ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1 СИСТЕМА КОМАНД PIC-КОНТРОЛЛЕРА PIC16F84

1.1 Программная модель PIC-контроллера

Микроконтроллер (МК) PIC16F84 относится к семейству 8-разрядных КМОП микроконтроллеров фирмы Microchip с гарвардской архитектурой. Он имеет внутреннее ОЗУ 1К×14 бит для программ, 8-битную организацию слов данных и 64 байт памяти данных типа EEPROM. МК имеет тридцать шесть 8-разрядных регистров общего назначения и пятнадцать специальных аппаратных регистров спецфункций (SFR). Все команды состоят из одного 14-битного слова и исполняются за один цикл (400 нс при тактовой частоте 10 МГц), кроме команд перехода, которые выполняются за два цикла (800 нс). PIC16F84 имеет четырехуровневую систему прерываний и восьмиуровневый аппаратный стек. Периферия включает 8-разрядный таймер/счетчик с 8-разрядным программируемым предварительным делителем (фактически 16-разрядный таймер), сторожевой таймер WDT с собственным встроенным генератором, обеспечивающим повышенную надежность, и 13 линий двунаправленного ввода/вывода, сгруппированных в два порта RA и RB. МК можно перевести в экономичный режим SLEEP. Обозначения выводов PIC16F84 представлены на рисунке 1.

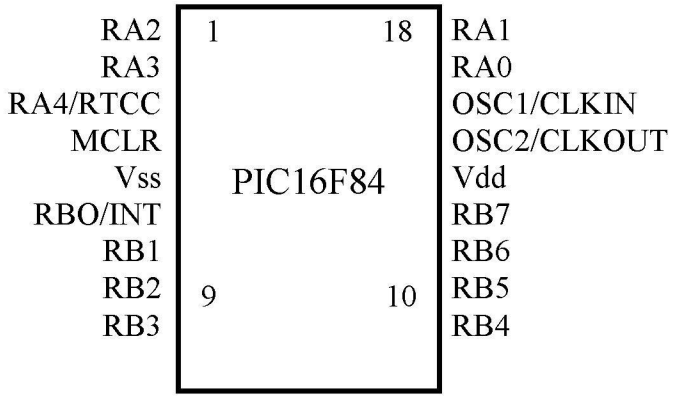

Рисунок 1 – Обозначения выводов PIC16F84

Обозначения выводов и их функциональное назначение представлены в таблице 1.

Таблица 1 – Обозначения выводов PIC16F84 и их функциональное назначение для двух режимов работы МК

Обозначение	Нормальный режим	Режим записи EEPROM
RA0 – RA3	Двунаправленные линии ввода/вывода. Входные уровни ТТЛ	–
RA4/RTCC	Вход через триггер Шмитта. Ножка порта ввода/вывода с открытым стоком или вход частоты для таймера/счетчика RTCC	–

Продолжение таблицы 1

Обозначение	Нормальный режим	Режим записи EEPROM
RB0/INT	Двунаправленная линия порта ввода/ вывода или внешний вход прерывания. Уровни ТТЛ	–
RB1 - RB5	Двунаправленные линии ввода/ вывода. Уровни ТТЛ	–
RB6	Двунаправленные линии ввода/ вывода. Уровни ТТЛ	Вход тактовой частоты для EEPROM
RB7	Двунаправленные линии ввода/ вывода. Уровни ТТЛ	Вход/выход EEPROM данных.
MCLR/Vpp	Низкий уровень на этом входе генерирует сигнал сброса для контроллера. Активный низкий	Сброс контроллера Для режима EEPROM – подать Vpp.
OSC1/CLKIN	Для подключения кварцевого генератора, RC-генератора или вход внешней тактовой частоты	–
OSC2/CLKOUT	Выход тактовой частоты в режиме RC-генератора, в остальных случаях – для подключения кварцевого генератора	–
Vdd	Напряжение питания	Напряжение питания
Vss	Общий(земля)	Общий

Структура PIC16F84 схематически представлена на рисунке 2.

Область ОЗУ организована как 128 х 8. К ячейкам ОЗУ можно адресоваться прямо или косвенно, через регистр – указатель FSR (04h). Это также относится и к EEPROM памяти данных – констант. Структура пространства регистров представлена в таблице 2

Таблица 2 – Структура и адреса регистров PIC16F84

	Page 0	**Page 1**	
00	Косв. адр.		80
01	RTCC	OPTION	81
02	PCL		82
03	STATUS		83
04	FSR		84
05	PORT A	TRISA	85
06	PORT B	TRISB	86
07			87
08	EEDATA	EECON1	88
09	EEADR	EECON2	89

Продолжение таблицы 2

	Page 0	Page 1	
0A	PCLATH		8A
0B	INTCON		8B
0C 2F	36 регистров общего назначения	То же	8C AF
30 7F	Не существует		B0 FF

