине. Разделение шин команд и данных позволяет увеличить разрядность команды по сравнению с разрядностью данных. Шина данных и память данных (ОЗУ) имеют ширину 8 бит, а программная шина и программная память (ПЗУ) - ширину 14 бит. Такая концепция обеспечивает простую, но мощную систему однословных команд, разработанную так, что битовые, байтовые и регистровые операции

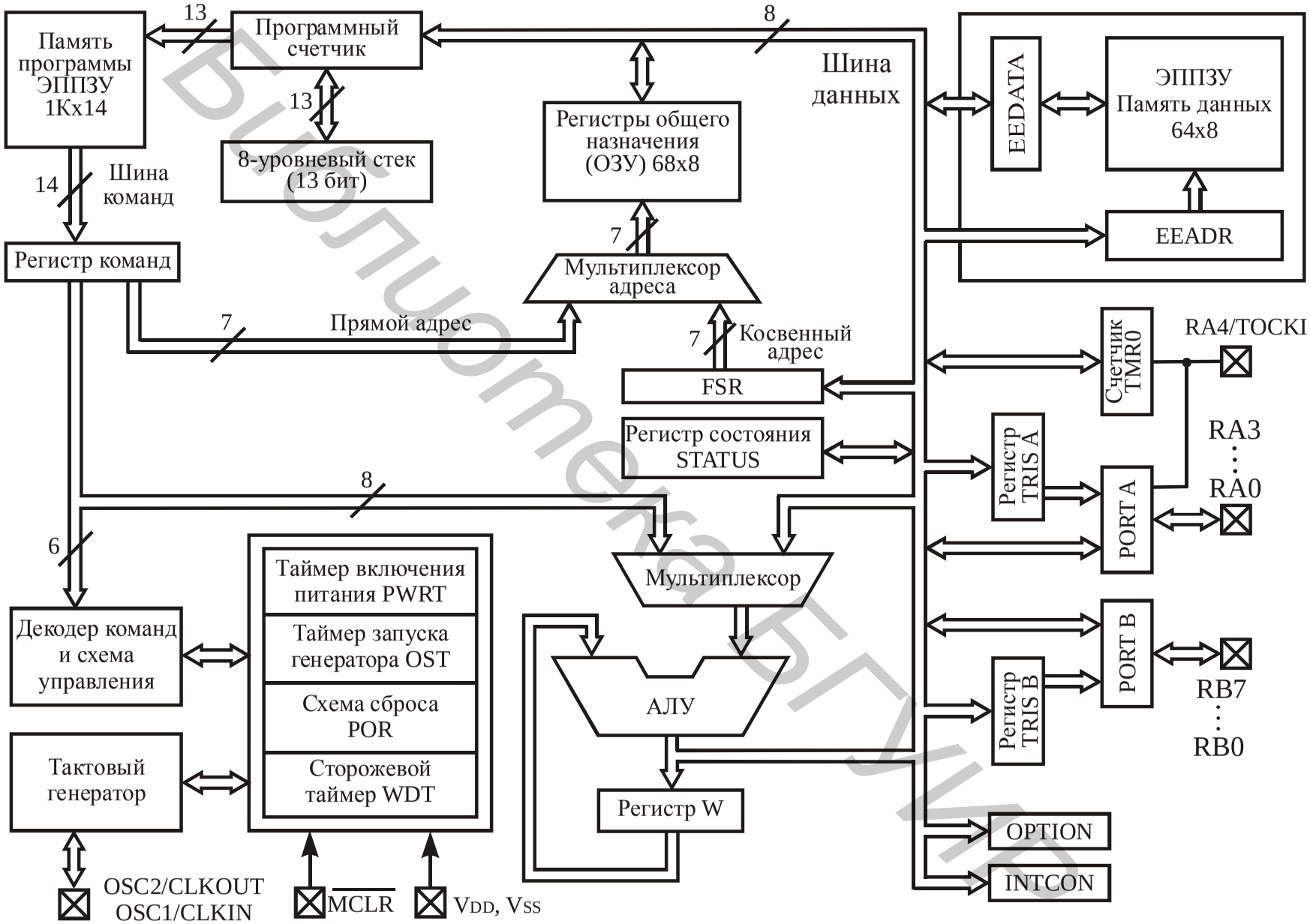


Рис. 1.1. Структурная схема PIC16F84

работают с высокой скоростью и с перекрытием по времени выборок команд и циклов выполнения. 14-битовая ширина программной памяти обеспечивает выборку 14-битовой команды за один цикл. Двухступенчатый конвейер обеспечивает одновременную выборку следующей и исполнение текущей команды. Все команды выполняются за один машинный цикл, исключая команды переходов, которые выполняются за два цикла. В PIC16F84 программная память объемом 1Кх14 расположена внутри кристалла. Исполняемая программа может находиться только во внутреннем ПЗУ.

Регистры PIC16F84 разделяются на две функциональные группы: специальные и общего назначения. Специальные регистры используются для управления режимами работы функциональных блоков контроллера, регистры общего назначения (память данных) - для хранения переменных.

Микроконтроллер PIC16F84 использует прямую и косвенную адресацию всех регистров и ячеек памяти. Все специальные регистры и счетчик команд также адресуются как память данных. Ортогональная (симметричная) система команд позволяет выполнять любую операцию с любым регистром, используя любой из названных выше методов адресации. Это облегчает программирование для PIC16F84 и значительно уменьшает время, необходимое на обучение работе с микроконтроллером.

В микроконтроллерах PIC16F84 имеется 8-разрядное арифметико-логическое устройство (АЛУ) и рабочий регистр W. АЛУ выполняет сложение, вычитание, сдвиг, битовые и логические операции. В командах, обрабатывающих два операнда, один из операндов содержится в рабочем регистре W. Второй операнд может быть константой или содержимым любого регистра ОЗУ. В командах с одним операндом операнд может быть содержимым рабочего регистра или любого регистра ОЗУ. Для выполнения всех операций АЛУ используется рабочий регистр W, который не может быть прямо адресован. Результат операции в АЛУ помещается либо в рабочий регистр W, либо в регистр ОЗУ.

В зависимости от результата выполнения операции изменяются значения битов переноса (C), десятичного переноса (DC) и нулевого результата (Z) в регистре состояния STATUS. При вычитании биты C и DC работают как биты заема и десятичного заема соответственно.

Условное графическое изображение микроконтроллера показано на рис. 1.2, а назначение выводов приведено в табл. 1.1.

Таблица 1.1

Наименование	Номер	Тип	Логические уровни	Назначение
1	2	3	4	5
				PORTA является двунаправленным портом ввода/вывода
RA0	17	I/O	TTL	
RA1	18	I/O	TTL	
RA2	1	I/O	TTL	
RA3	2	I/O	TTL	
RA4/T0CKI	3	I/O	ST	Выход с открытым стоком/вход таймера TMR0
				PORTB является двунаправленным портом ввода/вывода. Имеет на всех выводах программно включаемые подтягивающие резисторы
RB0/INT	6	I/O	TTL/ST(1)	Вход внешних прерываний
RB1	7	I/O	TTL	
RB2	8	I/O	TTL	
RB3	9	I/O	TTL	
RB4	10	I/O	TTL	Прерывание по изменению состояния
RB5	11	I/O	TTL	Прерывание по изменению состояния
RB6	12	I/O	TTL/ST(2)	Прерывание по изменению состояния, тактирование при программировании
RB7	13	I/O	TTL/ST(2)	Прерывание по изменению состояния, данные при программировании
MCLR/V_{PP}	4	I	ST	Вход сброса/напряжение программирования. Сброс низким уровнем
OSC1/CLKIN	16	I	ST/CMOS(3)	Вход генератора/внешняя тактовая частота
OSC2/CLKOUT	15	0	—	Выход генератора. Подключается к резонатору. В режиме RC-генератора выход 1/4 тактовой частоты OSC1

Окончание табл. 1.1

1	2	3	4	5
V_{DD}	14	P	—	Положительное напряжение питания
V_{SS}	5	P	—	Общий вывод

Примечания:

1. I - вход, O - выход, I/O - вход/выход, P - питание, — - не используется, TTL - вход ТТЛ, ST - вход с триггером Шмитта.
2. Буфер является триггером Шмитта при использовании в режиме внешних прерываний.
3. Буфер является триггером Шмитта при использовании в режиме последовательного программирования.
4. Буфер является триггером Шмитта при использовании в режиме RC-генератора и КМОП-входом — в остальных случаях.

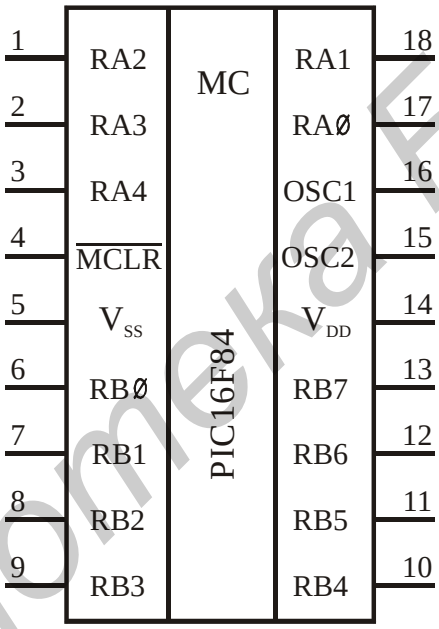


Рис. 1.2. Условное графическое изображение PIC16F84

Входная тактовая частота, поступающая с вывода OSC1/CLKIN, внутри делится на четыре, и из нее формируются четыре циклические неперекрывающиеся тактовые последовательности Q1, Q2, Q3 и Q4. Счетчик команд увеличивается в такте Q1, команда считывается из памяти программы и защелкивается в регистре команд в такте Q4. Команда декодируется и выполняется в течение последующего цикла в тактах Q1...Q4. Временная диаграмма тактирования и выполнения команд изображена на рис. 1.3.

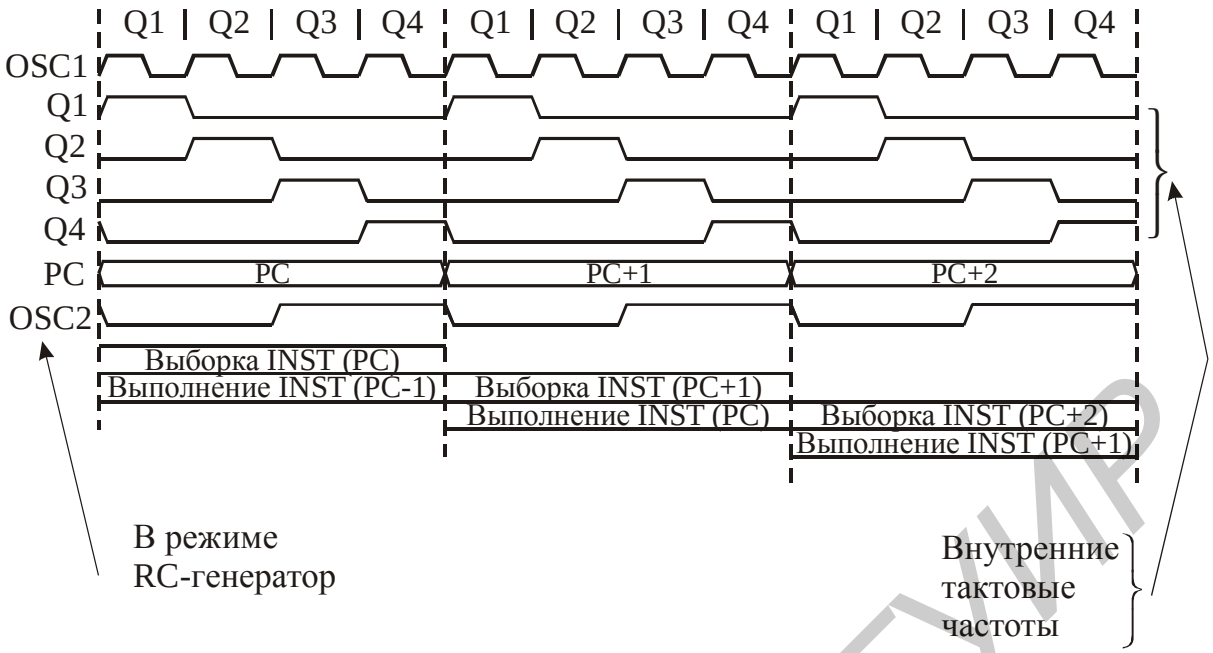


Рис. 1.3. Временные диаграммы тактирования и выполнения команд

Цикл выполнения команды состоит из четырех тактов: Q1...Q4. Выборка команды и ее выполнение совмещены по времени таким образом, что выборка команды занимает один цикл, а выполнение - следующий цикл. Эффективное время выполнения команды составляет один цикл. Если команда изменяет счетчик команд (например команда CALL), то для выполнения этой команды потребуется два цикла. Цикл выборки начинается с увеличения счетчика команд в такте Q1. В цикле выполнения команды выбранная команда защелкивается в регистр команд в такте Q1. В течение тактов Q2, Q3 и Q4 происходит декодирование и выполнение команды. В такте Q2 считывается память данных (чтение операнда), а запись происходит в такте Q4.

1.3. Организация памяти

Внутренняя память в микроконтроллере PIC16F84 состоит из двух частей: памяти программы и памяти данных. Память программы и память данных имеют раздельные шины, поэтому доступ к ним может происходить одновременно. Память данных делится на регистры общего назначения (ОЗУ) и

специальные регистры. Память данных PIC16F84 также содержит электрически перепрограммируемое ПЗУ (ЭППЗУ) данных. Эта память доступна косвенным образом через регистры EEADR, EEDATA, EECON1, EECON2. ЭППЗУ имеет объем 64 байта и адреса 00...3Fh.

1.3.1. Организация памяти программы

Микроконтроллер PIC16F84 имеет 13-разрядный счетчик команд, способный адресовать до 8Кх14 слов памяти программы. В PIC16F84 присутствуют только первые 1Кх14 (0000-03FFh) слов памяти программы, остальные адреса зарезервированы для будущих модификаций. При сбросе процессор запускается с адреса 0000h, вектор прерывания расположен по адресу 0004h. Организация памяти программы и стека показана на рис. 1.4.

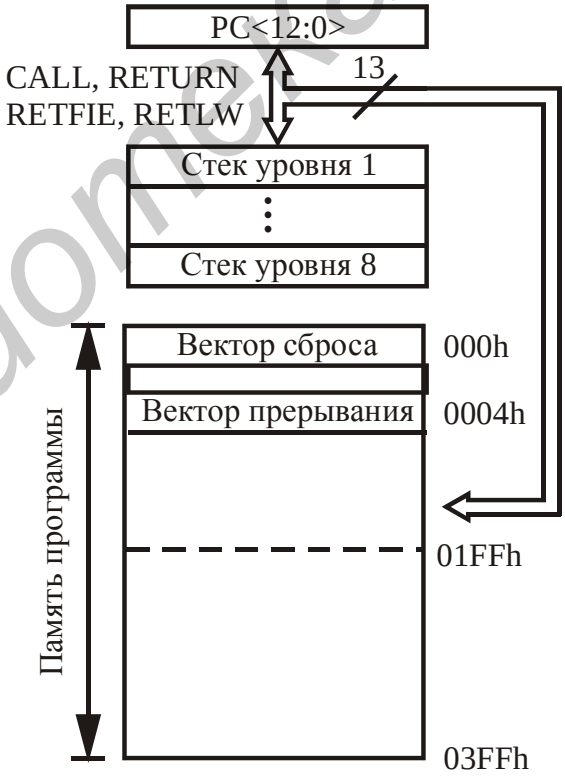


Рис. 1.4. Организация памяти программ и стека

1.3.2. Организация памяти данных

Память данных разделяется на две области. Первая представляет собой регистры специальных функций, вторая - регистры общего назначения (ОЗУ). Специальные регистры включают в себя регистр таймера/счетчика (TMR0), счетчик команд (PC), регистр состояния (STATUS), регистры ввода/ вывода (PORTA и PORTB), регистр косвенной адресации (FSR) и регистры управления встроенным электрически перепрограммируемым ПЗУ (EEADR, EEDATA, EECON1, EECON2). Кроме того, специальные регистры TRISA и TRISB управляют конфигурацией портов ввода-вывода, а OPTION — режимами работы предварительного делителя.

Регистры общего назначения используются для хранения переменных по усмотрению пользователя.

Доступ к части регистров специальных функций, управляющих периферийными устройствами, осуществляется через механизм выбора банков. Биты, управляющие переключением банков, находятся в регистре STATUS.

На рис. 1.5 приведена организация памяти данных PIC16F84.

