Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

А. Л. Гурский, Н. А. Певнева

ЦИФРОВЫЕ И МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА СРЕДСТВ ИЗМЕРЕНИЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 2-х частях

Часть 2

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

Рекомендовано УМО по образованию в области информатики и радиоэлектроники для специальности 1-54 01 04 «Метрологическое обеспечение информационных систем и сетей» в качестве учебно-методического пособия

Минск БГУИР 2013

УДК 004.31-022.53(076.5)
ББК 32.973.26-04я73
Г95

Р е ц е н з е н т ы:
кафедра информационно-измерительной техники и технологий
Белорусского национального технического университета
(протокол №2 от 18.09.2012);

доцент кафедры ядерной физики Белорусского государственного
университета, кандидат технических наук, доцент М. В. Комар

Гурский, А. Л.

Г95 Цифровые и микропроцессорные устройства средств измерений. Лабораторный практикум : учеб.-метод. пособие. В 2 ч. Ч. 2 : Микропроцессорные устройства / А. Л. Гурский, Н. А. Певнева. – Минск : БГУИР, 2013 – 48 с. : ил.

ISBN 978-985-488-905-4 (ч. 2).

Вторая часть учебно-методического пособия содержит лабораторный практикум по основам микропроцессорной техники. В его основу положена работа с моделью 8-разрядного PIC-микроконтроллера PIC16F84 в программной среде NI Multisim 10.0. Включает краткую информацию по архитектуре, структуре и набору команд PIC16F84, описание приемов работы с модулем MCU пакета Multisim 10.0 и 8 лабораторных работ для изучения системы команд и основных приемов программирования PIC-контроллеров. Основное внимание уделено использованию управляющих функций контроллеров, формированию временных интервалов, приемам создания ветвящихся алгоритмов, битовым операциям и выводу на индикаторные устройства.

УДК 004.31-022.53(076.5)
ББК 32.973.26-04я73

Часть 1 «Цифровые устройства» издана в БГУИР в 2011 г.

ISBN 978-985-488-905-4 (ч. 2)
ISBN 978-985-488-575-9

Содержание

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 1K×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.