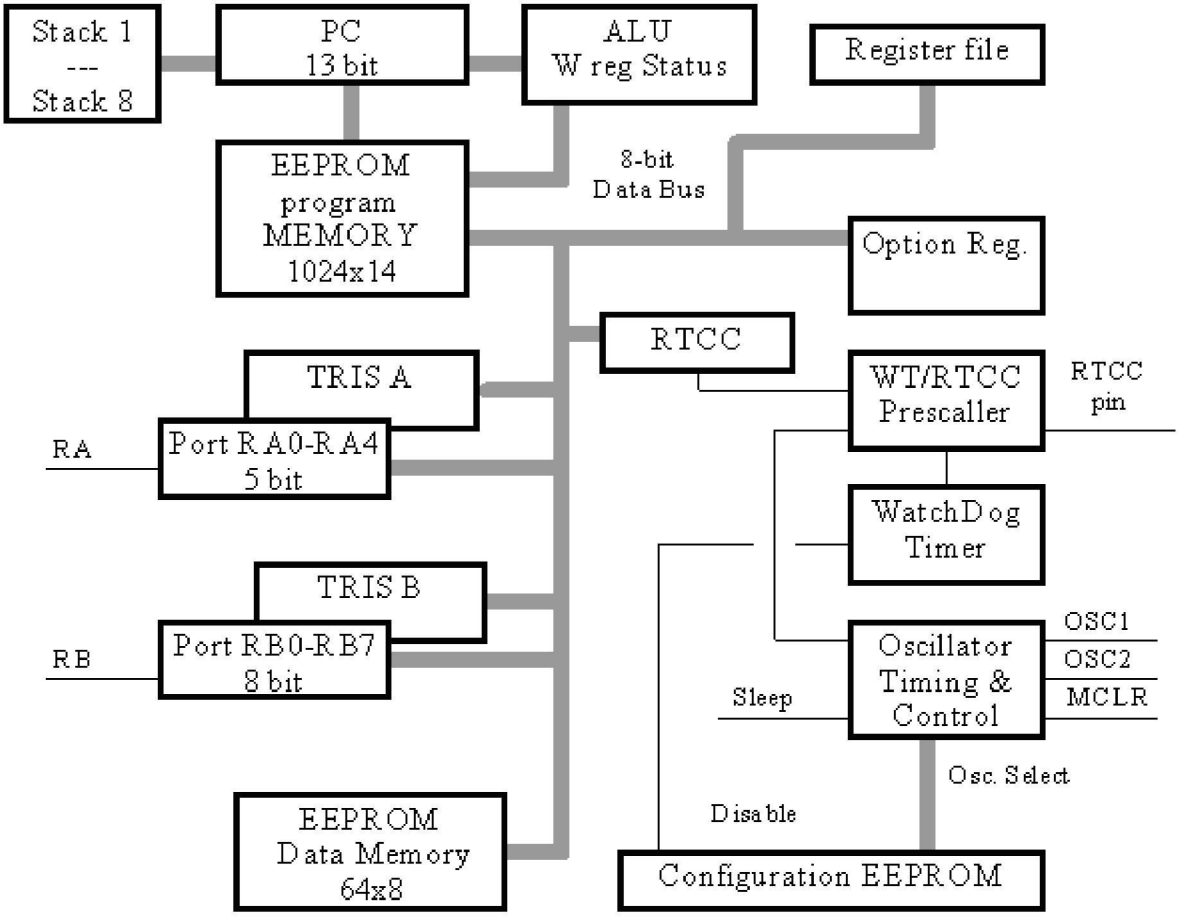

Рисунок 2 – Функциональная схема МК PIC16F84

В регистре состояния STATUS (03h) есть биты выбора страниц, которые позволяют обращаться к четырем страницам будущих модификаций этого кристалла. Однако для PIC16F84 память данных существует только до адреса 02Fh. Первые 12 адресов используются для размещения регистров специального назначения. Регистры с адресами 0Ch-2Fh могут быть использованы как регистры общего назначения, которые представляют собой статическое ОЗУ. Некоторые регистры специального назначения продублированы на обеих страницах,

а некоторые расположены на странице 1 отдельно. Когда установлена страница 1, то обращение к адресам 8Ch-AFh фактически адресует страницу 0. К регистрам можно адресоваться прямо или косвенно. В обоих случаях можно адресовать до пятисот двенадцати регистров. Роль аккумулятора выполняет регистр W, называемый рабочим регистром.

Регистр состояния содержит арифметические флаги АЛУ, состояние контроллера при сбросе и биты выбора страниц для памяти данных. Размещение флагов в регистре представлено в таблице 3. Он доступен для любой команды так же, как любой другой регистр. Однако биты TO и PD устанавливаются аппаратно и не могут быть записаны в статус программно. Это следует иметь в виду при выполнении команды с использованием регистра статуса. Например, команда CLRF f3 обнулит все биты, кроме бит TO и PD, а затем установит бит Z = 1. После выполнения этой команды регистр статуса может и не иметь нулевое значение (из-за бит TO и PD). Поэтому рекомендуется для изменения регистра состояния использовать только команды битовой установки BCF, BSF, MOVWF, которые не изменяют остальные биты.

Таблица 3 – Размещение флагов в регистре статуса

7	6	5	4	3	2	1	0
IRP	RP1	RP0	TO	PD	Z	DC	C

C – Флаг переноса/заема:

Для команд ADDWF и SUBWF. Этот бит устанавливается, если в результате операции из самого старшего разряда происходит перенос. Вычитание осуществляется путем прибавления дополнительного кода второго операнда. При выполнении команд сдвига этот бит всегда загружается из младшего или старшего бита сдвигаемого источника.

DC – Флаг десятичного переноса/заёма:

Для команд ADDWF и SUBWF. Этот бит устанавливается, если в результате операции из четвертого разряда происходит перенос. Механизм установки десятичного бита переноса «DC» тот же самый, отличается тем, что отслеживается перенос из четвертого бита (т. е. перенос из младшей тетрады).

Z – Флаг нулевого результата:

Устанавливается, если результатом арифметической или логической операции является ноль.

PD – Power Down (режим хранения данных):

Устанавливается в «1» при включении питания или команде CLRWDT. Сбрасывается в «0» командой SLEEP.