Все регистры могут быть адресованы прямо или косвенно, с использованием регистра косвенной адресации FSR. Непосредственная адресация поддерживается специальными командами, загружающими данные из памяти программы в рабочий регистр W.

1.4. Регистр состояния STATUS

Регистр состояния содержит флаги состояния АЛУ, состояния контроллера при сбросе и биты выбора страниц памяти данных. Регистр STATUS доступен для любой команды так же, как и любой другой регистр. Однако биты $\overline{TO}$ и $\overline{PD}$ устанавливаются аппаратно и не могут быть изменены программно. Это следует иметь в виду при выполнении команд с

использованием регистра состояния. Например, команда CLRF STATUS обнулит все биты, кроме бит $\overline{TO}$ и $\overline{PD}$, а затем установит бит Z=1. После выполнения этой команды регистр состояния будет иметь значение 000uu100 ($\overline{TO}$ и $\overline{PD}$ не изменились). Поэтому рекомендуется для изменения регистра состояния использовать только команды битовой установки BCF, BSF, а также MOVWF и SWAPF, которые дополнительно не меняют биты регистра состояния.

Рис. 1.5. Организация памяти данных:
(*) не физический регистр; ☐ отсутствует, считывается как ‘0’

Биты IRP и RP1 (STATUS<7:6>) не используются в PIC16F84 и должны быть программно установлены в '0'. Использование этих бит в качестве бит общего назначения не рекомендуется. Биты C и DC при вычитании функционируют как биты заёма и десятичного заёма соответственно. Подробное описание разрядов регистра приведено в табл. 1.2.

Таблица 1.2

Регистр STATUS	Адрес: 03h, 83h	Состояние по включению питания: 0001 1XXX

R/W	R/W	R/W	R	R	R/W	R/W	R/W
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
бит 7							бит 0

C	(Carry /Borrow) Перенос/Заем	Бит является флагом переноса для команд сложения ADDWF, ADDLW и инверсным флагом заема для команд вычитания SUBWF, SUBLW. Бит устанавливается в ‘1’ в командах сложения ADDWF и ADDLW, если в результате операции из старшего разряда произошел перенос (результат больше 0FFh). При выполнении команд сдвига RRF, RLF бит загружается из младшего или старшего бита сдвигаемого регистра соответственно. Вычитание осуществляется путем прибавления дополнительного кода второго операнда. Бит устанавливается в ‘1’ в командах вычитания SUBWF и SUBLW, если при вычитании не произошло заема (результат больше или равен 0)
DC	(Digit carry/Borrow) Десятичный перенос/ Заем	Бит устанавливается в ‘1’ в командах ADDWF и ADDLW, если произошел перенос из бита 3 результата. Бит устанавливается в ‘1’ в командах вычитания SUBWF и SUBLW, если при вычитании не произошло заема из бита 4
Z	(Zero) Ноль	Бит устанавливается в ‘1’, если результат арифметической или логической операции равен нулю
$\overline{PD}$	(Power down) Выключение питания	Бит устанавливается в ‘1’ по включению питания или по команде CLRWDT. Бит сбрасывается в '0' по команде SLEEP
$\overline{TO}$	(Time-out) Тайм-аут	Бит устанавливается в ‘1’ по включению питания, а также командами CLRWDT и SLEEP. Бит сбрасывается в ‘0’ по срабатыванию сторожевого таймера.
RPO RP1	(Register Bank Select) Выбор страницы памяти программы	Биты RP0, RP1 выбирают одну из четырех страниц памяти данных: RP1 RPO : 0 0 : страница 0 (000h...07Fh) 0 1 : страница 1 (080n...0FFh) 1 0 : страница 2 (100h...17Fh) 1 1 : страница 3 (180h...1FFh) Для PIC16F84 используется только бит RP0. Бит RP1 должен быть установлен в ‘0’
IRP	Не используется	Не используется. Бит должен быть установлен в '0'

Примечание. R — только чтение, R/W — чтение и запись.

Если регистр STATUS используется в качестве регистра операнда для команд, изменяющих биты Z, DC и C, то непосредственная запись в эти три бита запрещается. Биты устанавливаются в соответствии с внутренней логикой контроллера.

1.5. Регистр OPTION

Описание разрядов регистра приведено в табл. 1.3.

Таблица 1.3

Регистр: OPTION	Адрес: 81h	Состояние по включению питания: 1111 1111

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
бит 7							бит 0

		PS2	PS1	PS0	Коэффициент деления TMR0	Коэффициент деления WDT
PS0, PS1, PS2	(Prescaler Value) Значение предварительного делителя	0	0	0	1:2	1:1
		0	0	1	1:4	1:2
		0	1	0	1:8	1:4
		0	1	1	1:16	1:8
		1	0	0	1:32	1:16
		1	0	1	1:64	1:32
		1	1	0	1:128	1:64
		1	1	1	1:256	1:128
PSA	(Prescaler Assigment) Назначение предварительного делителя	1 — предварительный делитель включен после WDT 0 — предварительный делитель включен перед TMR0				
T0SE	(TMR0 source edge select) Выбор фронта переключения TMR0	1 —TMR0 увеличивается по перепаду 1/0 на входе T0CKI 0 — TMR0 увеличивается по перепаду 0/1 на входе T0CKI				
T0CS	(TMR0 clock source select) Выбор источника тактирования TMR0	1 — сигнал со входа T0CKI. 0 — внутренняя тактовая частота CLKOUT=CLKIN/4				
INTEDG	(Interrupt Edge select) Выбор фронта переключения RB0/INT	1 — прерывание по перепаду 0/1 на входе RB0/INT 0 — прерывание по перепаду 1/0 на входе RB0/INT				
RBPU	(PORTB Pull-Up enable) Разрешение подтягивающих резисторов на PORTB	1 — подтягивающие резисторы на PORTB отключены 0 — подтягивающие резисторы на PORTB подключены на всех разрядах, запрограммированных как входы				

Примечание. R/W—чтение и запись.

Регистр OPTION доступен для чтения и записи и содержит управляющие биты, которые определяют коэффициент деления и использование предварительного делителя, источник внешних прерываний, а также подтягивающие резисторы на PORTB. Если предварительный делитель установлен на сторожевой таймер WDT (PSA='1'), то таймер TMR0 имеет коэффициент деления 1:1.

1.6. Регистр INTCON

Описание разрядов регистра приведено в табл. 1.4.

Таблица 1.4

Регистр INTCON	Адрес: 0Bh, 8Bh	Состояние по включению питания: 0000 000X

R/W	R/W	R/W	R	R	R/W	R/W	R/W
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
бит 7							бит 0

RBIF	**(RB** port change **I**nterrupt **F**lag bit). Флаг прерывания по изменению состояния порта B	Устанавливается в ‘1’ по изменению состояния хотя бы на одном из входов RB7:RB4.Сбрасывается программно
INTF	(RB0/INT Interrupt Flag bit). Флаг прерывания INT	Устанавливается в ‘1’ по прерыванию INT. Сбрасывается программно
T0IF	(TMR0 Oferflow Interrupt Flag bit). Флаг прерывания по переполнению таймера TMR0	Устанавливается в ‘1’ при переполнении таймера TMR0. Сбрасывается программно
RBIE	(RB port change Interrupt enable bit) Маска прерывания RBIF	RBIE=0: запрещает прерывания от RBIF; RBIE=1: разрешает прерывания от RBIF
INTE	(RB0/INT Interrupt Enable bit). Маска прерывания INT	INTE=0: запрещает прерывания от INTF; INTE=1: разрешает прерывания от INTF
T0IE	(TMR0 Oferflow Interrupt Enable bit). Маска прерывания TMR0	T0IE=0: запрещает прерывания от T0IF; T0IE=1: разрешает прерывания от T0IF
EEIE	(EE write complete Interrupt Enable bit). Маска прерывания от записи в ЭППЗУ	EEIE=0: запрещает прерывания от EEIF; EEIE=1: разрешает прерывания от EEIF
GIE	(Global Interrupt Enable bit). Флаг разрешения прерываний	GIE=0: прерывания запрещены; GIE=1: разрешены все немаскированные прерывания

Примечание. R/W — чтение и запись.

Регистр INTCON доступен для чтения и записи и содержит биты разрешения прерываний и флаги прерываний от различных источников. Флаги прерываний устанавливаются при возникновении условия прерывания независимо от состояния соответствующих бит разрешения прерываний и бита общего разрешения прерываний GIE (INTCON).

1.7. Счетчик команд

Счетчик команд (PC) 13-разрядный. Младшим байтом счетчика команд служит регистр PCL, доступный для чтения и записи. Старшие разряды счетчика команд PC<12:8> не могут быть непосредственно считаны или записаны, а адресуются через регистр PCLATH. Содержимое регистра PCLATH заносится в счетчик команд, когда PC загружается новым значением. Это происходит в командах CALL, GOTO и при записи в PCL. По сигналу RESET в счетчик команд записывается '0'. Загрузка счетчика команд в различных ситуациях показана на рис. 1.6.

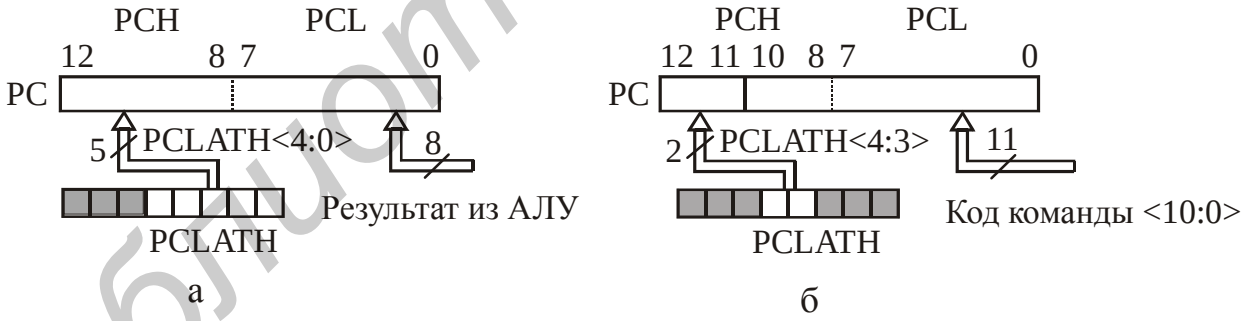


Рис. 1.6. Загрузка счетчика команд:
а - команда, модифицирующая PCL; б - команды GOTO, CALL

Вычисляемые переходы. Вычисляемый переход осуществляется путем добавления смещения к счетчику команд (ADDWF PCL). При осуществлении табличного чтения с использованием вычисляемых переходов следует

внимательно рассматривать случаи, когда таблица пересекает границу, определяемую разрядностью PCL (каждые 256 байт).

1.8. Стек

Микроконтроллер PIC16F84 содержат 8-уровневый аппаратный стек (см. рис. 1.4). Стек не является частью памяти программы или памяти данных, а указатель стека не может быть считан или записан. При выполнении команды CALL или при прерывании в вершину стека заносится содержимое счетчика команд, предварительно увеличенное на единицу. Одновременно старое значение из вершины стека переписывается в стек следующего уровня. Значение из стека извлекается по командам RETURN, RETLW и RETFIE. Содержимое регистра PCLATH не изменяется при операциях со стеком. Стек работает как последовательный буфер. После того как в стек было подряд занесено восемь значений счетчика команд, девятое значение переписывает значение, которое было занесено первым. Десятое значение перепишет значение, занесенное вторым, и т. д. При девятом и последующих последовательных извлечениях из стека будет считано то же значение, что и при восьмом извлечении, а именно значение, которое было занесено в стек первым.

Биты, указывающие на переполнение и пустоту стека, отсутствуют.

1.9. Косвенная адресация данных

Регистр INDF физически не существует и используется совместно с регистром FSR для косвенной адресации данных. Любая команда, использующая регистр INDF, в действительности обращается к регистру данных, адресуемому регистром FSR. Например, в команде ADDWF INDF к регистру W будет прибавлено содержимое регистра, адрес которого находится

в FSR. Чтение косвенным образом самого регистра INDF даст результат 00h (т.е. FSR=0). Косвенная запись в регистр INDF аналогична, хотя биты состояния могут быть изменены.

Программа, использующая косвенную адресацию для записи нулей в 16 последовательно расположенных ячеек ОЗУ с адреса 20h до адреса 2Fh, приведена ниже.

```
        MOWLW   0X20    ;Заносим в FSR адрес первой ячейки памяти.
        MOVWF   FSR     ;
NEXT    CLRF    INDF    ;Это начало цикла очистки ячеек . Команда ,
                        ;адресующаяся к регистру INDF,в результате
                        ;по очереди очищает все ячейки.
        INCF    FSR     ;Следующий адрес.
                        ;Когда пройдут все 16 адресов (10h),в регистре
                        ;FSR окажется установленным бит 4
                        ;(0011 0000=30h) и цикл закончится.
        BTFSS   FSR,4   ;Всё закончено ?
        GOTO    NEXT    ;Нет, следующая ячейка.
        CONT. . .       ;Продолжение программы.
```

1.10. Порты ввода/вывода

Микроконтроллеры PIC16F84 имеют два порта ввода/вывода, PORTA и PORTB. Некоторые разряды портов имеют дополнительные функции.

Программа может считывать и записывать данные в регистры ввода-вывода аналогично регистрам общего назначения. При чтении всегда считывается действительное состояние выводов, независимо от того, запрограммированы отдельные биты как входы или как выходы. После сброса все разряды программируются как входы (выходы находятся в высокоимпедансном состоянии), поскольку регистры управления портами TRISA и TRISB устанавливаются в '1'. Разряд порта ввода-вывода определяется как выход, если соответствующий бит в регистре управления портом установлен в '0'.

1.10.1. PORTA

Регистр ввода-вывода PORTA имеет разрядность 5 бит. Старшие 3 бита физически отсутствуют и считываются как '0'. Разряд RA4 имеет вход с триггером Шмитта и выход с открытым стоком. Все остальные разряды PORTA имеют ТТЛ-входные уровни и КМОП-выходы. Разряд RA4 объединен с входом таймера TMR0.

Описание выводов регистра PORTA приведено в табл. 1.1. Функциональная организация разрядов RA0…RA3 регистра показана на рис. 1.7, а разряда RA4 – на рис.1.8.

Ниже приведем пример программы инициализации (настройки) PORTA.

```
CLRF    PORTA          ;Очистка (сброс в «0») выходных защёлок PORTA.
BSF     STATUS,RP0     ;Выбор банка 1.
MOVLW   0x 0F          ;Значение константы для выбора режимов работы
                       ;разрядов.
MOVWF   TRISA          ;Установить RA<3:0> как входы и RA4 как
                       ;выход.
BCF     STATUS,RP0     ;Выбор банка 0.
```

1.10.2. PORTB

Регистр ввода/вывода PORTB 8-разрядный. Все разряды PORTB имеют внутренние подтягивающие резисторы, которые могут быть включены установкой в '0' бита RBPU (OPTION). Подтягивающие резисторы автоматически отключаются, если соответствующий разряд программируется как выход. По включении питания подтягивающие резисторы отключаются.

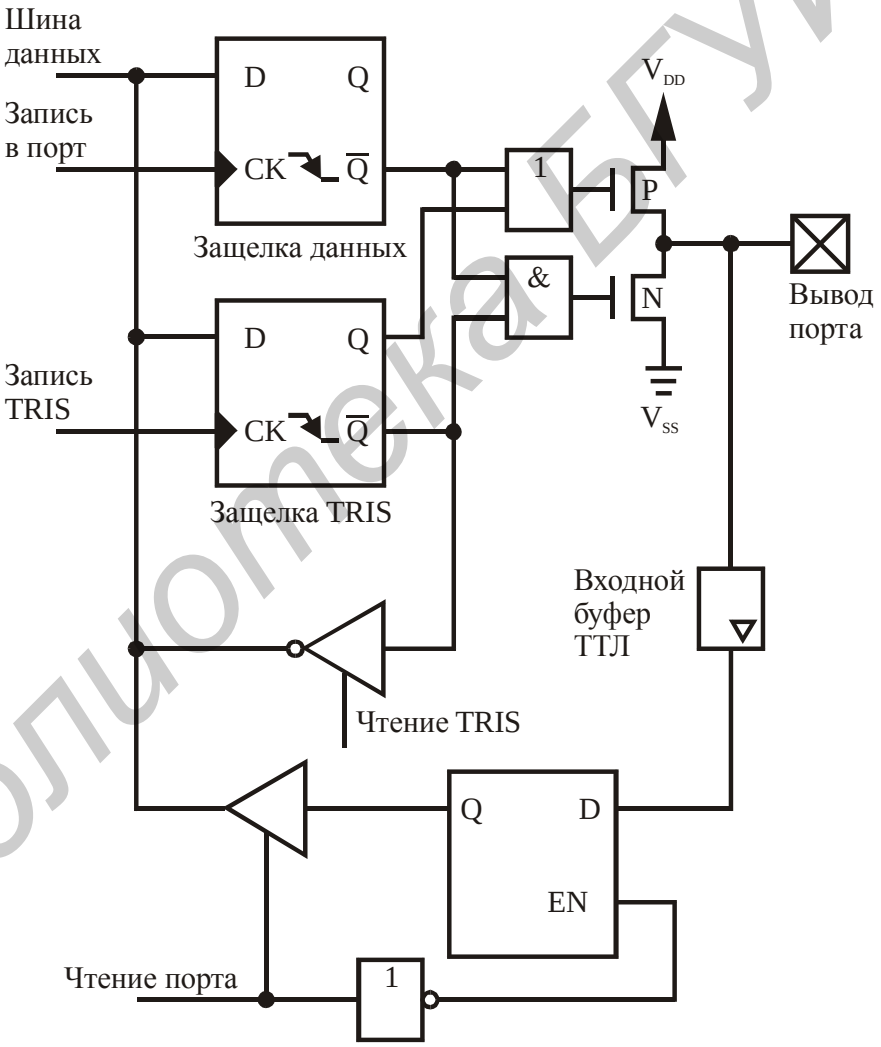
Функциональная организация выводов RB7…RB4 порта B показана на рис. 1.9, а выводов RB3…RB0 – на рис.1.10.