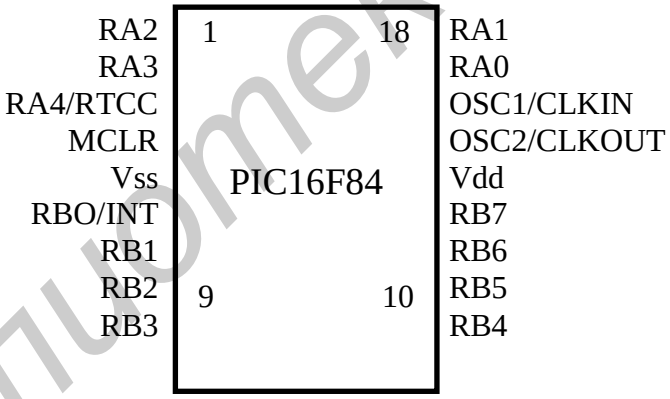


Рисунок 1 – Обозначения выводов PIC16F84

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

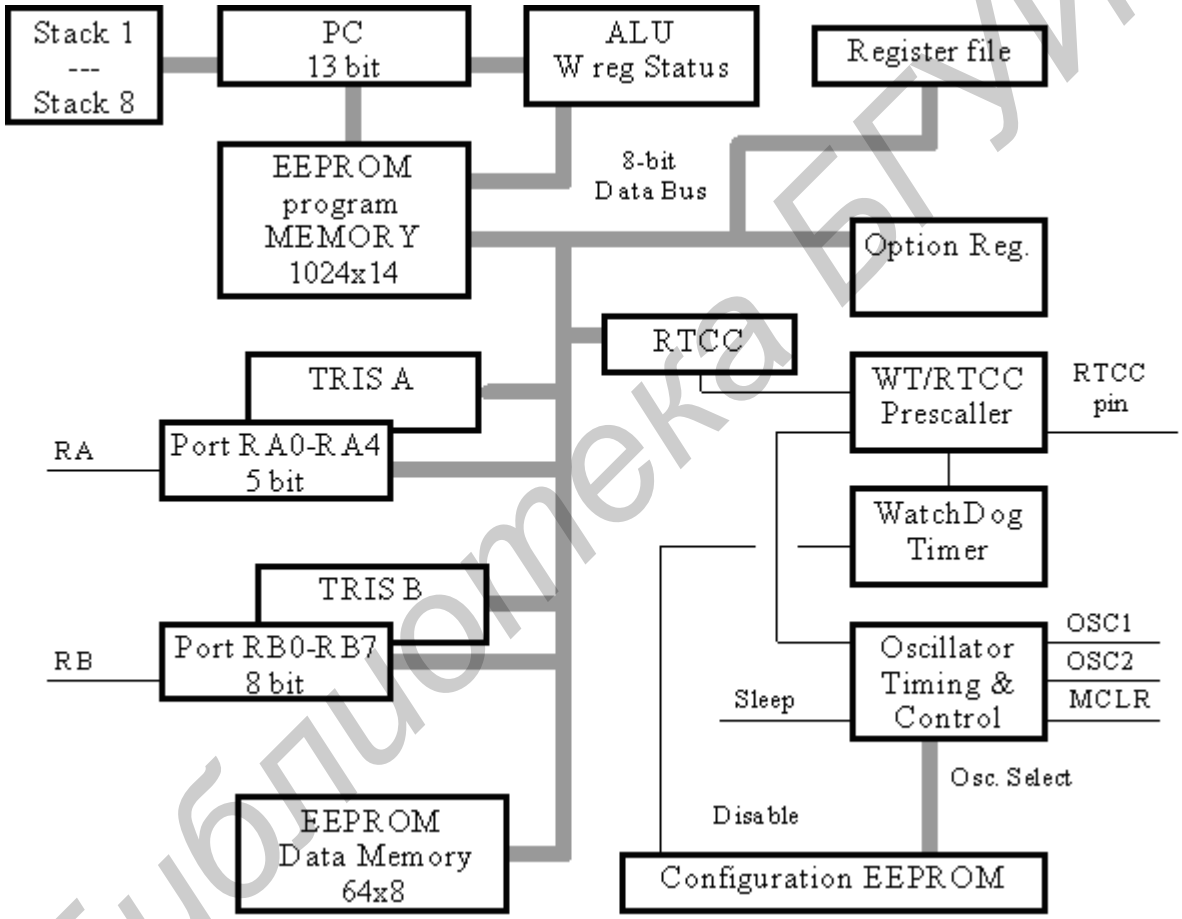


Рисунок 2 – Функциональная схема МК PIC16F84

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).