TO – Time Out. Флаг срабатывания Watchdog таймера:

Устанавливается в «1» при включению питания и командами CLRWDT, SLEEP. Сбрасывается в «0» по завершению выдержки времени таймера WDT.

RP1, RP0 – Биты выбора страницы памяти данных при прямой адресации.

Значения битов RP1,RP0:

00 = Страница 0 (00h-7Fh),

01 = Страница 1 (80h-FFh),
10 = Страница 2 (100h-17Fh),
11 = Страница 3 (180h-1FFh).

Каждая страница содержит 128 байт. В кристалле PIC16C84 используется только RP0. В этом кристалле RP1 может использоваться просто как бит общего назначения чтения/записи. Однако надо помнить, что в других контроллерах семейства PIC16 он может использоваться.

IRP – Бит выбора страницы памяти данных при косвенной адресации.

IRP0:

0 = Страницы 0,1 (00h-FFh),
1 = Страница 2,3 (100h-1FFh).

1.2 Линии порта А

Порт А – это порт шириной 5 бит, соответствующие выводы микросхемы RA<4:0>. Линии RA<3:0> двунаправленные, а линия RA4 – выход с открытым стоком. Адрес регистра порта А – 05h. Относящийся к порту А управляющий регистр TRISA расположен на первой странице регистров по адресу 85h. TRISA<4:0> – это регистр шириной 5 бит. Если бит управляющего TRISA регистра имеет значение «1», то соответствующая линия будет устанавливаться на ввод. Ноль переключает линию на вывод и одновременно выводит на нее содержимое соответствующего регистра-защелки (таблица 4).

Таблица 4 – Функции линий порта А

Название ножки	Функция ножки	Другие функции
PA0	Порт ввода/вывода. Входные уровни ТТЛ	–
PA1	Порт ввода/вывода. Входные уровни ТТЛ	–
PA2	Порт ввода/вывода. Входные уровни ТТЛ	–
PA3	Порт ввода/вывода. Входные уровни ТТЛ	–
PA4/RT	Порт ввода/вывода. Выход – открытый коллектор. Вход – триггер Шмитта	Вход внешнего тактового сигнала для RTCC

1.3 Линии порта В

Порт В – это двунаправленный порт шириной в восемь бит (адрес регистра 06h). Функции линий порта указаны в таблице 5. Относящийся к порту В управляющий регистр TRISB расположен на первой странице регистров по адресу 86h. Если бит управляющего TRISB регистра имеет значение «1», то соответствующая линия будет устанавливаться на ввод. Ноль переключает линию на вывод и одновременно выводит на нее содержимое соответствующего регистра защелки. У каждой линии порта В имеется небольшая активная нагрузка

(ток около 100 мкА) на линию питания («подтягивающий» резистор). Она автоматически отключается, если эта линия как вывод. Более того, установленный управляющий бит RBPU OPTION<7> может отключить все нагрузки. Сброс при включении питания также отключает все нагрузки. Четыре линии порта B (RB<7:4>) имеют способность вызвать прерывание при изменении значения сигнала на любой из них. Если эти линии настроены на ввод, то они опрашиваются и защелкиваются в цикле чтения Q1. Новая величина входного сигнала сравнивается со старой в каждом командном цикле. При несовпадении значения сигнала на ножке и в защелке генерируется высокий уровень. Выходы детекторов «несовпадений» RB4,RB5,RB6,RB7 объединяются по ИЛИ и генерируют прерывание RBIF (запоминаемое в INTCON<0>). Любая линия, настроенная на вывод, не участвует в этом сравнении. Прерывание может вывести контроллер из режима SLEEP. В подпрограмме обработки прерывания следует сбросить запрос прерывания одним из следующих способов:

1) запретить прерывания при помощи обнуления бита RBIE INTCON<3>;
2) прочитать содержимое порта B. Это завершит состояние сравнения;
3) обнулить бит RBIF INTCON<0>.

Прерывание по несовпадению и программно устанавливаемые внутренние активные нагрузки на этих четырех линиях могут обеспечить простой интерфейс, например, с клавиатурой, с выходом из режима SLEEP по нажатию клавиш. Линия RB0 совмещена с входом внешнего прерывания INT.

Запись в порт вывода происходит в конце командного цикла. Но при чтении данные должны быть стабильны в начале командного цикла. Здесь важно учитывать инерционность установления напряжения на выводах. Может потребоваться программная задержка, чтобы напряжение на ножке успело стабилизироваться до начала исполнения следующей команды чтения.

Таблица 5 – Функции линий порта B

Название ножки	Функция ножки	Другие функции
PB0	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	Вход внешнего прерывания
PB1	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	–
PB2	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	–
PB3	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	–
PB4	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	Прерывание при изменении
PB5	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	Прерывание при изменении
PB6	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	Прерывание при изменении
PB7	Порт ввода/вывода. Входные уровни ТТЛ и внутренняя программируемая активная нагрузка	Прерывание при изменении

1.4 Система команд PIC-контроллера PIC16F84

Система команд PIC16F84 включает 37 команд. Каждая команда – это 14-битное слово, которое разделено по смыслу на следующие части: 1 – код операции, 2 – поле для одного и более операндов, которые могут участвовать в этой команде. Система команд PIC16F84 включает в себя байт-ориентированные команды, бит-ориентированные команды, команды операций с константами и команды передачи управления.