Имеется возможность прерывания по изменению состояния четырех разрядов PORTB (выводы RB7…RB4). Прерывание может возникнуть только от тех разрядов PORTB<7:4>, которые запрограммированы как входы, выходы не включаются в процедуру сравнения. Текущее состояние разрядов

PORTB<7:4>, запрограммированных как входы, сравнивается с состоянием, защелкнутым в регистр PORTB при последнем считывании. При несовпадении возникает прерывание по изменению состояния. Это прерывание может вывести микроконтроллер из режима пониженного энергопотребления SLEEP. Для сброса прерывания в подпрограмме обработки необходимо выполнить следующие действия:

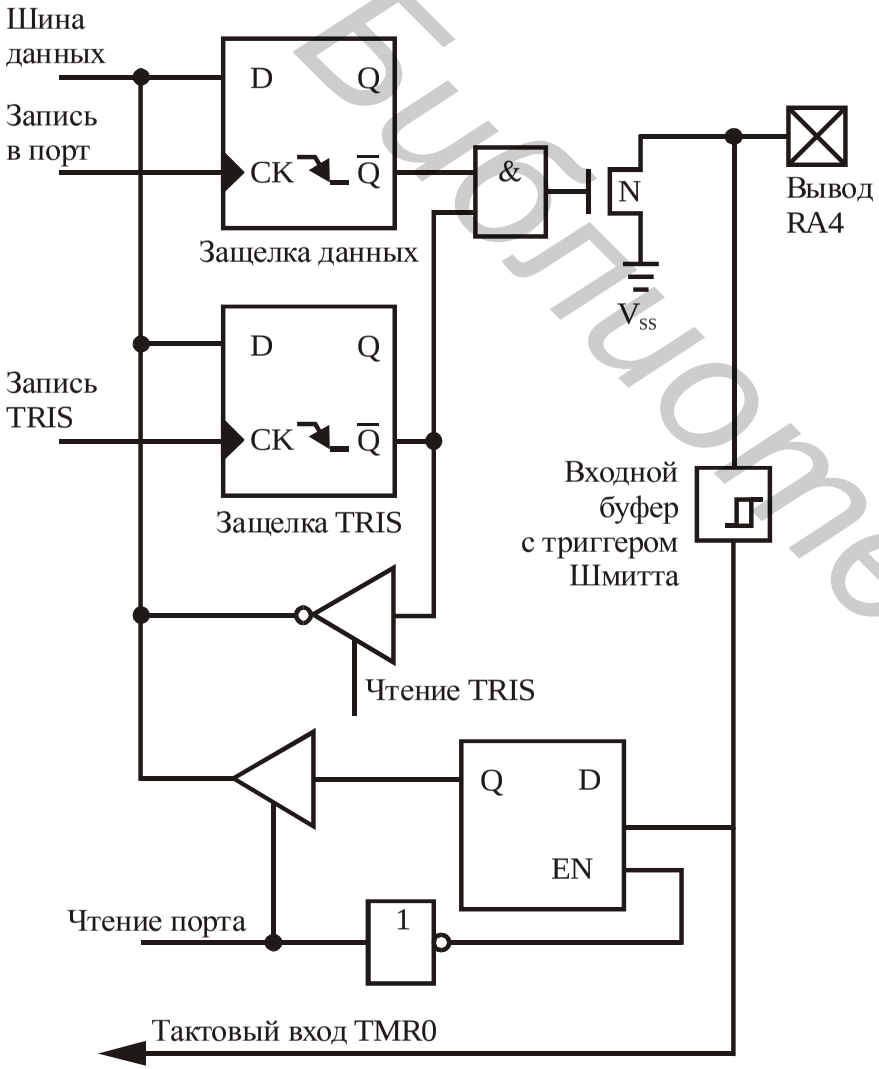
1) считать PORTB (это сбросит условие несовпадения);

2) сбросить флаг RBIF.



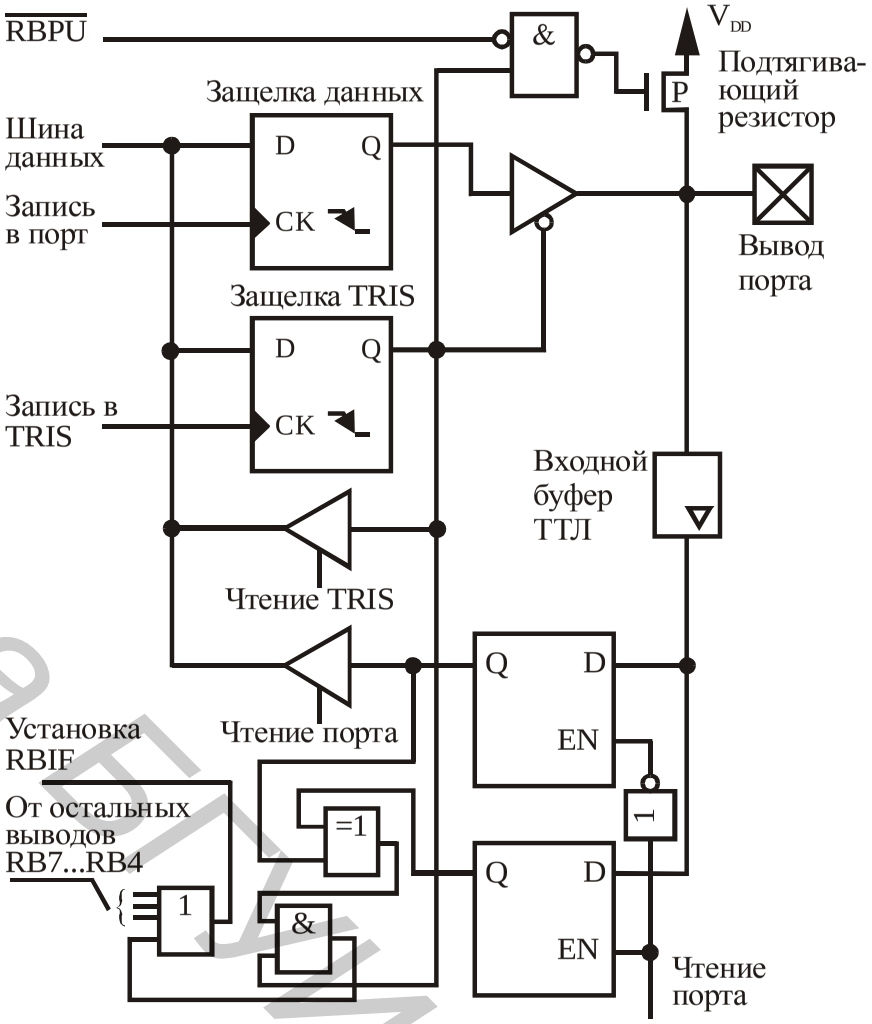
Примечание. Все выводы имеют защитные диоды на V_{DD} и V_{SS}.

Рис. 1.7. Функциональная организация выводов RA0...RA3 регистра PORTA



Примечание. Вывод имеет защитный диод только на V_{SS}.

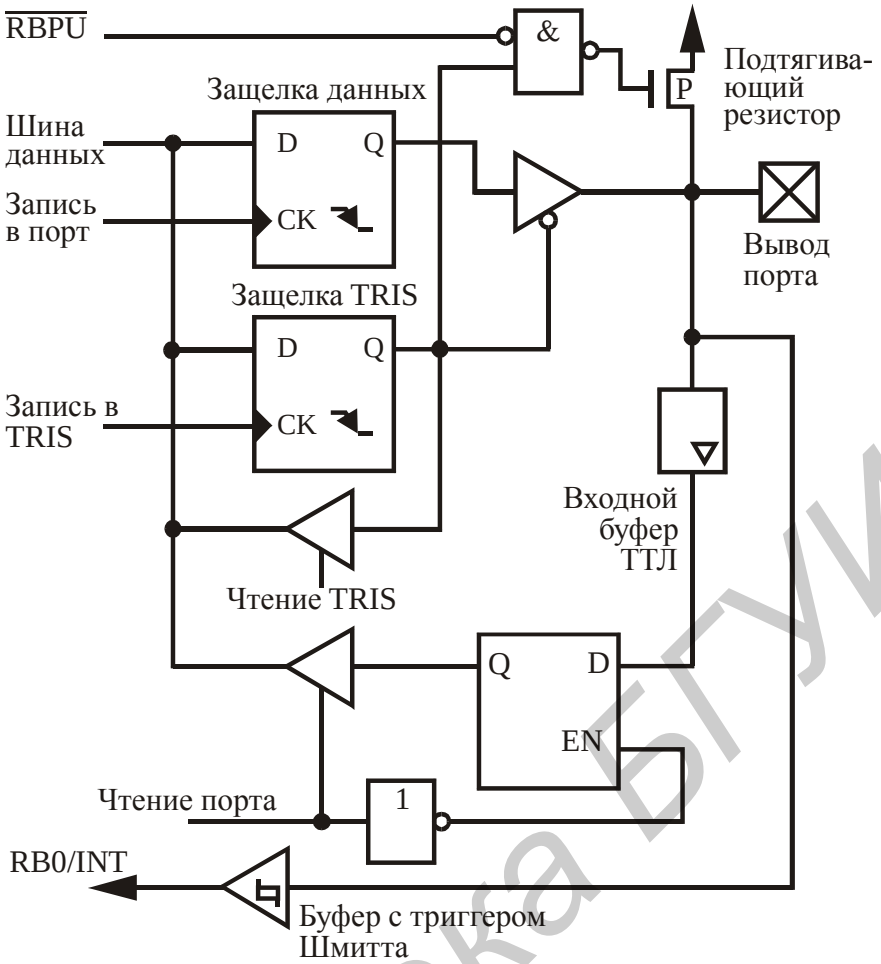
Рис. 1.8. Функциональная организация вывода RA4 регистра PORTA



Примечания:

1. Бит TRISB='1' разрешает подтягивающий резистор, если в регистре OPTION $\overline{RBPU}$='0'.
2. Все выводы имеют защитные диоды на V_{DD} и V_{SS}.

Рис. 1.9. Функциональная организация выводов RB7...RB4



Примечания:

1. Бит TRISB='1' разрешает подтягивающий резистор, если в регистре OPTION $\overline{\text{RBPU}}$='0'.
2. Все выводы имеют защитные диоды на V_{DD} и V_{SS}.

Рис. 1.10. Функциональная организация выводов RB3...RB0

Прерывание по несовпадению совместно с программно управляемыми подтягивающими резисторами позволяет легко реализовать интерфейс клавиатуры и обеспечить выход из режима пониженного энергопотребления по нажатии клавиши. Для возникновения прерывания по изменению состояния минимальная деятельность импульса должна быть не менее длительности цикла команды.

Ниже приведем пример инициализации (настройки) PORTB.

```
CLRF    PORTB       ;Обнуление выходных регистров PORTB.
BSF     STATUS,RP0  ;Выбор банка 1.
```

```
MOVLW   0xCF        ;Значение для задания направления.
MOVWF   TRISB       ;Установить RB<3:0> как входы, RB<5:4>
                    ; как выходы и RB<7:6> как входы
```

1.11. Особенности программирования портов

1.11.1 Организация двунаправленных портов

Некоторые команды выполняются в режиме чтение — модификация — запись.

Например, команды BCF и BSF считывают содержимое порта целиком, модифицируют один бит и выводят результат обратно. При использовании этих команд с портами, в которых некоторые разряды запрограммированы как входы, а некоторые как выходы, необходима осторожность. Например, команда BSF для пятого бита регистра PORTB сначала считывает все восемь бит, затем выполняется установка пятого бита и новое значение байта целиком записывается в выходную защелку. Если другой бит регистра PORTB используется в качестве двунаправленного входа/выхода (скажем, бит 0) и в данный момент он определен как вход, входной сигнал на этом выводе будет считан и записан обратно в выходную защелку этого же вывода, затирая ее предыдущее состояние. До тех пор пока этот вывод остается в режиме входа, никаких проблем не возникает. Однако если позднее линия 0 переключится в режим выхода, ее состояние будет неопределенным. Команда считывания порта считывает состояние вывода, а не выходных регистров. Например, если разряд порта запрограммирован как выход и установлен в '1', но внешняя схема поддерживает низкий уровень на выводе, порт будет считываться как '0'. На вывод, работающий в режиме выхода, не должны подключаться внешние нагрузки по схеме "монтажное И" либо "монтажное ИЛИ". Возникающие при этом большие токи могут повредить кристалл.

1.11.2 Обращение к портам ввода/вывода

Запись в порт вывода происходит в конце цикла выполнения команды. При чтении данные должны быть стабильны в начале цикла выполнения команды. Надо быть внимательным при операциях чтения, следующих сразу же за записью в тот же порт. Здесь надо учитывать инерционность установления напряжения на выводах. Может потребоваться программная задержка, чтобы напряжение на выводе успело стабилизироваться до начала исполнения следующей команды чтения. Время установления напряжения на выводе зависит от подключенной к нему нагрузки и может меняться в широких пределах.

Ниже рассмотрен пример выполнения операции чтение — модификация — запись с портом ввода/вывода.

```
;Начальные установки порта: PORTB<7…4> - входы,
;                           PORTB<3…0> - выходы.
:Разряды PORTB<7…6> имеют внешние нагрузки.
;                                   Защелка PORTB   Выводы PORTB
        BCF    PORTB,7              ;01pp pppp          11pp pppp
        BCF    PORTB,6              ;10pp pppp          11pp pppp
        BSF    STATUS,RP0           ;
        BCF    TRISB,7              ;10pp pppp          11pp pppp
        BCF    TRISB,6              ;10pp pppp          10pp pppp
; Примечание: пользователь мог бы ожидать, что результирующее значение будет
; 00pp pppp.
; Однако вторая команда BCF вызывает защелкивание в RB7 высокого уровня в
; соответствии с состоянием вывода RB7.
```

1.12. Модуль таймера

Модуль таймера имеет следующие особенности:

• 8-разрядный таймер/счетчик, доступен по чтению и записи;

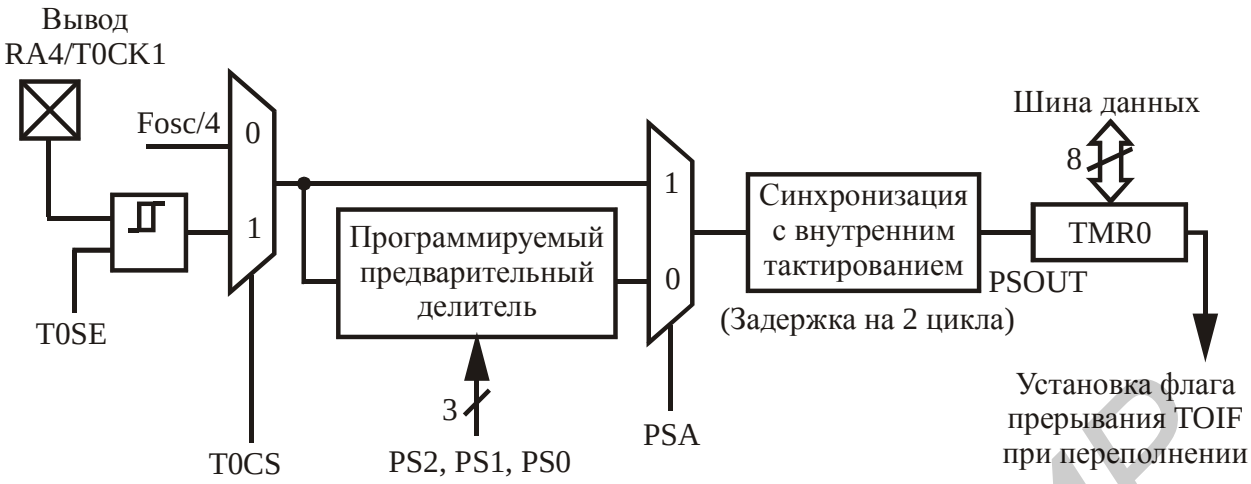
• 8-разрядный программируемый предварительный делитель;

• внутреннее или внешнее тактирование;

• прерывание по переполнению счетчика (переход от 0FFh к 00h);

• выбор фронта тактирующего импульса при внешнем тактировании.