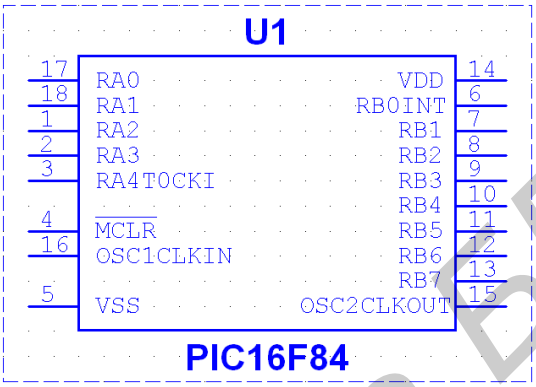


Рисунок 3 – Условно-графическое обозначение PIC-контроллера

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

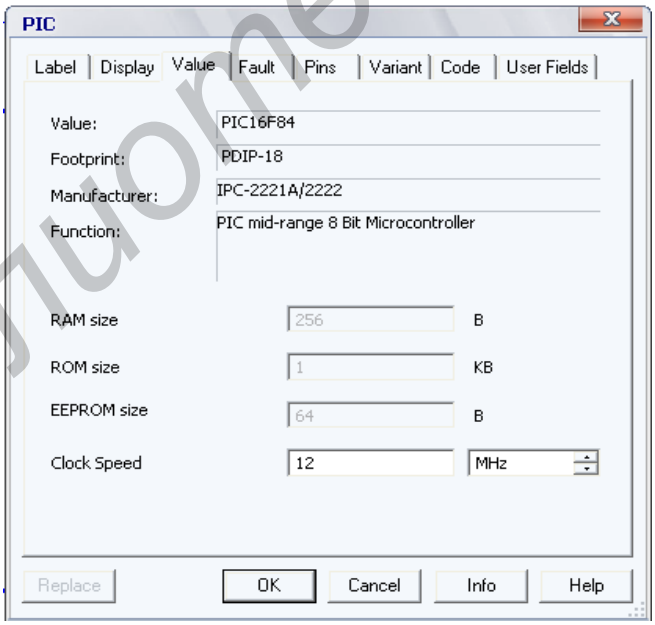


Рисунок 4 – Окно параметров PIC-контроллера в развернутом виде

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

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

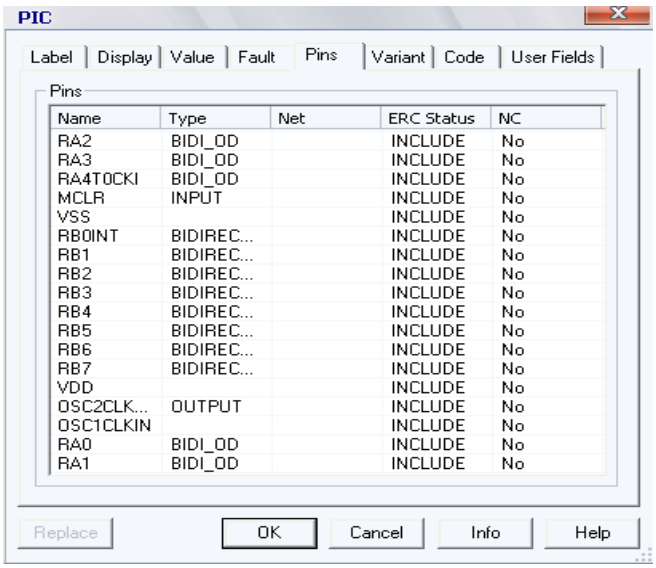


Рисунок 5 – PIC-контроллер в развернутом виде

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).

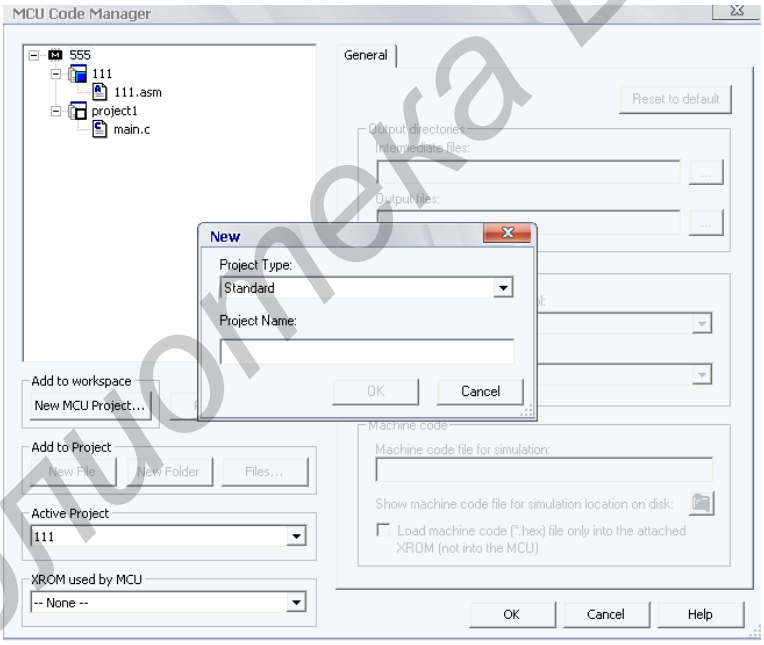


Рисунок 6 – Окно MCU Code Manager

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).