В мнемонике команд для байт-ориентированных команд f обозначает регистр, с которым производится действие; d – определяет, где сохраняется результат. Если d=0, то результат будет помещен в W регистр, при d=1 результат будет помещен в f, упомянутом в команде. Для бит-ориентированных команд b обозначает номер бита, участвующего в команде, а f – это регистр, в котором этот бит расположен.

Для команд передачи управления и операций с константами, k обозначает 8- или 11-битную константу.

Все команды выполняются в течение одного командного цикла. В двух случаях исполнение команды занимает два командных цикла: 1 – проверка условия и переход, 2 – изменение программного счетчика как результат выполнения команды. Один командный цикл состоит из четырех периодов сигнала тактового генератора. Таким образом, для генератора с частотой 4 МГц время исполнения командного цикла будет 1 мкс.

Таблица 6 – Байт-ориентированные команды

Мнемокод	Описание	Флаги	Примечание

ADDWF f,d	Сложение содержимого регистров W и f
	Если d = 0, результат сохраняется в регистре W.
	Если d = 1, результат сохраняется в регистре f.
	Пример:
	ADDWF REG,0
	До выполнения: W = 0x17; REG = 0xC2
	После выполнения: W = 0xD9; REG = 0xC2
	Косвенная адресация: для ее выполнения необходимо обратиться к регистру INDF. Это вызовет действие с регистром, адрес которого указан в регистре FSR.
	Запись в регистр INDF не вызовет никаких действий (кроме воздействия на флаги в регистре STATUS).
	Чтение INDF (FSR = 0) даст результат 00h.
	Пример:
	ADDWF INDF,1
	До выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x20)
	После выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x37)
	Изменение адреса счетчика команд PC (**вычисляемый переход**) выполняется командой приращения к регистру PCL (ADDWF PCL,1)		

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
	При этом необходимо следить, чтобы не произошло пересечения границы между блоками памяти программ (в блоке 256 слов). PCL – младший байт (8 бит <7:0>) счетчика команд (PC), доступен для чтения и записи. PCH – старший байт (5 бит <12:8>) счетчика команд PC, не доступен для чтения и записи. Все операции с PCH происходят через дополнительный регистр PCLATH. В случае вычисляемого перехода при переполнении PCL инкремента PCH не происходит. *Примеры*: 1) ADDWF PCL,1 До выполнения: W = 0x10; PCL = 0x37 После выполнения: PCL = 0x47; C = 0 2) ADDWF PCL,1 До выполнения: W = 0x10; PCL = 0xF7; PCH = 0x08 После выполнения: PCL = 0x07; PCH = 0x08; C = 1	C,DC,Z	2,3
ANDWF f,d	Логическое И содержимого регистров W и f *Примеры*: 1) ANDWF REG,1 До выполнения: W = 0x17; REG = 0xC2 После выполнения: W = 0x17; REG = 0x02; 2) ANDWF REG,0 До выполнения: W = 0x17; REG = 0xC2 После выполнения: W = 0x02; REG = 0xC2 **Косвенная адресация:** ANDWF INDF,1 До выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x5A) После выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x12)	Z	2,3
CLRF f	Очистить содержимое регистра f и установить флаг Z *Пример*: CLRF REG До выполнения: REG = 0x5A После выполнения: REG = 0x00; Z = 1 **Косвенная адресация:** CLRF INDF До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0xAA) После выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x00); Z = 1	Z	3
CLRW	Очистить содержимое регистра W и установить флаг Z *Пример*: CLRW До выполнения: W = 0x5A После выполнения: W = 0x00; Z = 1	Z	

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
COMF f,d	Инвертировать все биты в регистре f *Примеры*: 1) COMF REG,0 До выполнения: REG = 0x13 После выполнения: REG = 0x13; W = 0xEC; 2) COMF REG,1 До выполнения: REG = 0xFF После выполнения: REG = 0x00; Z = 1 **Косвенная адресация:** COMF INDF,1 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0xAA); После выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x55)	Z	2,3
DECF f,d	Декремент содержимого регистра f *Примеры*: 1) DECF REG,1 До выполнения: REG = 0x01; Z = 0 После выполнения: REG = 0x00; Z = 1; DECF REG,0 До выполнения: REG = 0x10; W = x; Z = 0 После выполнения: REG = 0x10; W = 0x0F; Z = 0 **Косвенная адресация:** DECF INDF,1 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x01) После выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x00); Z = 1	Z	2,3
DECFSZ f,d	Декремент содержимого регистра f с ветвлением Если результат не равен 0, то исполняется следующая инструкция. Если результат равен 0, то следующая инструкция не исполняется (вместо нее – «виртуальный» NOP), а команда исполняется за 2 м.ц. *Примеры*: LOOP DECFSZ REG,1; GOTOLOOP; CONTINUE .... 1. До выполнения REG=0x01 После выполнения: REG=0x00; PC=адрес CONTINUE 2. До выполнения REG=0x02 После выполнения REG=0x01 Переход на LOOP	–	2,3
INCF f,d	Инкремент содержимого регистра f *Примеры*: 1) INCF REG,1 До выполнения: REG = 0xFF; Z = 0 После выполнения: REG = 0x00; Z = 1 2) INCF REG,0 До выполнения: REG = 0x10; W = x; Z = 0 После выполнения REG = 0x10; W = 0x11; Z = 0 **Косвенная адресация:** INCF INDF,1	Z	2,3