Упрощенная структурная схема модуля таймера приведена на рис. 1.11.



Примечания:
1. Биты T0CS, T0SE, PSA, PS2, PS1 и PS0 находятся в регистре OPTION.
2. Предварительный делитель используется совместно со сторожевым таймером.

Рис. 1.11. Структурная схема таймера

Режим **таймера** выбирается установкой в '0' бита T0CS (OPTION<5>). В режиме таймера TMR0 увеличивается в каждом командном цикле (в отсутствии предварительного делителя). Если происходит запись в TMR0, то увеличение счетчика задерживается на два последующих цикла выполнения команды. Запись в TMR0 должна вестись с учетом этой задержки. При необходимости проверки регистра TMR0 на ноль без влияния на процесс счета рекомендуется пользоваться командой **MOVF** TMR0,W.

Режим **счетчика** выбирается установкой в '1' бита T0CS (OPTION<5>). В этом режиме TMR0 увеличивается по каждому перепаду 1/0 или 0/1 на выводе T0CKI. Перепад, увеличивающий значение TMR0, выбирается битом выбора фронта переключения T0SE (OPTION<4>). Установка этого бита в '0' вызывает увеличение TMR0 по перепаду 0/1.

Предварительный делитель может использоваться модулем сторожевого таймера WDT или модулем таймера. Подключение предварительного делителя задается битом PSA (OPTION<3>). Установка бита PSA в '1' подключает предварительный делитель к модулю WDT и устанавливает коэффициент деления для TMR0 1:1. Установка бита PSA в '0' подключает предварительный делитель к модулю таймера. Коэффициент деления предварительного делителя

может быть установлен битами PS0-PS2 регистра OPTION (см. табл. 1.3). Сам предварительный делитель недоступен для чтения и записи.

1.12.1. Прерывание от таймера

Прерывание от TMR0 вырабатывается при переполнении счетчика (переходе от 0FFh к 00h). При переполнении устанавливается в '1' бит T0IF (INTCON<2>). Прерывание может быть замаскировано установкой в '0' бита T0IE (INTCON<5>). Бит T0IF должен быть сброшен в '0' в процедуре обработки прерывания от TMR0 до того, как прерывания снова будут разрешены. Прерывание от TMR0 не может вывести микроконтроллер из режима пониженного энергопотребления SLEEP, поскольку в режиме SLEEP таймер TMR0 выключен.

1.12.2. Использование TMR0 с внешним сигналом

Если для тактирования TMR0 используется внешний сигнал, то он должен удовлетворять определенным требованиям для синхронизации с внутренней тактовой частотой. Кроме того, между перепадом на выводе T0CKI и реальным увеличением счетчика TMR0 есть некоторая задержка.

Если предварительный делитель не используется, внешний тактовый сигнал на входе T0CKI должен сохранять как высокий, так и низкий уровень в течение не менее двух периодов тактового генератора.

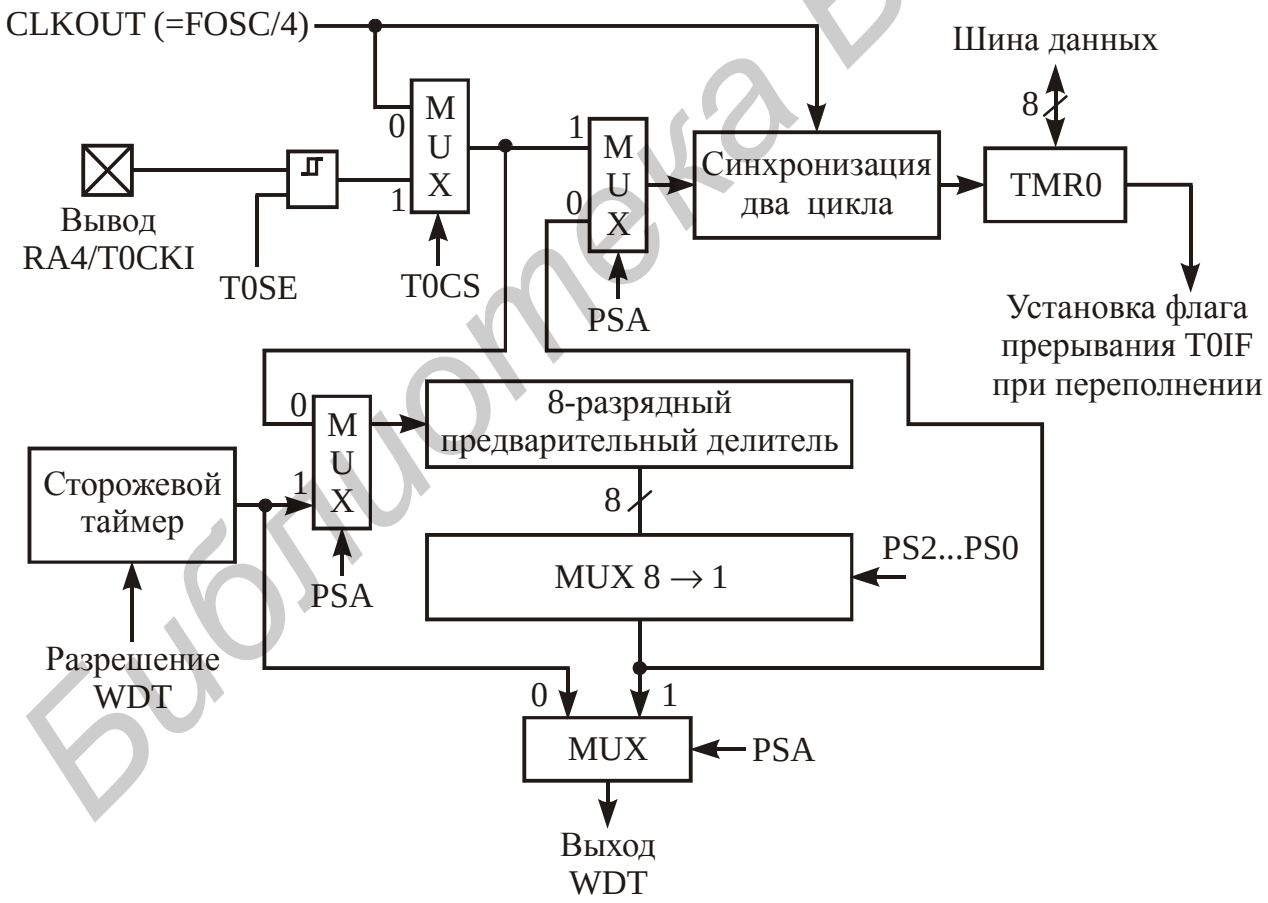
Когда используется предварительный делитель, входной сигнал TMR0 делится асинхронным счетчиком предварительного делителя, поэтому выходной сигнал делителя является симметричным. Период сигнала на входе TMR0 должен быть не менее четырех периодов тактового генератора. Сигнал же на входе T0CKI должен иметь высокие и низкие уровни длительностями не менее 10 нс.

Так как выход предварительного делителя синхронизирован с внутренней тактовой частотой, то возможна небольшая задержка между перепадом сигнала на выводе T0CKI и моментом увеличения содержимого TMR0.

1.12.3. Предварительный делитель

Встроенный 8-разрядный счетчик может использоваться как предварительный делитель для TMR0 или как дополнительный делитель для сторожевого таймера WDT. Необходимо учесть, что делитель может быть использован либо с TMR0, либо со сторожевым таймером WDT, но не одновременно.

Схема использования предварительного делителя отражена на рис 1.12.



Примечание. Биты T0CS, T0SE, PSA, PS2...PS0 – биты регистра OPTION <6...0>.

Рис. 1.12. Схема использования предварительного делителя

Биты PSA и PS0-PS2 в регистре OPTION<3:0> задают режим использования предварительного делителя и его коэффициент деления.

Когда предварительный делитель используется с TMR0, все команды, производящие запись в регистр TMR0 (например, **CLRF** TMR0, **MOVWF** TMR0, **BSF** TMR0,b и т.д.), очищают предварительный делитель.

Когда предварительный делитель используется со сторожевым таймером WDT, команда **CLRWDT** очищает предварительный делитель одновременно со сбросом сторожевого таймера WDT. Предварительный делитель не может быть считан или записан программно. По сбросу предварительный делитель содержит все '0'.

Назначение предварительного делителя задается программно и может быть изменено в процессе выполнения программы. Чтобы избежать непредусмотренного сброса контроллера, при **переключении** предварительного делителя **с TMR0 на WDT** должна быть выполнена такая последовательность команд:

```
BCF     STATUS,RP0      ;Установка банка 0.
CLRF    TMR0            ;Сброс TMR0.
BSF     STATUS,RP0      ; Установка банка 1.
CLRWDT                  ;Сброс WDT и предварительного делителя.
MOVLW   b'xxxx1xxx'     ;Укажите новое значение предварительного
MOVWF   OPTION          ; делителя.
BCF     STATUS,RP0      ; Установка банка 0.
```

Для **переключения** предварительного делителя **с WDT на TMR0** должна быть выполнена последовательность команд:

```
CLRWDT                  ;Сброс WDT и предварительного делителя.
BSF     STATUS,RP0;Установка банка 1.
MOVLW   b'xxxx0xxx'  ;Указать новое значение предварительного делителя,
                     ;источник тактирования и фронт переключения TMR0.
MOVWF   OPTION       ;
BCF     STATUS,RP0;Установка банка 0.
```

Эта последовательность должна быть выполнена даже в том случае, если сторожевой таймер WDT запрещен.

1.13. Специальные функции

Микроконтроллер PIC16F84 имеет набор специальных функций, предназначенных для расширения возможностей системы, минимизации стоимости, исключения навесных компонентов, обеспечения минимального энергопотребления и защиты кода от считывания. В нём реализованы следующие специальные функции:

- выбор типа генератора;
- сброс:
  - схема сброса по включению питания (POR);
  - таймер сброса (PWRT);
  - таймер запуска генератора (OST);
- прерывания;
- сторожевой таймер (WDT);
- режим пониженного энергопотребления (SLEEP);
- защита кода от считывания;
- биты идентификации;
- внутрисхемное программирование.

Микроконтроллер имеет сторожевой таймер WDT, который может быть выключен только через бит конфигурации WDTE. Для повышения надежности он работает от собственного RC-генератора. Сторожевой таймер предназначен для формирования сигнала сброса в случае «зависания» рабочей программы.

Имеются также два таймера, формирующие необходимые задержки при включении питания. Таймер запуска генератора OST сохраняет микроконтроллер в состоянии сброса до стабилизации работы генератора. Таймер сброса PWRT формирует фиксированную задержку 18 мс после включения питания. Присутствие этих таймеров позволяет во многих применениях отказаться от схемы внешнего сброса.

Режим пониженного энергопотребления предназначен для обеспечения очень малого тока потребления в ожидании (менее 1 мкА при выключенном сторожевом таймере). Вход в режим SLEEP осуществляется программно по соответствующей команде. Выход из режима SLEEP возможен по внешнему сигналу сброса или по окончании выдержки сторожевого таймера.

Возможность выбора типа генератора позволяет эффективно использовать микроконтроллер в различных приложениях. Использование RC-генератора позволяет уменьшить стоимость системы, а LP-генератор сокращает энергопотребление.

1.13.1. Биты конфигурации

Описание разрядов слова конфигурации приведено в табл. 1.5.

Таблица 1.5

Регистр:CONFIG				Адрес: 2007h									
P	P	P	P	P	P	P	P	P	P	P	P	P	P
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRTE	WDTE	FOSC1	FOSC0
бит 13													бит 0

FOSC1 FOSC0	(OSC Selection) Выбор типа генератора	FOSC1 FOSC0: 1 1 RC-генератор 1 0 HS-генератор 0 1 XT-генератор 0 0 LP-генератор
WDTE	(Watchdog Timer Enable) Разрешение сторожевого таймера	1 — сторожевой таймер разрешен. 0 — сторожевой таймер запрещен
PWRTE	(Power-up Timer Enable) Разрешение таймера включения питания	1 — таймер включения питания запрещен. 0 — таймер включения питания разрешен
CP	(Program Memory Code Protection) Защита от считывания памяти программы	1 — защита от считывания выключена. 0 — защита от считывания включена. Для защиты от считывания памяти программы должны быть установлены все биты CP

Примечание. P - программируемый бит.

Микроконтроллер имеет набор бит конфигурации, устанавливающихся на этапе программирования кристалла. Эти биты отображаются на специальный

адрес 2007h. Адрес 2007h находится за пределами памяти программы в диапазоне адресов конфигурации/тестирования (2000h-3FFFh), который доступен только в режиме программирования.

Биты FOSC1, FOSC0 определяют тип генератора, бит WDTE разрешает использование сторожевого таймера WDT, бит PWRTE разрешает работу таймера включения питания, а бит CP предназначен для защиты от считывания памяти программы.

1.13.2. Типы генераторов

В микроконтроллере предусмотрены четыре типа генераторов. Пользователь должен запрограммировать два конфигурационных бита (FOSC1 и FOSC0) для выбора одного из четырех режимов:

- LP (Low Power) — микромощный резонатор;
- XT (Crystal) — керамический или кварцевый резонатор;
- HS (High Speed) — высокочастотный кварцевый резонатор;
- RC (Resistor/Capacitor) — RC-цепочка.

1.13.3. Кварцевый генератор

В режимах XT, LP и HS к выводам OSC1/CLKIN и OSC2/CLKOUT подключается кварцевый или керамический резонатор (рис. 1.13). Схема генератора предусматривает использование резонаторов с параллельным резонансом. Использование резонаторов с последовательным резонансом может привести к возбуждению резонатора на частоте, выходящей за пределы параметров резонатора. Резистор R_S может использоваться в режимах HS и XT для защиты от перегрузки резонаторов с низким уровнем возбуждения. Рекомендуемый диапазон значений R_S от 100 Ом до 1 кОм. В режимах XT, LP и

HS генератор PIC16F84 может также тактироваться от внешнего источника, подключаемого к выводу OSC1/CLKIN (рис. 1.14).

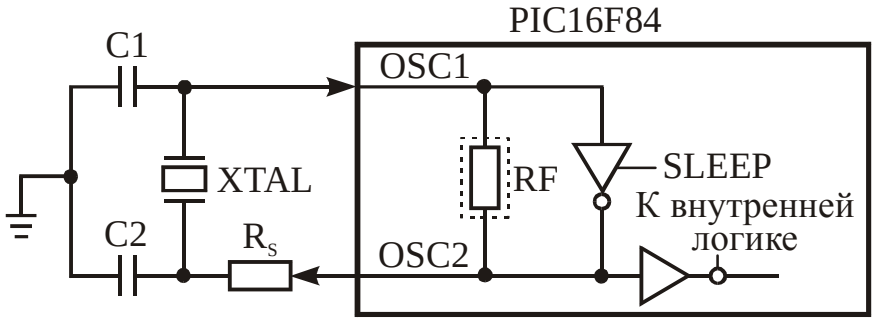


Рис. 1.13. Схема подключения кварцевого или керамического резонатора

В табл. 1.6 приведены рекомендуемые изготовителем емкости конденсаторов для схемы синхронизации с кварцевым или керамическим резонаторами.

Таблица 1.6

Тип генератора	Рабочая частота, кГц	Емкость конденсаторов (C1=C2), пФ
LP	32 200	68...100 15...33
XT	100 2 000 4 000	100...150 15...33 15...33
HS	4 000 10 000	15...33 15...33

1.13.4. RC-генератор

Когда не предъявляются высокие требования к точности и стабильности частоты генератора, использование RC-генератора позволяет дополнительно уменьшить стоимость системы.