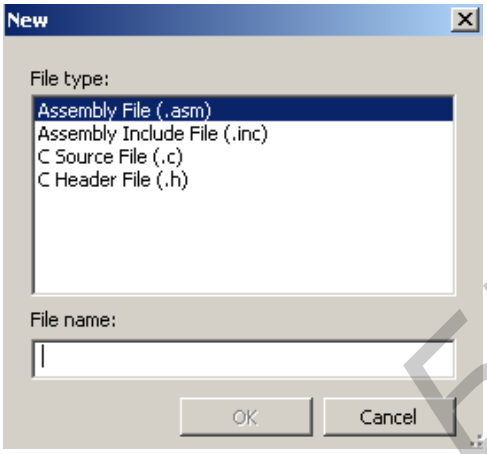


Рисунок 7 – Окно выбора языка программирования для файла программы

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.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ЛАБОРАТОРНАЯ РАБОТА №1

ОЗНАКОМЛЕНИЕ С РАБОТОЙ ПРОГРАММЫ MULTISIM

Цель работы – ознакомление с программным симулятором Multisim, изучение элементов анализа схем и основных функций этого программного пакета, используемых при моделировании и отладке схем с мироконтроллерами.

Теоретическая часть. Ознакомьтесь с содержанием разделов 1 и 2 данного пособия.

Задания

1 Ознакомиться с программной средой Multisim, изучить ее элементы управления и процедуру создания проектов с использованием микроконтроллеров.

Для этого необходимо запустить программу Multisim двойным щелчком по иконке на рабочем столе. После этого создать новый проект с применением микроконтроллеров и создать в нем простейшую схему, показанную на рисунке 8. Загрузить в контроллер вариант демонстрационной программы, указанный преподавателем, запустить ее в режиме прогона и пошаговом режиме, наблюдая результаты выполнения программы.

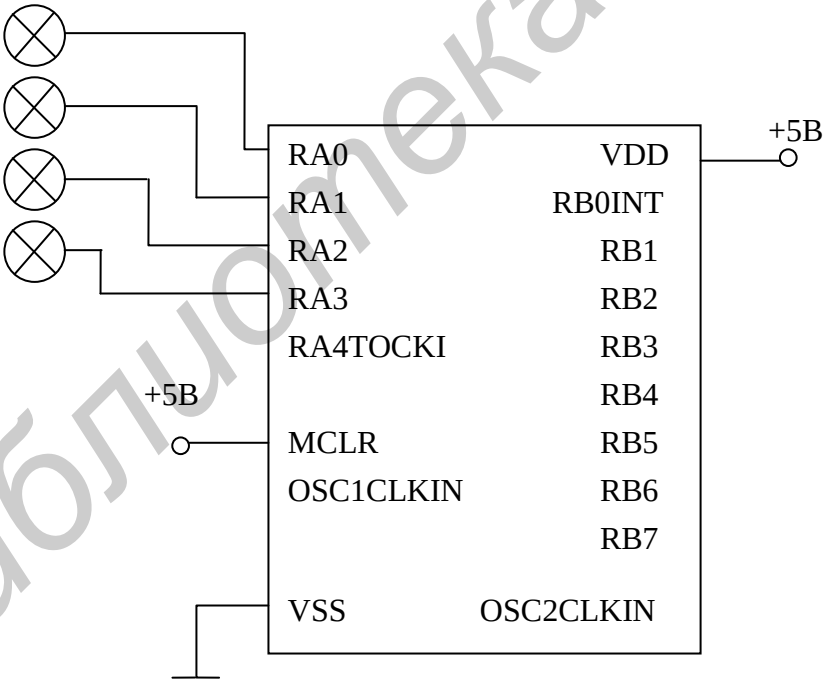


Рисунок 8 – Принципиальная схема тестового устройства

2 Изучить основные команды контроллера PIC16F84. Уяснить функции команд программы.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Блок-схема алгоритма демонстрационной программы.
4 Листинг программы с комментариями по функциям команд.
5 Вывод.

Контрольные вопросы

1 Назовите основные элементы окна MCU Window.
2 Какую операцию выполняет команда COMF?
3 Какую операцию выполняет команда SUBWF?
4 Какую операцию выполняет команда TRIS?
5 Как в пакете Multisim создать проект с использованием микроконтроллера?

ЛАБОРАТОРНАЯ РАБОТА №2

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ В КОДАХ PIC-КОНТРОЛЛЕРА

Цель работы – приобретение навыков программирования арифметических и логических операций в кодах PIC-контроллера.

Теоретическая часть

В состав команд арифметических операций микроконтроллера входят команды байтовых операций суммирования, вычитания, инкремента и декремента, а также команды операций с двухбайтными операциями: суммирования, инкремента и декремента.