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
	До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0xFF); Z = 0 После выполнения FSR = 0xC2 (по этому адресу «лежит» число 0x00); Z = 1		
INCFZ f,d	Инкремент содержимого регистра f с ветвлением Если результат не равен 0, то исполняется следующая инструкция. Если результат равен 0, то следующая инструкция не исполняется (вместо нее исполняется «виртуальный» NOP), а команда исполняется за 2 м.ц. *Примеры*: LOOP INCFSZ REG,1; GOTO LOOP CONTINUE ..... 1. До выполнения: REG = 0xFF После выполнения: REG = 0x00; PC = адрес CONTINUE 2. До выполнения: REG = 0x02 После выполнения: REG = 0x03 Переход на LOOP	–	2,3
IORWF f,d	Логическое ИЛИ содержимого регистров W и f *Примеры*: 1) IORWF REG,0 До выполнения: REG = 0x13; W = 0x91 После выполнения: REG = 0x13; W = 0x93; Z = 0 2) IORWF REG,1 До выполнения: REG = 0x13; W = 0x91 После выполнения: REG = 0x93; W = 0x91; Z = 0 3) IORWF REG,1 До выполнения REG = 0x00; W = 0x00 После выполнения REG = 0x00; W = 0x004; Z = 1 **Косвенная адресация:** IORWF INDF,1 До выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x30) После выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x37); Z = 0	Z	2,3

MOVF f,d	Содержимое регистра f пересылается в регистр адресата
	Примеры:
	MOVF REG,0
	До выполнения: W = 0x00; REG = 0xC2
	После выполнения: W = 0xC2; REG = 0xC2; Z = 0
	MOVF REG,1
	1. До выполнения REG = 0x43
	После выполнения REG = 0x43; Z = 0
	2. До выполнения REG = 0x00
	После выполнения REG = 0x00; Z = 1
	Косвенная адресация: MOVF INDF,1
	До выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x00)
	После выполнения: W = 0x17; FSR = 0xC2 (по этому адресу «лежит» число 0x00); Z = 1
		Z	2,3

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
MOVWF f	Пересылка содержимого W в f *Примеры*: MOVWF REG До выполнения REG=0xFF; W=0x4F После выполнения REG=0x4F; W=0x4F **Косвенная адресация:** MOVWF INDF До выполнения W=0x17 FSR=0xC2 (по этому адресу «лежит» число 0x00) После выполнения W=0x17 FSR=0xC2 (по этому адресу «лежит» число 0x17)	–	3
NOP	Нет операции Пример: NOP До выполнения: PC = адрес X После выполнения: PC=адрес X + 1	–	
RLF f,d	Выполняется циклический сдвиг влево содержимого регистра f через бит C регистра STATUS *Примеры*: RLF REG,0 До выполнения: REG = 11100110; W = xxxxxxxx; C = 0 После выполнения: REG = 11100110; W = 11001100; C = 1 **Косвенная адресация:** RLF INDF,1 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x3A → 00111010); C = 1 После выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x75 → 01110101); C = 0 RLF INDF,1 До выполнения FSR = 0xC2 (по этому адресу «лежит» число 0xB9 → 10111001); C = 0 После выполнения FSR = 0xC2 (по этому адресу «лежит» число 0x72 → 01110010); C = 1	C	2,3
RRF f,d	Выполняется циклический сдвиг вправо содержимого регистра f через бит C регистра STATUS *Примеры*: RRF REG,0 До выполнения REG = 11100110; W=xx; C=0 После выполнения REG = 11100110; W = 01110011; C = =0 **Косвенная адресация:** RRF INDF,1 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x3A → 00111010); C = 1 После выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x9D → 10011101); C=0 RRF INDF,1 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x39 → 00111001); C = 0 После выполнения: FSR=0xC2 (по этому адресу «лежит» число 0x1C → 00011100); C = 1	C	2,3

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
SUBWF f,d	Вычесть содержимое регистра W из содержимого регистра f *Примеры*: 1) SUBWF REG,1 До выполнения: REG = 0x03; W = 0x02; C = x; Z = x После выполнения: REG = 0x01; W = 0x02; C = 1; Z = 0 («+» результат); 2) SUBWF REG,1 До выполнения: REG = 0x02; W = 0x02; C = x; Z = x После выполнения REG = 0x00; W = 0x02; C = 1, Z = 1 («0» результат) 3) SUBWF REG,1 До выполнения: REG = 0x01; W = 0x02; C = x; Z = x После выполнения: REG = 0xFF; W = 0x02; C = 0, Z = 0 («–» результат)	C,DC, Z	2,3
SWAPF f,d	Поменять местами старший и младший полубайты регистра f *Примеры*: 1) SWAPF REG,0 До выполнения: REG=0xA5 → 1010 0101; W = x После выполнения: REG = 0xA5; W = 0x5A → 0101 1010; 2) SWAPF REG,1 До выполнения: REG = 0xA5 После выполнения: REG = 0x5A **Косвенная адресация:** SWAPF INDF,1 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x20 → 0010 0000 После выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x02 → 0000 0010	–	2,3
XORWF f,d	Исключающее ИЛИ содержимого регистров W и f (проверка на одинаковость) *Примеры*: 1) XORWF REG,1 До выполнения REG = 0xAF; W = 0xB5 После выполнения: REG = 0x1A; W = 0xB5; XORWF REG,0 До выполнения REG = 0xAF; W = 0xB5 После выполнения REG = 0xAF; W = 0x1A **Косвенная адресация:** XORWF INDF,1 До выполнения: W = 0xB5; FSR = 0xC2 (значение регистра с адресом в FSR = 0xAF) После выполнения W = 0xB5; FSR = 0xC2 (значение регистра с адресом в FSR = 0x1A) Содержимое регистра W складывается с 8-разрядной константой k Результат сохраняется в регистре W	Z	2,3