Частота RC-генератора зависит от питающего напряжения, значений резистора R_{EXT}, конденсатора C_{EXT}, рабочей температуры и незначительно изменяется от разброса характеристик кристаллов. На частоту генерации при малых значениях C_{EXT} также влияет собственная емкость корпуса кристалла и емкость монтажа. Кроме того, нужно учитывать также температурный дрейф

резистора R_{EXT} и конденсатора C_{EXT}. На рис. 1.15 приведена схема включения RC-генератора.

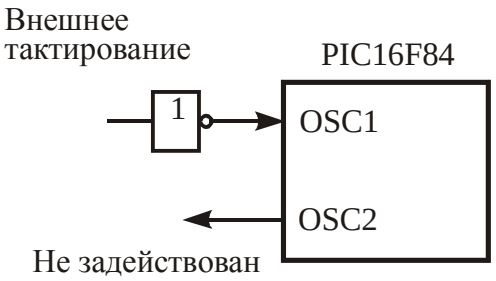


Рис. 1.14. Схема внешнего тактирования

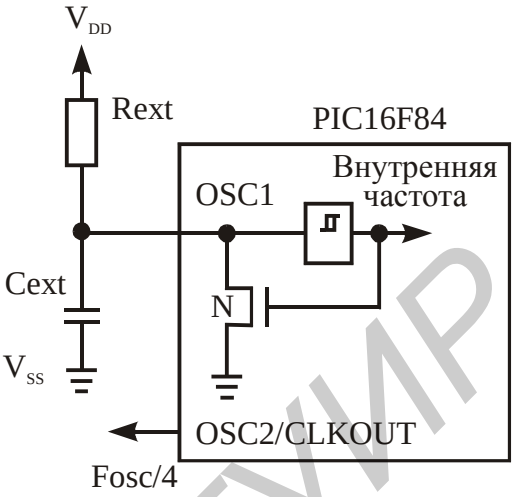


Рис. 1.15. Схема включения RC-генератора

При значениях R_{EXT} ниже 2,2 кОм генератор может работать нестабильно или не запускаться. При очень больших значениях R_{EXT} (например 1 МОм) генератор становится чувствительным к помехам, утечкам и влажности. Рекомендуемый диапазон значений R_{EXT} от 5 до 100 кОм. Хотя генератор работоспособен и при отсутствии внешнего конденсатора (C_{EXT}=0), для увеличения стабильности работы рекомендуется использовать конденсатор емкостью более 20 пФ. При малой емкости C_{EXT} или вообще без него частота генератора сильно зависит от монтажных емкостей. Разброс будет тем больше, чем больше величина резистора R_{EXT} (так как влияние токов утечки на частоту RC-генератора сильнее при больших значениях R_{EXT}) и чем меньше величина емкости C_{EXT} (так как в этом случае сильнее проявляется влияние монтажных емкостей).

На выводе OSC2/CLKOUT в режиме RC присутствует сигнал с частотой генератора, деленной на четыре, который может быть использован для синхронизации других схем.

При использовании режима RC не допускается тактировать микроконтроллеры от внешней логики во избежание повреждения входных цепей генератора.

В табл. 1.7 приведены ориентировочные значения тактовой частоты для различных соотношений параметров RC-цепочки.

Таблица 1.7

C1, пФ	R1, кОм	Частота, МГц
20	3,3	4,6
	5,1	3,9
	10	2,3
	100	0,25
100	3,3	1,5
	5,1	1,1
	10	0,62
	100	0,09
300	3,3	0,52
	5,1	0,42
	10	0,27
	100	0,025

1.13.5. Сброс

В микроконтроллере PIC16F84 реализованы следующие способы сброса:

• сброс по включению питания (POR);

• сброс по входу MCLR при обычной работе;

• сброс по входу MCLR в режиме пониженного энергопотребления SLEEP;

• сброс по сторожевому таймеру WDT при обычной работе;

• сброс по сторожевому таймеру WDT в режиме пониженного энергопотребления SLEEP.

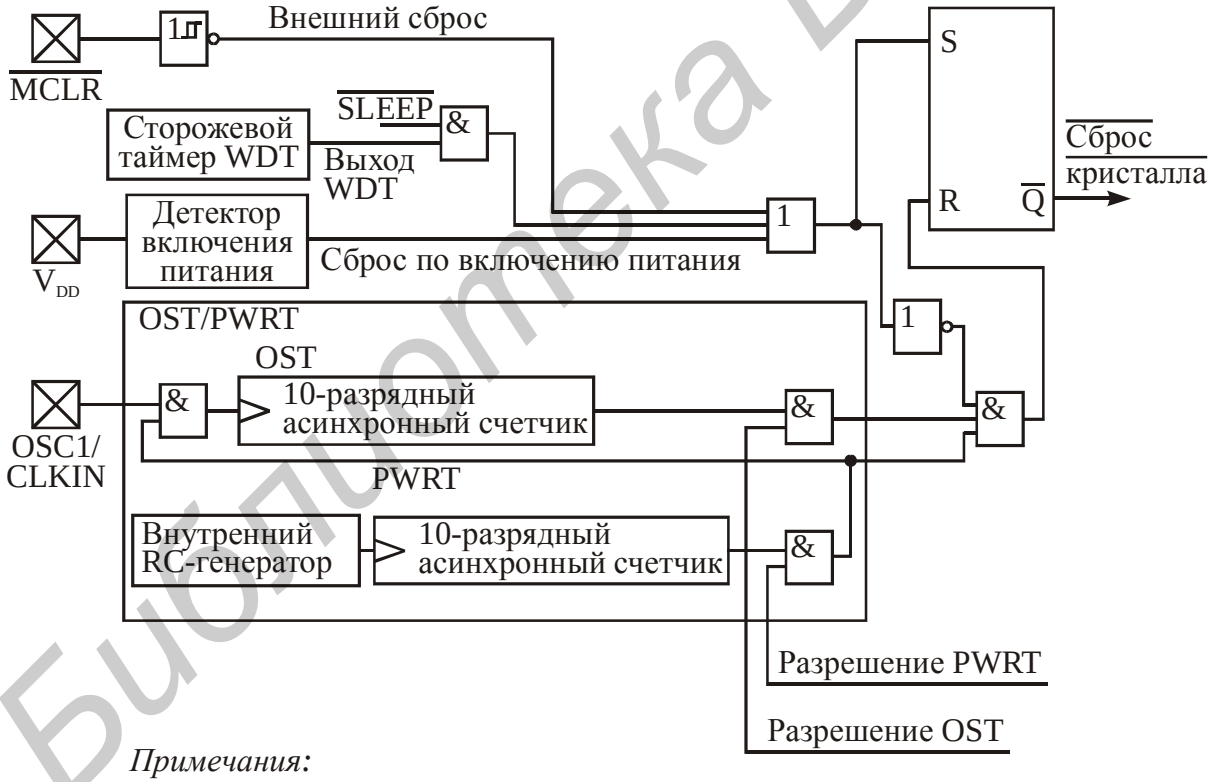
При сбросе некоторые регистры не изменяют свое состояние. При включении питания они имеют неопределенное значение, в остальных случаях их состояние не изменяется. Большинство других регистров устанавливаются в определенное состояние в случаях сброса по включению питания, по входу MCLR или по сторожевому таймеру при обычной работе. Они не изменяются при сбросе по сторожевому таймеру из режима пониженного энергопотребления SLEEP, поскольку эта ситуация рассматривается как продолжение обычной работы.

Биты TO и PD (STATUS<4:3>)устанавливаются в зависимости от причины сброса (табл. 1.8).

Таблица 1.8

TO	PD	Состояние
1	1	Сброс по включению питания
0	X	Бит TO установлен в процессе сброса по включению питания
X	0	Бит PD установлен в процессе сброса по включению питания
0	1	Сброс по WDT из рабочего режима
0	0	Сброс по WDT из режима SLEEP
1	1	Сброс по MCLR из рабочего режима
1	0	Сброс по MCLR из режима SLEEP

На рис. 1.16 приведена структурная схема узла сброса. На входе MCLR имеется фильтр, защищающий от небольших импульсов помехи.



Примечания:

1. Таймер PWRT разрешен при включении питания и активном значении соответствующего бита в слове конфигурации.
2. Таймер OST разрешен при включении питания или при выходе из режима пониженного энергопотребления, если используется XT-, LP- или HS-генератор.

Рис. 1.16. Структурная схема узла сброса

Сброс по включению питания (POR). Микроконтроллер содержит специальную схему внутреннего аппаратного сброса по включению питания POR. Для использования этой схемы достаточно соединить выводы MCLR и V_{DD}. При этом отсутствует необходимость во внешних RC компонентах, обычно используемых в схемах сброса. Внутренний импульс сброса по включению питания формируется при достижении питающим напряжением минимального уровня (1,2-1,7 В). Работа схемы POR гарантируется при скорости установления напряжения питания не менее 0,05 В/мс и начальном напряжении V_{DD} не более 0,2 В. Схема сброса POR не вырабатывает сброса при выключении питания.

Таймер сброса PWRT предназначен для формирования выдержки времени (типовое значение 72 мс) при включении питания микроконтроллера и работает от отдельного внутреннего RC-генератора. В течение этого времени микроконтроллер находится в состоянии сброса. Эта выдержка времени позволяет напряжению питания V_{DD} установиться до нормального значения. Бит конфигурации PWRTE может разрешить или запретить работу с PWRT.

Таймер запуска генератора OST формирует задержку в 1024 импульса генератора со входа OSC1 после окончания выдержки времени PWRT. Эта задержка предназначена для запуска и стабилизации работы кварцевого резонатора. Задержка OST формируется только в режимах XT, LP и HS и только при включении питания и выходе из режима пониженного энергопотребления SLEEP.

Последовательность включения питания. При включении питания сначала формируется сигнал внутреннего сброса POR, затем выдержка времени PWRT, потом включается таймер запуска генератора OST. Общее время запуска зависит от конфигурации генератора и состояния бита PWRTE (табл. 1.9). Например, в режиме RC с запрещенным PWRT задержка будет отсутствовать. Все выдержки времени отсчитываются от внутреннего сигнала POR. Если сигнал MCLR находился в низком уровне достаточно долго после

включения питания, чтобы выдержки времени успели закончиться, то при переключении MCLR в высокий уровень выполнение программы начнется немедленно. Этот режим удобно использовать для тестовых целей или для синхронизации более чем одного микроконтроллера для одновременного запуска.

Таблица 1.9

Тип генератора	Включение питания		Выход из режима SLIEP
	PWRT разрешен	PWRT запрещен	
XT,HS,LP	72мс+ 1024T_{OSC}	1024 T_{OSC}	1024 T_{OSC}
RC	72 мс	–	–

Примечание. T_{OSC} - период тактового генератора.

В табл. 1.10 описаны условия сброса для специальных регистров, а в табл. 1.11 приведены условия сброса для всех остальных регистров.

Таблица 1.10

Условие	Счетчик команд	Регистр STATUS
Включение питания	000h	0001 1xxx
Сброс по MCLR из рабочего режима	000h	0001 luuu
Сброс по MCLR из режима SLEEP	000h	0001 0uuu
Сброс по WDT из рабочего режима	000h	0000 luuu
Сброс по WDT из режима SLEEP	PC+1	uuu0 0uuu
Выход по прерыванию из режима SLEEP	PC+1(1)	uuul 0uuu

Примечания : 1. x - не определено, u - не меняется.
2. Если при выходе по прерыванию из режима SLEEP установлен бит GIE, в счетчик команд загружается адрес вектора прерывания (0004h).

Таблица 1.11

Регистр	Адрес	Сброс по включению питания	Сброс по MCLR из рабочего режима и режима SLEEP, сброс по WDT из рабочего режима	Выход из режима SLEEP по прерыванию и по WDT
1	2	3	4	5
W	—	xxxxxxxx	uuuuuuuu	uuuuuuuu
INDF	00h		--------	
TMR0	01h	xxxxxxxx	uuuuuuuu	uuuuuuuu
PCL	02h	0000h	0000h	PC+l(2)

Окончание табл. 1.11

1	2	3	4	5
STATUS	03h	00011xxx	000??uuu(3)	uuu??uuu(3)
FSR	04h	xxxxxxxx	uuuuuuu	uuuuuuuu
PORTA	05h	---xxxxx	---uuuuu	---uuuuu
PORTB	06h	xxxxxxxx	uuuuuuuu	uuuuuuuu
EEDATA	08h	xxxxxxxx	uuuuuuuu	uuuuuuuu
EEADR	09h	xxxxxxxx	uuuuuuuu	uuuuuuuu
PCLATH	0Ah	---00000	---00000	---uuuuu
INTCON	0Bh	0000000x	0000000u	uuuuuuuu(1)
INDF	80h	—	—	—
OPTION	81h	11111111	11111111	uuuuuuuu
PCL	82h	0000h	0000h	PC+l(2)
STATUS	83h	00011xxx	000??uuu(3)	uuu??uuu(3)
FSR	84h	xxxxxxxx	uuuuuuuu	uuuuuuuu
TRISA	85h	---11111	---11111	---uuuuu
TRISB	86h	11111111	11111111	uuuuuuuu
EECON1	88h	---0x000	---0?000	---0uuuu
EECON2	89h	--------	--------	--------
PCLATH	8Ah	---00000	---00000	---uuuuu
INTCON	8Bh	0000000x	0000000u	uuuuuuuu(1)

Примечания: 1. x - не определено, u - не меняется, — - отсутствует, читается как '0', ? - значение зависит от условий.
2. Один или более бит регистра INTCON будут изменены (причина выхода из режима SLEEP).
3. Если при выходе по прерыванию из режима SLEEP установлен бит GIE, в счетчик команд загружается адрес вектора прерывания (0004h).

Внешний сброс. На рис. 1.17 приведена типовая схема внешнего сброса. Данная схема необходима, если скорость нарастания напряжения питания недостаточна (менее 0,05 В/мс) либо используется низкочастотный кварцевый резонатор. Диод D предназначен для разряда конденсатора C при выключении питания. Величина резистора R не должна превышать 40 кОм, чтобы падение напряжения на нем не превысило 0,2 В при максимальном токе утечки входа MCLR 5 мкА. При большем значении падения напряжения уровень напряжения на выводе MCLR/V_{PP} будет недостаточен для надежного сброса. Резистор R1 может иметь значение в диапазоне от 1 до 100 кОм и предназначен для ограничения тока по выводу MCLR при перезаряде конденсатора C и импульсных помехах.

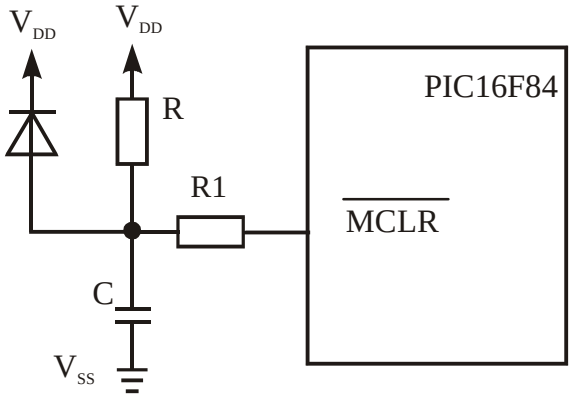


Рис. 1.17. Схема внешнего сброса

1.13.6. Прерывания

Микроконтроллер PIC16F84X имеет четыре источника прерывания:

• внешнее прерывание от вывода RB0/INT;

• прерывание при переполнении счетчика/таймера TMR0;

• прерывание по окончании записи данных в EEPROM;

• прерывание при изменении сигналов на выводах порта RB<7:4>.

Все прерывания имеют один и тот же адрес/вектор 0004h. При этом в регистре управления прерываниями INTCON (0Bh) записывается, от какого именно источника поступил запрос прерывания. Прерывания могут быть запрещены индивидуально или общим битом.

Бит общего разрешения/запрещения прерывания GIE (INTCON <7>) разрешает (если установлен в '1') все индивидуально не замаскированные прерывания или запрещает их (если сброшен в '0'). Каждое прерывание в отдельности может быть дополнительно разрешено/запрещено установкой/сбросом соответствующего бита в регистре INTCON. Бит GIE обнуляется при сбросе. Когда начинает обрабатываться прерывание, бит GIE обнуляется, чтобы запретить дальнейшие прерывания, адрес возврата сохраняется в стеке, а в счетчик команд загружается адрес 0004h. Логика прерываний изображена на рис. 1.18.