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
ADDLW k	*Примеры*: 1) ADDLW 0x15 До выполнения: W = 0x10 После выполнения: W = 0x25 2) ADDLW REG До выполнения W = 0x10; REG = 0x37 (адрес регистра, а не его содержимое) После выполнения W = 0x47 3) ADDLW CONST До выполнения «Прописка в шапке» программы: CONST EQU 0x37; W=0x10 После выполнения W=0x47	C,DC, Z	
ANDLW k	Логическое И содержимого регистра W и 8-разрядной константы k Результат сохраняется в регистре W *Примеры*: 1) ANDLW 0x5F → 01011111 До выполнения W = 0xA3 → 10100011 После выполнения W = 0x03 → 00000011 2) ANDLW REG До выполнения W = 0xA3 REG=0x37 (адрес регистра, а не его содержимое) После выполнения: W = 0x23 3) ANDLW CONST До выполнения: «Прописка в шапке» программы: CONST EQU 0x37; W = 0xA3 После выполнения: W = 0x23	Z	
IORLW k	Логическое ИЛИ содержимого регистра W и 8-разрядной константы k Результат сохраняется в регистре W *Примеры*: 1) IORLW 0x35 → 00110101 До выполнения: W = 0x9A → 10011010 После выполнения: W = 0xBF → 10111111; Z = 0 2) IORLW REG До выполнения: W = 0x9A REG=0x37 (адрес регистра, а не его содержимое) После выполнения: W = 0xBF; Z = 0 3) IORLW CONST До выполнения: W = 0x9A «Прописка в шапке» программы: CONST EQU 0x37 После выполнения: W = 0x9F; Z = 0 4) IORLW 0x00 До выполнения: W = 0x00 После выполнения: W = 0x00; Z = 1	Z	

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
SUBLW k	Вычесть содержимое регистра W из 8-разрядной константы k Результат сохраняется в регистре W *Примеры*: 1) SUBLW 0x02 До выполнения: W = 0x01; C = ? Z = ? После выполнения W = 0x01; C = 1, Z = 0; («+» результат) 2) SUBLW 0x02 До выполнения: W = 0x03; C = ?; Z = ? После выполнения: W = 0xFF; C = 0; Z = 0 («–» результат) 3) SUBLW 0x02 До выполнения: W = 0x02; C = ?; Z = ? После выполнения W = 0x00; C = 1; Z = 1 («0» результат) 4) SUBLW REG До выполнения: W = 0x10; REG = 0x37 (адрес регистра, а не его содержимое) После выполнения: W = 0x27; C = 1	C,DC,Z	
MOVLW k	Пересылка константы k в регистр W *Примеры*: 1) MOVLW 0x5A До выполнения: W = x После выполнения: W = 0x5A 2) MOVLW REG До выполнения: W = x; REG=0x37 (адрес регистра, а не его содержимое) После выполнения: W = 0x37; Z = 0 3) MOVLW CONST До выполнения: «Прописка в шапке» программы: CONST EQU 0x37; W=x После выполнения: W=0x37	–	
XORLW k	Исключающее ИЛИ содержимого регистра W и 8-разрядной константы k (проверка на «одинаковость») Результат сохраняется в регистре W *Примеры*: 1) XORLW 0xAF → 10101111 До выполнения: W = 0xB5 → 10110101 После выполнения: W = 0x1A → 00011010; Z = 0; 2) XORLW REG До выполнения: W = 0xAF; REG = 0x37 (адрес регистра, а не его содержимое) После выполнения: W = 0x98; Z = 0; XORLW CONST До выполнения W = 0xAF «Прописка в шапке» программы: CONST EQU 0x37 После выполнения: W = 0x18; Z = 0	Z	

Продолжение таблицы 6

Мнемокод	Описание	Флаги	Примечание
CALL	Выполнить условный переход Адрес следующей инструкции (PC+1) загружается в вершину стека. 11 бит адреса загружаются в счетчик команд из кода команды. 2 старших бита загружаются в счетчик команд из регистра PCLATH. Пример: HERE CALL ABC До выполнения PC = адрес HERE После выполнения PC = адрес ABC В вершине стека, адрес HERE+1		
GOTO k	Выполнить безусловный переход 11 бит адреса загружаются в счетчик команд из кода команды. 2 старших бита загружаются в счетчик команд из регистра PCLATH. *Пример*: GOTO ABC После выполнения PC= адрес ABC		
RETURN	Возврат из подпрограммы Содержимое вершины стека «выгружается» в счетчик команд PC.		
OPTION	Переслать содержимое регистра W в регистр OPTION. Инструкция поддерживается для совместимости программы с семейством PIC16C5x. Запись – чтение регистра OPTION можно выполнить прямой или косвенной адресацией. Не рекомендуется использовать при работе с другими (отличными от PIC16C5x) типами ПИКов.	–	1
TRIS	Переслать содержимое регистра W в регистр TRIS. Не рекомендуется использовать при работе с другими (отличными от PIC16C5x) типами ПИКов	–	1

Набор команд, используемых при работе с битами, представлен в таблице 7.

Таблица 7 – Бит-ориентированные команды

Мнемокод	Описание
BCF f,b	Установить в 0 бит b регистра f *Примеры*: BCF REG,7 До выполнения: REG = 0xC7 → 11000111 После выполнения: REG = 0x47 → 01000111 **Косвенная адресация:** BCF INDF,3 До выполнения FSR = 0xC2 (по этому адресу «лежит» число 0x2F → 00101111) После выполнения FSR=0xC2 (по этому адресу «лежит» число 0x27 → 00100111)

Продолжение таблицы 7

Мнемокод	Описание
BSF f,b	Установить в 1 бит b регистра f *Примеры*: BSF REG,7 До выполнения REG = 0x0A → 00001010 После выполнения REG = 0x8A → 10001010 **Косвенная адресация:** BSF INDF,3 До выполнения: FSR = 0xC2 (по этому адресу «лежит» число 0x20 → 00100000) После выполнения FSR = 0xC2 (по этому адресу «лежит» число 0x28 → 00101000)
BTFSC f,b	Если бит b в регистре f =1, то исполняется следующая инструкция Если бит b в регистре f =0, то следующая инструкция не исполняется (пропускается, вместо нее исполняется «виртуальный» NOP), а команда исполняется за 2 м.ц. *Примеры*: BTFSC REG,4; GOTO LOOP; TRUE ……. 1. До выполнения REG=xxx0xxxx После выполнения, так как REG<4>=0, исполняется TRUE 2. До выполнения REG=xxx1xxxx После выполнения, так как REG<4>=1, исполняется GOTO LOOP
BTFSS f,b	Если бит b в регистре f=0, исполняется следующая инструкция Если бит b в регистре f=1, то следующая инструкция не исполняется (пропускается, вместо нее исполняется «виртуальный» NOP), а команда исполняется за 2 м.ц. *Примеры*: BTFSS REG,4; GOTO LOOP; TRUE ……. 1. До выполнения REG=xxx0xxxx После выполнения, так как REG<4>=0, исполняется GOTO LOOP 2. До выполнения REG=xxx1xxxx После выполнения, так как REG<4>=1, исполняется TRUE

Примечания –

1 Команды TRIS и OPTION помещены в перечень команд для совместимости с семейством PIC16C5X. Их использование не рекомендуется. В PIC16C84 регистры TRIS и OPTION доступны для чтения и записи как обычные регистры с номером. Предупреждаем, что эти команды могут не поддерживаться в дальнейших разработках PIC16CXX.

2 Когда модифицируется регистр ввода/вывода, например MOVF 6,1, значение, используемое для модификации, считывается непосредственно с ножек кристалла. Если значение защелки вывода для ножки, запрограммированной на вывод, равно «1», но внешний сигнал на этом выводе «0» из-за «навала» снаружи, то будет считываться «0».

3 Если операндом этой команды является регистр f1 (и, если допустимо, d=1), то делитель, если он подключен к RTCC, будет обнулен.

2 СИМУЛЯТОР PIC-КОНТРОЛЛЕРОВ MULTISIM

В данном разделе работа с программой Multisim описывается для случаев работы с микроконтроллерами. Подробнее ознакомиться с интерфейсом программы Multisim позволяет первая часть лабораторного практикума [1].

Multisim содержит программный модуль MCU, позволяющий симулировать микропроцессоры. Из семейства PIC-контроллеров программа позволяет работать с контроллерами PIC16F84 и PIC16F84A. Для того чтобы активизировать эти компоненты, нужно в окне «Выбор компонентов» выбрать группу компонентов «MCU Module», а затем выбрать компонент PIC16F84 (рисунок 3).

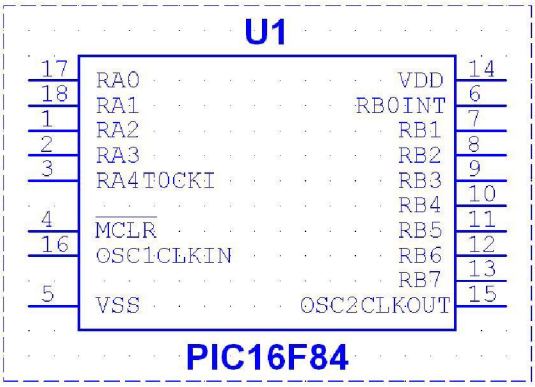

Рисунок 3 – Условно-графическое обозначение PIC-контроллера

Свойства контроллера можно изменять, раскрыв окно параметров (рисунок 4).

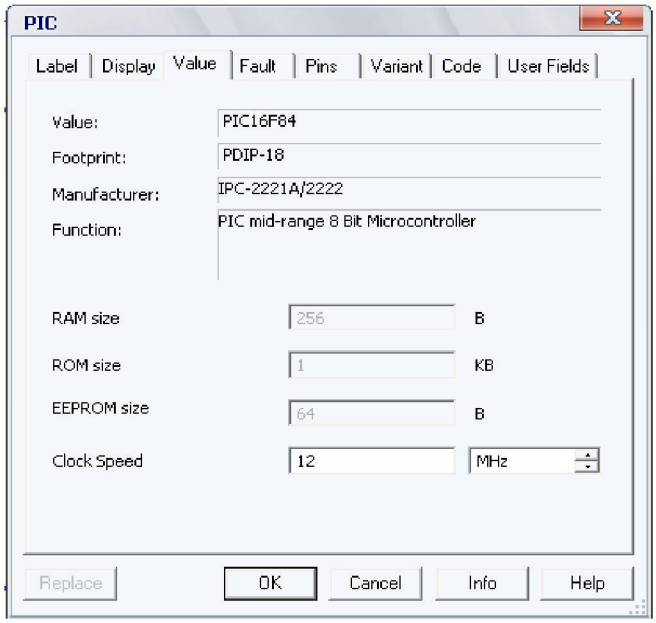

Рисунок 4 – Окно параметров PIC-контроллера в развернутом виде

Изменять параметры контроллера можно в поле «Value» (см. рисунок 4).