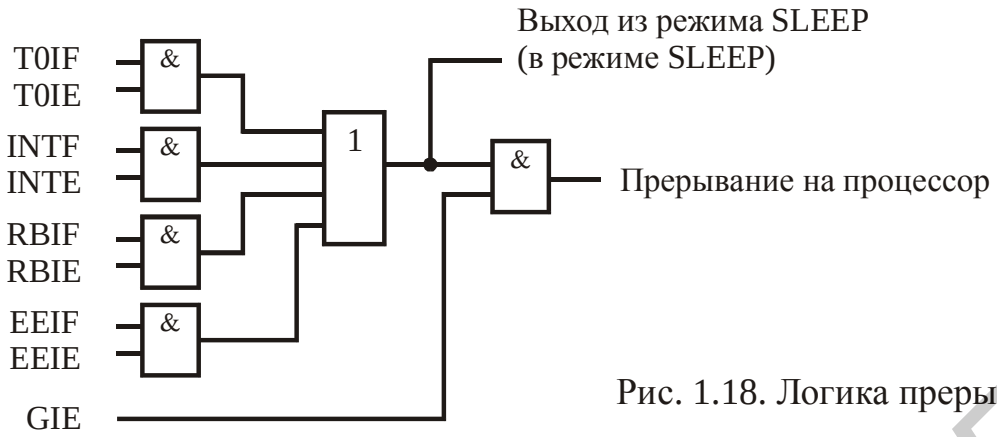


Рис. 1.18. Логика прерываний

Время реакции на прерывание для внешних событий, таких как прерывание от вывода INT или порта B, составляет от трех до четырех периодов тактовой частоты.

В подпрограмме обработки прерывания источник прерывания может быть определен по соответствующему биту в регистре INTCON. Этот бит должен быть программно сброшен внутри подпрограммы обработки прерывания для исключения рекурсивных прерываний, флаги запросов прерываний не зависят от соответствующих маскирующих бит и бита общего разрешения GIE. Команда возврата из прерывания RETFIE завершает прерывающую подпрограмму и устанавливает бит GIE, чтобы опять разрешить прерывания.

Внешнее прерывание по выводу RB0/INT осуществляется по перепаду сигнала: либо по перепаду 0/1, если бит INTEDG в регистре OPTION<6> установлен, либо по перепаду 1/0, если бит INTEDG сброшен. Когда на выводе INT фиксируется перепад сигнала, устанавливается бит запроса INTF (INTCON<1>). Это прерывание может быть запрещено сбросом управляющего бита INTE (INTCON<4>). Бит запроса INTF должен быть очищен обработчиком прерывания перед тем, как опять разрешить это прерывание. Прерывание INT может вывести процессор из режима SLEEP, если перед входом в этот режим бит INTE был установлен. Состояние бита GIE при этом

определяет, будет ли процессор переходить на подпрограмму прерывания после выхода из режима SLEEP.

Прерывание от таймера/счетчика TMR0. Переполнение счетчика TMR0 (FFh/00h) установит бит запроса T0IF (INTCON<2>). Это прерывание может быть разрешено/запрещено установкой/сбросом бита маски T0IE (INTCON<5>).

Прерывание от порта RB. Любое изменение сигналов на четырех входах порта RB<7…4> установит бит RBIF (INTCON<0>). Это прерывание может быть разрешено/запрещено установкой/сбросом бита маски RBIE (INTCON<3>).

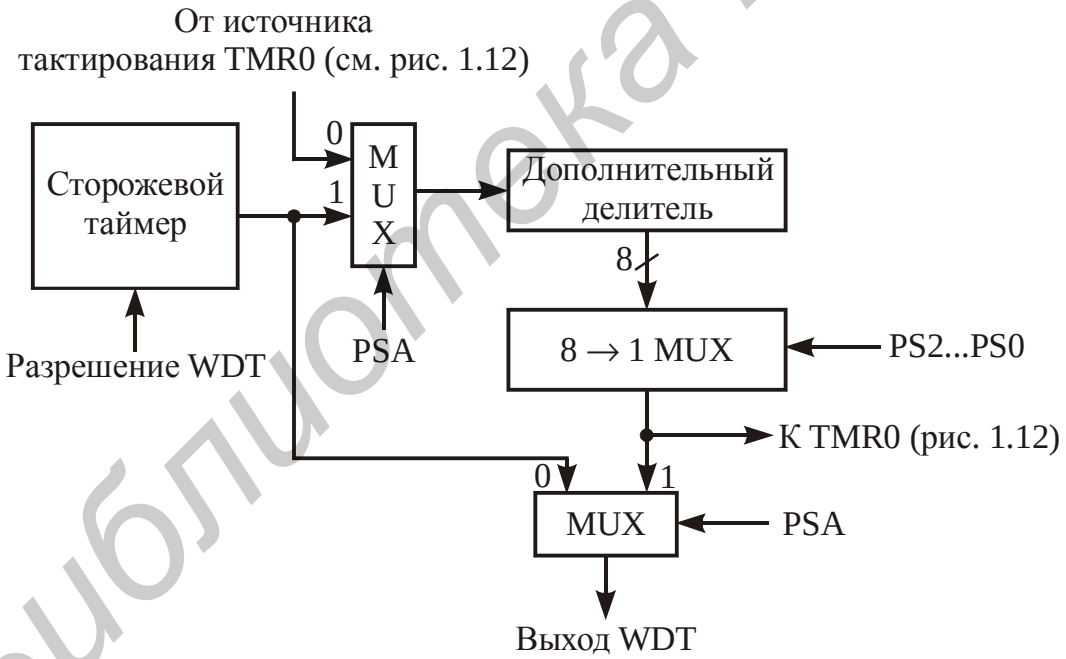
1.13.7. Сохранение состояния при прерываниях

При прерывании в стеке автоматически сохраняется только значение счетчика команд PC. Обычно пользователю также необходимо сохранить некоторые регистры (например, регистры W и STATUS). В примере ниже приведена возможная процедура сохранения и восстановления регистров W и STATUS. Для временного хранения используются регистры W_TEMP и STATUS_TEMP.

```
PUSH MOVWF    W_TEMP          ;Сохранить W в регистре W_TEMP.
     SWAPF    STATUS,W        ;
     MOVWF    STATUS_TEMP     ;Сохранить STATUS в регистре STATUS_TEMP.
ISR                           ;Процедура обработки прерывания.
     .
     .
     .
POP  SWAPF    STATUS_TEMP,W   ;
     MOVWF    STATUS          ;Восстановить STATUS.
     SWAPF    W_TEMP,F        ;
     SWAPF    W_TEMP,W        ;Восстановить W , не меняя  STATUS.
```

1.13.8. Сторожевой таймер

Сторожевой таймер WDT (рис. 1.19) представляет собой независимый встроенный RC-генератор, не требующий никаких внешних цепей. Он работает, даже если основной тактовый генератор остановлен, как это происходит при исполнении команды SLEEP. При нормальной работе сторожевой таймер по истечении заданной выдержки времени вырабатывает сигнал сброса. Если контроллер находился в режиме пониженного энергопотребления SLEEP, срабатывание сторожевого таймера выводит его из этого режима и переводит в режим нормальной работы. Функционирование сторожевого таймера может быть запрещено путем записи '0' в бит конфигурации WDTE на этапе программирования микросхем.



Примечание. PSA и PS2...PS0 – биты регистра OPTION.

Рис. 1.19. Структурная схема сторожевого таймера

Период сторожевого таймера. Номинальная выдержка сторожевого таймера составляет 18 мс (без использования предварительного делителя). Она

зависит от температуры и напряжения питания, а также меняется от разброса характеристик кристаллов. Если требуется увеличить время задержки, то к WDT может быть подключен встроенный предварительный делитель с коэффициентом деления до 1:128, программируемый путем записи в регистр OPTION. Таким образом могут быть реализованы выдержки до 2,3 с.

Команды 'CLRWDT' и 'SLEEP' обнуляют WDT и предварительный делитель, если он подключен к WDT. Таким образом, выдержка времени начинает отсчитываться сначала и на некоторое время выработка сигнала СБРОС предотвращается. При срабатывании сторожевого таймера обнуляется бит TO в регистре STATUS.

Выдержка времени сторожевого таймера максимальна при следующей комбинации параметров: минимальное напряжение V_{DD}, максимальная температура и максимальный коэффициент деления делителя. При этих условиях выдержка времени сторожевого таймера может составлять несколько секунд.

1.13.9. Режим пониженного энергопотребления (SLEEP)

Вход в режим пониженного энергопотребления SLEEP осуществляется командой SLEEP.

По этой команде сторожевой таймер, если он был разрешен при программировании контроллера, сбрасывается и начинает отсчет времени. В регистре состояния STATUS сбрасывается бит PD и устанавливается бит TO, а тактовый генератор выключается. Порты ввода/вывода сохраняют состояние, которое они имели до входа в режим SLEEP.

Для снижения тока, потребляемого в этом режиме, все разряды внешних портов кристалла должны находиться либо на уровне V_{SS}, либо на уровне V_{DD}. Все разряды портов, находящиеся в высокоимпедансном состоянии, должны быть соединены через внешние резисторы с V_{SS} или V_{DD}, чтобы избежать токов переключения, вызываемых плавающим потенциалом на высокоомных входах.

Вход T0CK1 также должен иметь потенциал V_{DD} или V_{SS}. Вывод MCLR должен иметь высокий логический уровень.

Следует отметить, что сигнал внутреннего сброса, вырабатываемый сторожевым таймером, не переключает вывод MCLR в низкий уровень.

Выход из режима SLEEP осуществляется по одному из следующих событий:

• внешний сброс — импульс низкого уровня на выводе MCLR;

• сброс при срабатывании сторожевого таймера WDT (если он разрешен);

• прерывание от вывода RB0/INT, по изменению порта RB<7…4> или по окончании записи ЭППЗУ.

Биты TO и PD регистра состояния могут быть использованы для определения причины сброса. Бит PD регистра состояния устанавливается при включении питания и обнуляется командой SLEEP. Бит TO позволяет определить, чем был вызван выход из режима SLEEP: внешним сигналом на выводе MCLR или срабатыванием сторожевого таймера. Он обнуляется только при сбросе от сторожевого таймера.

При выполнении команды SLEEP следующая команда по адресу PC+1 выбирается из памяти. Чтобы микроконтроллер вышел из режима пониженного энергопотребления по прерыванию, необходимо установить (разрешить) соответствующие биты разрешения прерывания. Выход из режима пониженного энергопотребления произойдет независимо от состояния бита GIE. Если бит GIE установлен в '0' (запрещен), микроконтроллер продолжает выполнение с команды, следующей после команды SLEEP. Если же бит GIE установлен в '1' (разрешен), микроконтроллер выполняет команду, следующую за командой SLEEP, и затем переходит на адрес прерывания 0004h. Если выполнение команды, следующей за командой SLEEP, недопустимо, необходимо после команды SLEEP добавить команду NOP.

Если прерывания запрещены (бит GIE установлен в '0'), но хотя бы для одного из источников прерывания установлен бит разрешения прерываний и соответствующий флаг разрешения прерываний, микроконтроллер немедленно выйдет из режима SLEEP. Команда SLEEP будет выполнена.

Независимо от причины сброса при выходе из режима SLEEP сторожевой таймер WDT обнуляется.

1.13.10. Защита программы от копирования

Программа, записанная в ЭППЗУ или в ПЗУ, может быть защищена от считывания при помощи установки в '0' бита защиты CP в слове конфигурации. В режиме защиты программы содержимое памяти программы не может быть прочитано в исходном виде, тем самым невозможно реконструировать записанную программу. Кроме того, при установленном бите защиты невозможно допрограммировать контроллер.

1.13.11. Индивидуальная метка

Микроконтроллер имеет четыре специальных адреса (2000h…2003h), не являющиеся частью памяти программы. Они предназначены для хранения идентификационного кода (ID) пользователя, контрольной суммы или другой информации. Как и слово конфигурации, они могут быть прочитаны или записаны только с помощью программатора. Программно эти адреса недоступны.

Для обеспечения возможности чтения одинаковой информации, как в обычном режиме, так и в режиме с установленным битом защиты от считывания рекомендуется использовать только четыре младших бита по каждому адресу для хранения кода ID. Старшие четыре бита рекомендуется устанавливать в '1'.

1.13.12. Внутрисхемное программирование

Микроконтроллер может быть запрограммирован непосредственно в готовом устройстве. Для этого требуется два вывода для подключения тактового сигнала CLK и данных DATA и три вывода для питающего напряжения V_{DD}, программирующего напряжения V_{PP} и общего вывода GND.

Эта возможность позволяет изготавливать устройства с незапрограммированными микроконтроллерами, занося в них код программы непосредственно перед продажей. Таким способом можно обеспечить использование самой последней версии программного обеспечения, а также учесть особенности конкретного потребителя.

Переход в режим программирования происходит по переходу сигнала MCLR от логической единицы до напряжения программирования (+12 В), когда на выводах RB6 и RB7 установлен низкий уровень. Вывод RB6 используется в качестве тактового сигнала, а вывод RB7 — в качестве двунаправленного разряда данных. В режиме программирования разряды RB6 и RB7 имеют на входе триггеры Шмитта.

1.14. Предельные эксплуатационные характеристики

Диапазон рабочих температур …….	-55...+125°C
Диапазон температуры хранения…….	-65...+150°C
Напряжение на выводе V_{DD} относительно V_{SS}……	0...+7,5 В
Напряжение на выводе MCLR относительно V_{SS} …..	0...+14 В
Напряжение на любом выводе относительно V_{SS}, кроме V_{DD} и MCLR ……...	-0,6...V_{DD}+0,6 В
Общая рассеиваемая мощность……	800 мВт
Максимальный ток по выводу V_{SS}……..	150 мА

Максимальный ток по выводу V_{DD}…….. 100 мА

Входной ток ограничения I_{1K} (V_1<0 или V_1> V_{DD})….. ±20 мА

Выходной ток ограничения I_{0K} (V_0<0 или V_0> V_{DD})... ±20 мА

Максимальный выходной втекающий ток по любому разряду порта……... 25 мА

Максимальный выходной вытекающий ток по любому разряду порта…….. 20 мА

Максимальный суммарный втекающий ток всех разрядов одного порта А…….. 80 мА

Максимальный суммарный втекающий ток всех разрядов одного порта В…….. 150 мА

Максимальный суммарный вытекающий ток всех разрядов одного порта А…….. 50 мА

Максимальный суммарный вытекающий ток всех разрядов одного порта В……... 100 мА

Полная рассеиваемая корпусом мощность не должна превышать 800 мВт.

Пики напряжения на выводе MCLR ниже уровня общего V_{SS} или индуцированные токи более 80 мА могут привести к "защелкиванию" кристалла. Поэтому рекомендуется подавать сигналы на вывод MCLR через ограничивающий резистор 50...100 Ом.

2. ОСНОВЫ ПРОГРАММИРОВАНИЯ PIC16F84 НА АССЕМБЛЕРЕ

2.1. Система команд

Каждая команда микроконтроллера представляет собой 14-разрядное слово, содержащее поле кода операции и поле операндов. Система команд PIC16F84 приведена в табл. 2.1. Она команд включает в себя команды работы с байтами, команды работы с битами, команды передачи управления и операции с константами. Форматы команд показаны на рис. 2.1.

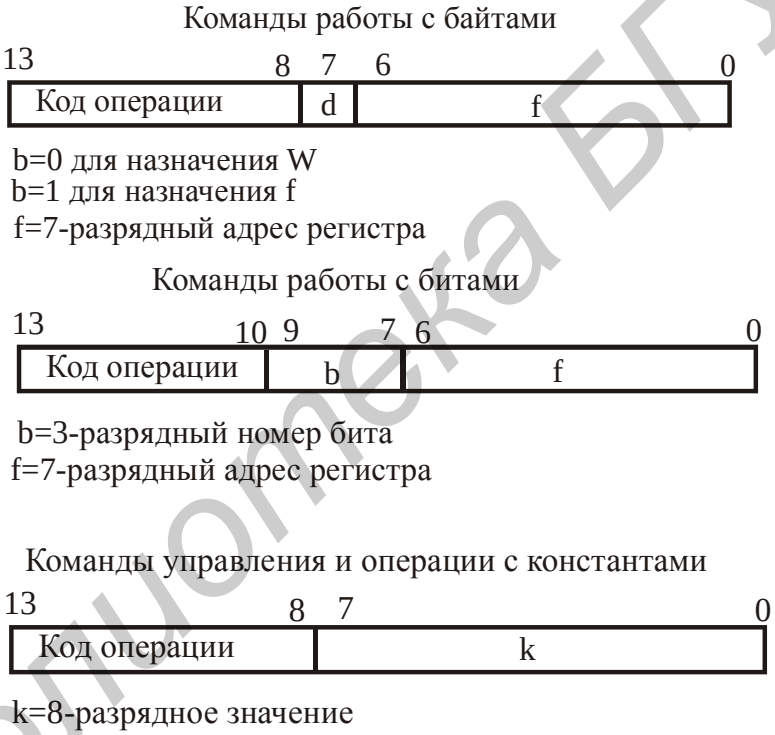


Рис. 2.1. Форматы команд

Для **команд работы с байтами** 'f' обозначает регистр, с которым производится действие, а бит 'd' определяет регистр назначения. При 'd'=0 результат помещается в регистр W, при 'd'=1 результат помещается в регистр 'f', заданный в команде.