Название, тип и подключение выводов контроллера можно определить в поле «Pins» (рисунок 5).

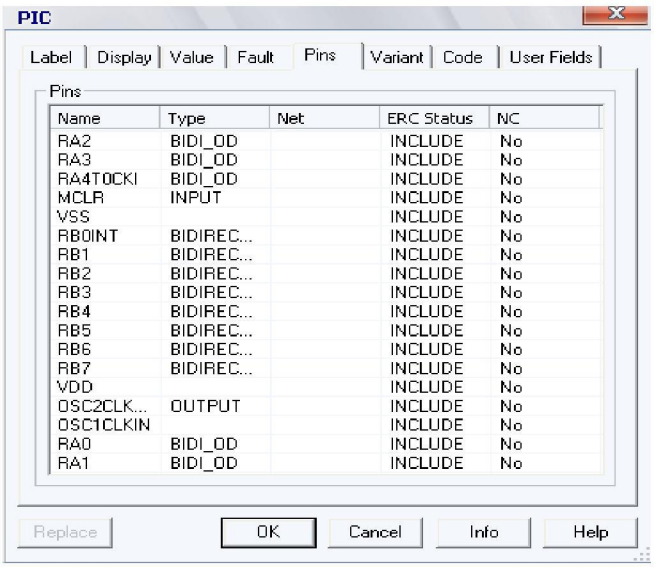

Рисунок 5 – PIC-контроллер в развернутом виде

В поле «Code» после нажатия кнопки Properties появляется окно «MCU Code Manager». В нем создается новый проект, в котором можно создать программу и запрограммировать контроллер (рисунок 6).

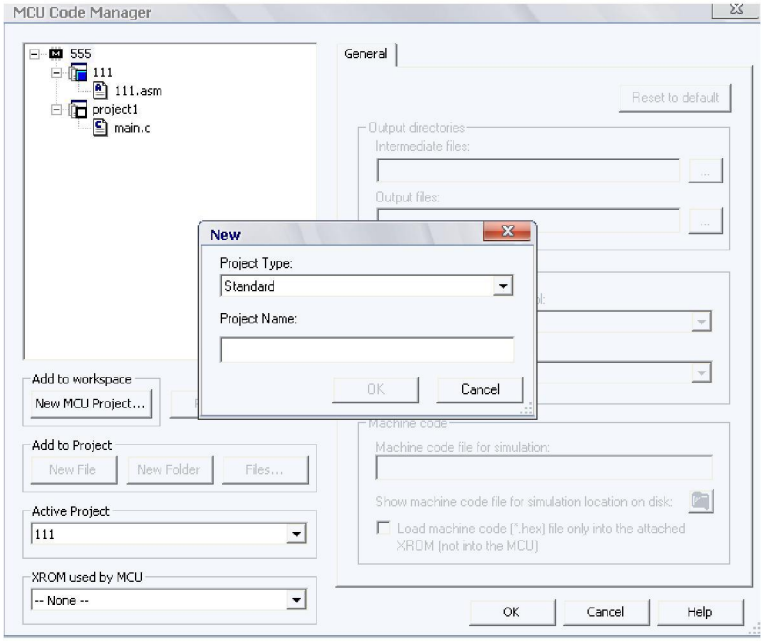

Рисунок 6 – Окно MCU Code Manager

Таким образом, интерфейс программы Multisim позволяет использовать цифровые и микропроцессорные устройства в одном модуле.

По умолчанию программирование микроконтроллера осуществляется на языке С. В данном лабораторном практикуме лабораторные работы будут выполняться с использованием языка Ассемблера. Далее приводится алгоритм перехода с языка С на язык Ассемблера.

Правой кнопкой мыши щелкаем по изображению микроконтроллера, выбираем меню Properties, заходим в закладку Code, открываем окно Properties. Откроется окно, изображенное на рисунке 6.

В левом верхнем углу выбираем «main.c» и нажимаем Remove Selected. Далее выбираем «project1», и во вкладке Select assembler/compiler tool вместо «Hi-Tech PICC-Lite compiler» выбираем «Microchip MPASM for PIC 16». Теперь в области «Add to Project» нажимаем кнопку «New File». Открывается окно (рисунок 7).

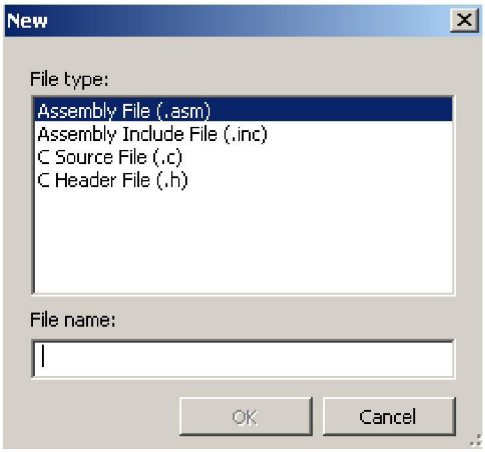

Рисунок 7 – Окно выбора языка программирования для файла программы

Выбираем «Assembly File (.asm)», в строке File name пишем имя файла и нажимаем «OK».

При работе с симулятором можно, зайдя в пункт меню «MCU», включить отображение окон MCU Windows, отображающих содержимое регисторв, памяти программ и данных, ячеек стека и бит конфигурации. Эта информация полезна при отладке программного обеспечения. Кнопки режимов отладки (трассировка, пошаговое выполнение и т.д.) находятся вверху панели инструментов под основным меню.