Для **команд работы с битами** 'b' обозначает номер бита, участвующего в команде, а 'f'— регистр, в котором этот бит расположен.

Для **команд управления и операций с константами** 'k' обозначает 8- или 11-битовую константу или идентификатор.

Все команды выполняются в течение одного командного цикла, кроме следующих двух случаев:

• переход по проверке условия, если результат проверки условия — истина;

• изменение счетчика команд как результат выполнения команды.

В этих случаях команда выполняется за два цикла с выполнением второго цикла как NOP. Один командный цикл состоит из четырех периодов генератора. Таким образом, для генератора с частотой 4 МГц время выполнения команды составит 1 мкс. Если выполняется переход по проверке условия или в результате выполнения команды изменился счетчик команд, время выполнения этой команды при тактовой частоте 4 МГц составит 2 мкс.

Таблица 2.1

Мнемоника	**Операнды**	**Описание**	**Флаги**
1	2	3	4
Команды работы с байтами			
ADDWF	f, d	Сложить регистр и W	C, DC, Z
ANDWF	f, d	Выполнить логическое И регистра и W	Z
CLRF	f	Обнулить регистр	Z
CRLW		Обнулить W	Z
COMF	f, d	Инвертировать регистр	Z
DECF	f, d	Декрементировать регистр	Z
DECFSZ	f, d	Декрементировать регистр и пропустить следующую команду, если результат равен 0	
INCF	f, d	Инкрементировать регистр	Z
INCFSZ	f, d	Инкрементировать регистр и пропустить следующую команду, если результат равен 0	
IORWF	f, d	Выполнить логическое ИЛИ с регистром и W	Z
MOVF	f, d	Переместить регистр в W	Z
MOVWF	f	Переместить W в регистр	
NOP		Нет операции	
RLF	f, d	Сдвинуть регистр влево через флаг переноса	C
RRF	f, d	Сдвинуть регистр вправо через флаг переноса	C

Окончание табл. 2.1

1	2	3	4
SUBWF	f, d	Вычесть W из регистра	C, DC, Z
SWAPF	f, d	Поменять тетрады регистра местами	
XORWF	f, d	Выполнить логическое ИСКЛЮЧАЮЩЕЕ ИЛИ с регистром и W	Z
Команды работы с битами			
BCF	f, b	Сбросить бит регистра	
BSF	f, b	Установить бит регистра	
BTFSC	f, b	Пропустить, если сброшен бит в регистре	
BTFSS	f, b	Пропустить, если установлен бит в регистре	
Команды работы с константами			
ADDLW	k	Сложить W и константу	C, DC, Z
ANDLW	k	Выполнить логическое И с константой и W	Z
IORLW	k	Выполнить логическое ИЛИ с константой и W	Z
MOVLW	k	Записать константу в W	
SUBLW	k	Вычесть константу из W	C, DC, Z
XORLW	k	Выполнить логическое ИСКЛЮЧАЮЩЕЕ ИЛИ с константой и W	Z
Команды передачи управления			
CALL	k	Перейти на подпрограмму	
GOTO	k	Перейти на метку	
RETFIE		Вернуться из прерывания	
RETURN		Вернуться из подпрограммы	
RETLW	k	Вернуться из подпрограммы и записать константу в W	
CLRWDT		Сбросить сторожевой таймер	TO, PD
SLEEP		Войти в режим пониженного энергопотребления	TO, PD

2.2. Составление схем алгоритмов

Современные вычислительные машины способны выполнять широкий круг задач по получению, передаче, хранению, переработке информации, принятию решений и т. д. Но все эти действия должны быть заранее подготовлены, запрограммированы человеком.

Создание программы основывается на алгоритме решения задачи, в соответствии с ним создается последовательность команд, которая и составляет программу.

Под алгоритмом понимается конечный набор действий для выполнения некоторой процедуры, удовлетворяющий трем основным требованиям: массовости, детерминированности и результативности.

Требование массовости предполагает, что предписание должно обеспечивать выполнение не одной конкретной процедуры, а быть пригодным для реализации класса однородных процедур. Бессмысленно писать программу для получения суммы двух констант, но имеет смысл программа, определяющая сумму двух переменных, которые могут принимать множество значений в некотором диапазоне.

Детерминированность означает, что действия, образующие алгоритм, должны быть однозначно понимаемы, т.е. обеспечивается одинаковость результатов при одинаковых исходных данных независимо от исполнителя.

Результативность обеспечивает конечность применения указаний. Результат должен быть получен за конечное число шагов либо за конечное число шагов должно быть получено указание о неприменимости данного алгоритма для решения поставленной задачи.

Для успешного решения задачи на вычислительной машине (микроконтроллере) разработчик должен пройти следующих семь этапов: 1) постановка задачи; 2) выбор приемлемого алгоритма; 3) определение типов входных и выходных данных; 4) распределение аппаратных ресурсов микроконтроллера, т.е. портов ввода/вывода и периферийных устройств на кристалле; 5) проектирование и анализ решения, в том числе составление схем, описаний и пр.; 6) кодирование алгоритма на языке программирования; 7) проверка и отладка программы.

Процесс решения задачи носит, как правило, итерационный характер. Это означает, что, получив решение, разработчик часто бывает вынужден вернуться вновь к третьему, второму и даже первому этапу.

По-видимому, первые рассмотренные выше четыре этапа относятся к тому, что принято называть «искусством разработки». Здесь успех в основном

определяется опытом разработчика, его знанием объекта разработки, и дать конкретные рекомендации не представляется возможным. Однако можно детальнее остановиться на составлении схем алгоритмов.

Изображение алгоритма решения задачи в виде схемы - важный этап подготовки задачи к решению на вычислительной машине. Схема позволяет разработчику адекватно представить работу программы. Кроме этого, схема алгоритма является одной из важных частей документации на разрабатываемую систему или устройство.

Схема алгоритма составляется из отдельных операторов. Различают шесть типов операторов, каждый из которых имеет один или несколько входов или один или несколько выходов (рис. 2.2). Стрелками обозначают направление хода действий.

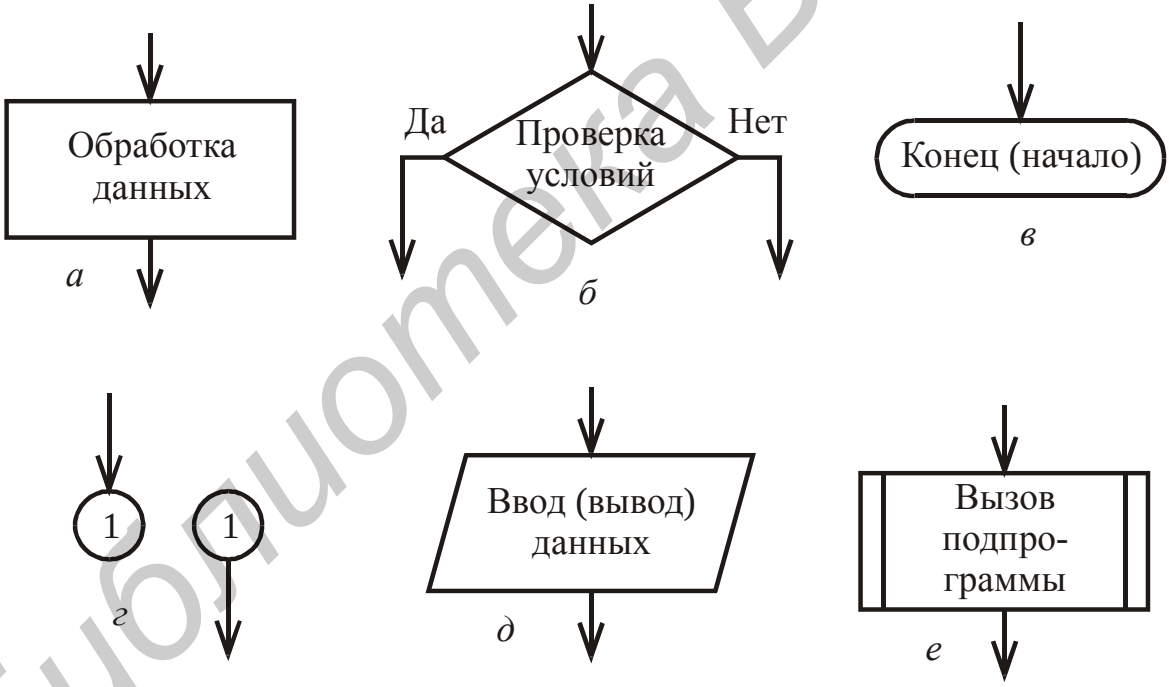


Рис. 2.2. Операторы обработки данных (*а*), проверки условия (*б*) начала или конца алгоритма (*в*), соединения (*г*), ввода или вывода данных (*д*), вызова подпрограммы (*е*)

Оператор в форме прямоугольника (рис. 2.2, *а*) символизирует выполнение каких-либо операций по обработке данных; текст внутри прямоугольника является кратким описанием этого процесса обработки.

Например, если в схеме алгоритма содержится оператор «Очистка аккумулятора», то это означает, что на данном этапе работы машины аккумулятор должен быть обнулен. В таком случае для выполнения этой операции микроконтроллеру требуется одна машинная команда. Однако могут быть заданы сложные действия, для которых требуется целый набор команд.

Оператор, имеющий форму ромба (рис. 2.2, *б*), используется для символического обозначения проверки выполнения какого-либо условия с целью принятия решения о направлении последующего хода вычислений. Внутри ромба описывается условие, подлежащее проверке в той точке схемы алгоритма, где размещается данный оператор. Возможные результаты проверки указываются на линиях, выходящих из ромба. Для проведения подобной проверки требуется использование одной или нескольких команд микроконтроллера.

Оператор овальной формы используется для символического обозначения начала или конца алгоритма (рис. 2.2, *в*). Текст внутри овала, как правило, состоит из одного слова — «Начало» или «Конец».

В тех случаях, когда необходимо «разорвать» линию потока вычислений, идущую от одного оператора к другому, применяются так называемые соединители в виде окружности с указанной внутри нее цифрой или буквой (рис. 2.2, *г*). Наличие другого идентичного соединителя (с той же цифрой или буквой) означает, что прерванная в месте расположения первого соединителя линия продолжается с того места, где находится второй подобный соединитель. Использование соединителей упрощает внешний вид схемы алгоритма, что позволяет избежать пересечения линий и даёт возможность размещать схему алгоритма на нескольких страницах.

Для обозначения процедур ввода или вывода применяется оператор, имеющий форму параллелограмма (рис. 2.2, *д*). Внутри параллелограмма указывают обычно переменные, подлежащие вводу или выводу.

Оператор, изображенный на рис. 2.2 *е*, используется для обозначения вызова подпрограммы. Внутри него обычно помещается имя вызываемой подпрограммы.

Алгоритмы в зависимости от порядка следования операций бывают трех типов: линейные, ветвящиеся и циклические.

Схемы линейных алгоритмов не содержат операторов проверки условий, в них операторы располагаются строго друг за другом.

Ветвящиеся алгоритмы состоят из двух и более параллельных линейных ветвей, при этом ветвления реализуются через операторы проверки условий.

Циклический алгоритм предписывает многократные циклические проходы группы операторов.

Реальные схемы алгоритмов, как правило, представляют собой комбинации из трех перечисленных выше типов схем алгоритмов.

2.3. Общие правила Ассемблера

Обычно программа начинается со строки комментариев, в которых указывается, чья это программа и зачем написана. Строки комментариев могут быть в любом месте программы.

Далее, после комментариев (если они есть) и перед первыми командами программы размещаются строки директив. Директивы – это указания ассемблеру, как именно работать. Список директив ассемблера довольно обширный, но мы на данном этапе будем использовать только самые необходимые. Комментарии и директивы не включаются в исполняемый код программы и, следовательно, не попадают в память программ. Строки директив также могут быть в любом месте программы, но некоторые директивы обязательно должны быть заданы до первой строки команд.

Рассмотрим, что в программе должно быть обязательно.

Директива LIST с опцией P, она указывает, с каким микроконтроллером идет работа.

Директива EQU присваивает символическому имени определенное выражение или значение. Присвоенное значение впоследствии в программе переопределить нельзя. Эта директива позволяет программисту оперировать в программе не физическими адресами регистров (ячеек памяти), а их условными именами, которые придумывает сам программист. Если при этом в условное имя закладывается физический смысл переменной, размещающейся в этом регистре, то его легче помнить.

Директива END означает, что текст программы закончился. Эта директива должна быть в последней строке программы.

Каждая строка программы может иметь до четырех разделов (полей) и может содержать до 255 символов.

С первой позиции в строке начинается поле метки. Метка – это условное символическое имя, присваиваемое конкретной строке, а также переменной или константе. Метка должна начинаться с буквы или символа подчеркивания (_) и содержать до 32 символов букв или цифр. В поле метки прописные и строчные буквы различаются. Поле метки должно заканчиваться символом пробела, табуляции или конца строки.

Далее следует второе поле – мнемонический код команды. Оно начинается со второй позиции в строке, а если перед ним стоит метка, то от метки мнемоника должна быть отделена двоеточием, одним или более пробелами или символом табуляции.

В третьем поле задаются операнды. От мнемоники они отделяются одним или более пробелами или символом табуляции. Операндов должно быть столько, сколько требует формат команды. Между собой операнды разделяются запятой. Если команда предусматривает переменное количество операндов, то они считаются до конца строки или до четвертого поля (начало комментария).

Четвертое поле – комментарии. Начинаются с символа точки с запятой. От остальных полей отделяются одним или более пробелами или символом табуляции. Могут занимать всю строку, с первой позиции.

Программа обязательно заканчивается директивой END.

2.4. Пример простейшей программы управления портами

Пусть требуется спроектировать устройство на микроконтроллере (МК) PIC16F84, которое обеспечивает мигание светодиода на время нажатия и удержания кнопки. Другими словами: пока нажата кнопка, светодиод мигает, при отпущенной кнопке светодиод погашен.

Решение. В этой задаче один источник информации и один объект управления. Источник информации представляет собой два разомкнутых контакта. При нажатии кнопки контакты замыкаются. Микроконтроллер должен определять состояние кнопки. Таких состояний два: "разомкнуто" и "замкнуто". Эти состояния можно условно описать одной булевой переменной *x*, которая может принимать два значения: 0 и 1. Таким образом, кнопка как источник даёт один бит информации. Для приёма этой информации достаточно задействовать лишь какой-либо один из входов МК. Сама по себе кнопка является электрически пассивным источником, и для определения ее состояния с помощью порта МК необходимо преобразовать состояние контактов в электрический сигнал напряжения. Наиболее просто это можно выполнить по схеме, изображенной на рис. 2.3.

В состоянии «разомкнуто» по порту МК будет читаться логическая единица, а в состоянии «замкнуто» — логический нуль. Учитывая, что в PIC16F84 все разряды порта B имеют подтягивающие внутренние резисторы, внешний резистор R можно не использовать, а кнопку подключить к выводу порта и к общему проводу. Подключим в проектируемом устройстве кнопку SB

к порту RB0 и будем помнить, что в программе необходимо позаботиться о включении подтягивающих резисторов.

Объектом управления в устройстве является светодиод. Его состояние «включен» обеспечивается пропусканием тока $I_Д$ силой в несколько миллиампер (обычно 5-10 мА). Поскольку нагрузочная способность портов МК 20 мА, то светодиод можно подключать без дополнительного усилителя непосредственно к порту, предусмотрев лишь токоограничивающий резистор R (рис. 2.4). Сопротивление резистора можно рассчитать по формуле $R=(E^1-U_{VD})/I_Д$, где E^1=5 В — уровень логической единицы, U_{VD}=2,4 В — падение напряжения на открытом светодиоде. Зададимся $I_Д$=5мА, тогда

$R=(5-2,4)/(5\cdot10^{-3})=2,6/(5\cdot10^{-3})$ =2600/5=520 Ом.

Выберем для управления светодиодом порт RA0.

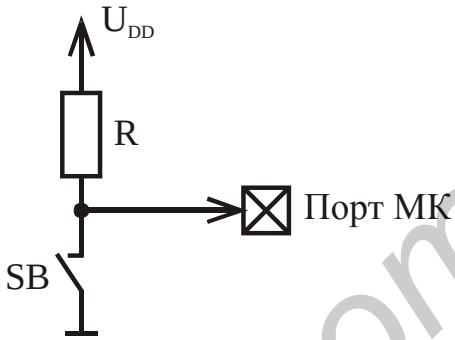


Рис. 2.3. Схема подключения кнопки к порту МК

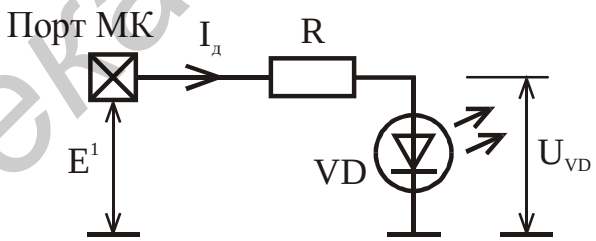


Рис. 2.4. Схема подключения светодиода к МК

Для обеспечения требований к временным параметрам устройства необходимо выбрать частоту и вариант тактирования работы МК. Выберем вариант синхронизации с помощью кварцевого резонатора, имеющего резонансную частоту 4 МГц. Таким образом, длительность машинного цикла (время выполнения большинства команд) составит 1 мкс.

Составим схему алгоритма управляющей программы (рис. 2.5).

От схемы алгоритма легко перейти к программе.

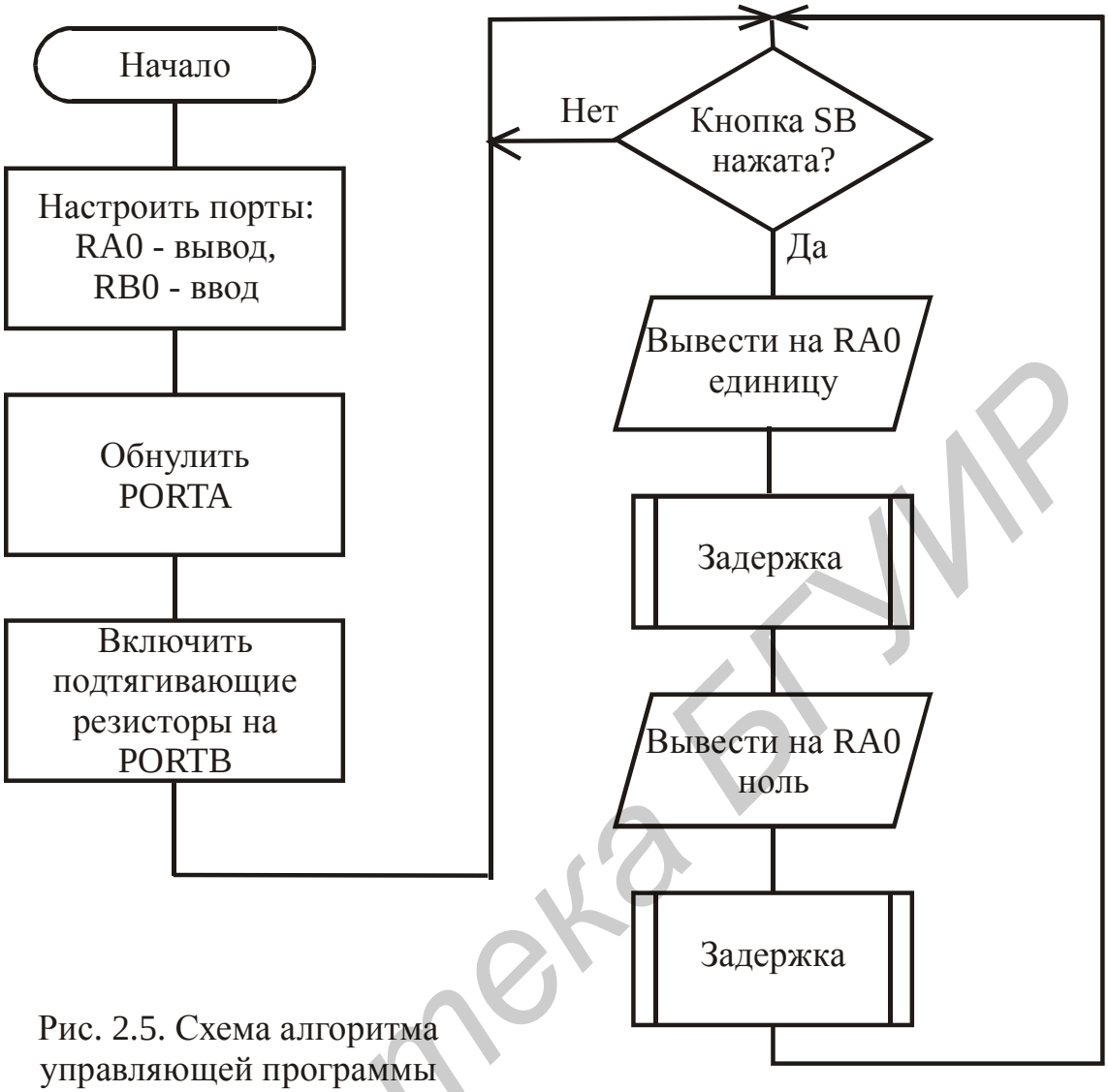


Рис. 2.5. Схема алгоритма управляющей программы

;Учебная программа №1.

LIST p=16f84 ; Директива Ассемблеру с указанием типа МК PIC16F84.

; Определяем имена констант, используемых в этой программе.
; После этого определения можно вместо шестнадцатеричного
; адреса регистра или номера разряда в регистре указывать его имя,
; которое сами придумаем. Это облегчает чтение и понимание
; работы программы.

```
STATUS   EQU   03h   ; Символ  h  в конце записи числа является
RP0      EQU   05h   ; указателем шестнадцатеричной
TRISA    EQU   05h   ; системы счисления.
TRISB    EQU   06h   ;
PORTA    EQU   05h   ;
PORTB    EQU   06h   ;
RA0      EQU   0h    ;
RB0      EQU   0h    ;
OPTION   EQU   01h   ;
RBPU     EQU   07h   ;
```

```
        ; Настраиваем порты
        BSF   STATUS,RP0      ; Включаем банк памяти 1 для доступа к регистрам TRIS.
        BCF   TRISA,RA0       ; Линия RA0 – на вывод  данных.
        BSF   TRISB,RB0       ; Линия RB0 – на ввод данных.
        BCF   OPTION,RBPU     ; Включаем подтягивающие резисторы на входах порта B
                              ; запрограммированных как входы.
        BCF   STATUS,RP0      ; Включаем банк памяти 1 для доступа к портам.
        CLRF PORTA            ; Обнуляем порт A (гасим светодиод).
                    ; Ожидаем нажатия кнопки SB.
Wait    BTFSC PORTB,RB0       ; В цикле проверяем линию RB0.
        GOTO    Wait          ; При нажатии кнопки на линии будет читаться ноль,
                              ; что вызовет пропуск команды GOTO и выход из цикла.
        BSF PORTA,RA0         ; Зажигаем светодиод.
        CALL DELAY            ; Вызываем подпрограмму задержки.
        BCF PORTA,RA0         ; Гасим светодиод.
        CALL DELAY            ; Снова задержка.
        GOTO    Wait          ; Переход в точку контроля нажатия кнопки.
DELAY               ; Начало подпрограммы задержки.
        …
        ; Тело (цепь команд) подпрограммы задержки.
        ; Она рассматривается в следующем разделе.
        …
        RETURN      ; Конец подпрограммы задержки.
        END         ; Конец программы.
```

2.5. Программирование задержек

Процедура задержки является одной из самых часто применяемых, особенно в алгоритмах управления, а также формирования и анализа сигналов. В рассматриваемом микроконтроллере она может быть реализована тремя способами:

• линейной цепочкой пустых команд NOP;

• многократным циклическим повторением цепочки команд, в том числе NOP;

• с применением счетчика-таймера.

Первый способ самый простой, но он пригоден только для реализации коротких задержек. Величина задержки рассчитывается по формуле $\Delta t = t_{кц}\ N$, где $t_{кц}$ – длительность командного цикла (при тактовой частоте 4МГц она равна 1 мкс), N- количество команд NOP в цепочке.

По второму способу организуется конечное число проходов цепочки команд. Для этого отводится одна из ячеек памяти данных, в которой организуется счетчик циклов. В ячейку заносится константа, которая определяет количество повторений. После каждого прохода из счетчика циклов вычитается единица. Выход из цикла обеспечивается по нулевому значению счетчика. Максимальное значение счетчика циклов ограничивается разрядностью ячейки. Оно не может быть больше 255. Задержка, обеспечиваемая этим способом, рассчитывается по формуле $\Delta t = t_{цк}\ N$, где $t_{цк}$ – время выполнения цепочки команд, N- количество повторений.

Рассмотрим пример циклической программы задержки.

```
     N      EQU  0Ch   ;Резервируем ячейку памяти по адресу 0Ch под счетчик циклов.
     MOVLW  07h    ; Заносим число 7 в рабочий регистр,
     MOVWF  N,1    ; а затем переписываем его в счетчик циклов.
Cycl NOP           ; Начало циклического участка программы.
     .             ; Команды NOP для удлинения времени выполнения участка,
     .             ; в реальной программе могут отсутствовать.
     NOP           ;
     DECFSZ N,1    ; Вычитание из счетчика циклов единицы и проверка условия
     GOTO Cycl     ; выхода из цикла.
```

Для организации больших задержек могут быть использованы вложенные циклы. Рассмотрим фрагмент программы, реализующий задержку с двухкратным вложенным циклом.

```
     N0     EQU  0Ch   ; Ячейка памяти по адресу 0Ch под счетчик внутренних циклов.
     N1     EQU  0Dh   ; Ячейка памяти по адресу 0Dh под счетчик внешних циклов.
```

```
        MOVLW   0Eh   ; Задаем количество внешних циклов
        MOVWF   N1,1  ;       N1=14.
Cycl_1                ; Начало внешнего цикла.
        MOVLW   08h   ; Задаем количество внутренних циклов
        MOVWF   N0,1  ;       N0=8.
Cycl_0                ; Начало внутреннего цикла.
        DECFSZ  N0,1  ; Вычитание из счетчика внутренних циклов единицы
        GOTO    Cycl_0; и проверка условия выхода из цикла.
        DECFSZ  N1,1  ; Вычитание из счетчика внешних циклов единицы
        GOTO    Cycl_1; и проверка условия выхода из цикла.
```

Недостатком рассмотренных способов организации задержек является то, что микроконтроллер занят только задержками и не может выполнять других полезных действий.

Наиболее совершенный способ реализации задержек — с помощью счетчика-таймера TMR0. Как и в предыдущем случае, задержка отмеряется по обнулению счетчика, в который предварительно записывается начальная константа К. Но в отличие от предыдущего случая счетчик переключается не программно, а аппаратно и обнуление счетчика фиксируется через систему прерываний. Так что в промежутках времени от запуска задержки до ее окончания микроконтроллер может выполнять любые другие полезные действия. TMR0 в процессе работы инкрементируется с частотой командных циклов. Его обнуление происходит через переполнение. Таким образом, реализуемая задержка $\Delta t=(255-K)t_{кц}$.

Увеличить время задержки можно, подключив на вход TMR0 предварительный делитель и задав с помощью регистра OPTION его коэффициент деления N. Тогда реализуемая задержка $\Delta t=N(255-K)t_{кц}$.

Рассмотрим реализацию генератора прямоугольных импульсов (меандра) на базе таймера TMR0. Пусть выходной сигнал формируется на выводе RA0, а частота синхронизации микроконтроллера равна 4 МГц. Длительность командного цикла $t_{кц}=1$ мкс.

```
; Пример программной  реализации генератора прямоугольных импульсов.
      LIST   p=16f84
; Определяем константы и адреса регистров, флагов и переменных.
K           EQU   AAh   ; Начальное значение таймера TMR0=170.
STATUS      EQU   03h   ; Символ  h  в конце записи числа является
RP0         EQU   05h   ; указателем шестнадцатеричной
TRISA       EQU   05h   ; системы счисления.
PORTA       EQU   05h   ;
RA0         EQU   00h   ;
OPTION      EQU   01h   ;
T0CS        EQU   05h   ; Источник тактирования для TMR0.
TMR0        EQU   01h   ;
INTCON      EQU   0Bh   ;
T0IF        EQU   02h   ; Флаг прерывания по переполнению TMR0.
T0IE        EQU   05h   ; Маска прерывания по переполнению TMR0.
GIE         EQU   07h   ; Маска глобального прерывания.
W_TEMP      EQU   0Ch   ; Ячейка памяти для сохранения регистра W
                        ; на время обработки прерывания.
STATUS_TEMP  EQU 0Bh    ; Ячейка памяти для сохранения регистра
                        ; STATUS на время обработки прерывания.
; Начало программы.
      ORG 0       ;  Директива Ассемблеру установить программный счетчик в нуль.
      GOTO  Begin ; Перейти к началу основной программы.
      ORG  4      ; Установить программный счетчик на адрес вектора прерывания.
      GOTO  Int   ; Перейти на программу обработки прерывания.
Begin ; Начало основной программы.
; Выполняем настройки режимов работы функциональных узлов микроконтроллера.
      BSF STATUS,RP0  ; Включаем банк регистров 1.
      BCF TRISA,RA0   ; Порт RA0 включаем на выдачу сигналов.
      BCF OPTION,T0CS ; Включаем тактирование TMR0 с частотой командных циклов.
      BSF INTCON,T0IE ; Разрешаем прерывания по переполнению TMR0.
      BCF STATUS,RP0  ; Включаем банк регистров 0.
```

```
        CLRF PORTA      ; Обнуление выходного сигнала.
        MOVLW   K       ; Загрузка TMR0
        MOVWF TMR0      ; начальным значением.
        BSF INTCON,GIE  ; Разрешаем прерывания от переполнения TMR0.
Wait    ; Здесь могут располагаться команды,
        ; выполняющие некоторые полезные действия, например
        ; обслуживание индикатора, ввод информации с клавиатуры и т.д.
        GOTO Wait       ; Зацикливаем участок программы в ожидании прерывания.
                        ; Выход отсюда возможен только по прерыванию.
Int     ; Начало подпрограммы обработки прерывания. Здесь на каждое
        ; прерывание производится инверсия порта A. Поскольку на выход
        ; настроен только вывод RA0, то инвертировать будет только он.
        MOVWF W_TEMP ; Сохранить W в регистре TEMP.
        SWAPF STATUS,0
        MOVWF STATUS_TEMP ; Сохранить STATUS в регистре TEMP.
        BCF STATUS,RP0  ; Включаем банк регистров 0.
        COMF PORTA,1    ; Инверсия выходной линии.
        MOVLW  K        ; Запись начальной константы K
        MOVWF TMR0,1    ; в таймер TMR0.
        BCF INTCON,T0IF ; Сброс флага прерывания по переполнению TMR0.
; Восстанавливаем STATUS и W.
        SWAPF STATUS_TEMP,0;
        MOVWF STATUS    ;
        SWAPF W_TEMP,1  ;
        SWAPF W_TEMP,0  ;
        RETFIE          ; Вернуться из прерывания, разрешить следующее прерывание.
        END     ; Конец программы.
```

ЛИТЕРАТУРА

1. Однокристальные микроконтроллеры Microchip: PIC16c8x/Пер. с англ.; Под ред. А.Н. Владимирова. –Рига:ORMIX, 1996. –120 с.

2. Дружинин А.А. PIC и его команда. Рига:MEMEX BALTIC, 1996.-129 с.

СОДЕРЖАНИЕ

Св. план 2002, поз.68

Учебное издание

Левкович Василий Николаевич

АРХИТЕКТУРА И ОСНОВЫ ПРОГРАММИРОВАНИЯ ОДНОКРИСТАЛЬНЫХ МИКРОКОНТРОЛЛЕРОВ PIC16F84

Методическое пособие
к лабораторным работам и курсовому проектированию
по дисциплине “Вычислительные и микропроцессорные устройства»
для студентов специальностей 39 01 02 «Радиоэлектронные системы» и
39 01 01 «Радиотехника» БГУИР всех форм обучения

Редактор Т.А. Лейко

Корректор Е.Н. Батурчик

Подписано в печать Формат 60x84 1/16.
Бумага Печать Гарнитура
Усл. -печ. л. Уч.-изд. л. 3,5. Тираж 150 экз.
Заказ

Издатель и полиграфическое оформление:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП №156 от 05.02.2001.
Лицензия ЛВ №509 от 03.08.2001.
220013, Минск, П.Бровки,6