К командам логических операций относятся команды операций конъюнкции, дизъюнкции, суммирования по модулю 2 и сдвигов. При выполнении логических операций устанавливаются флаги нуля, переноса и паритета.

С целью изучения арифметических и логических операций выделите эти операции из таблиц 6–7 и создайте проект с принципиальной схемой, представленной на рисунке 9.

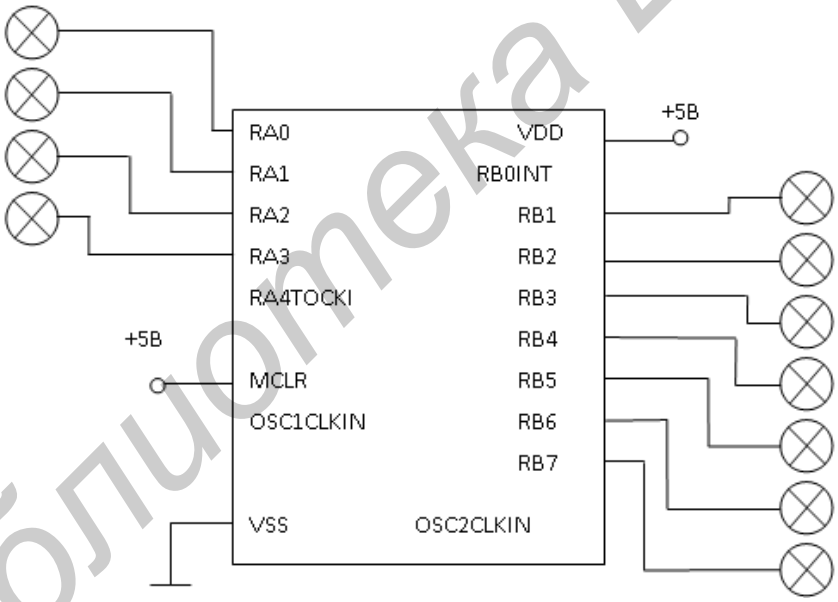


Рисунок 9 – Принципиальная схема тестового устройства

Для визуализации работы программы в схему включены индикаторные лампы, которые по желанию можно заменить, например, светодиодами, включенными последовательно с соответствующими балластными резисторами.

На линии порта A отображается работа арифметических операций, а на линии порта B – работа логических операций.

Листинг программы, позволяющей получать одновременно мигающие лампочки на портах A и B:

```
#include "p16f84.inc"  ;Включение описания PIC16F84 для Ассемблера MPASM
MY_REG              EQU     0x0C ; делаем все линии порта A выходными
```

```
        BSF             STATUS,RP0      ; переходим в первый банк
        MOVLW           0x00
        MOVWF           TRISA       ; делаем все линии порта B выходными
        MOVLW           0x00
        MOVWF           TRISB
        BCF             STATUS,RP0      ; обратно в нулевой банк
        MOVLW           B'00001'        ; зажигаем первую лампу
        MOVWF           PORTA           ; через рабочий регистр
        MOVLW           B'10101010'     ; зажигаем четные лампочки
        MOVWF           PORTB           ; через рабочий регистр
START   MOVLW   0FFh                    ; загружаем 0FFh в регистр MY_REG
        MOVWF   MY_REG                  ; через рабочий регистр
LOOP    DECFSZ  MY_REG,1    ; уменьшаем MY_REG на 1
        GOTO    LOOP        ; если флаг нуля не выставился (не досчитали
              ;до нуля), то переходим обратно, иначе пропускается команда goto
        COMF    PORTB       ; инвертируем
        INCF    PORTA,1     ; лампочки загораются, выдавая двоичную запись
                            ;десятичного числа
        GOTO    START       ; и снова на начало
        END
```

Задания

1 Создать проект со схемой, представленной на рисунке 9, и написать программу, позволяющую получить на портах A поочередное совместное мигание лампочек, начиная со второй, а на портах B – поочередное мигание лампочек, начиная с первой. Использовать для этого арифметические и логические операции соответственно.

2 Создать проект со схемой, представленной на рисунке 9, и написать программу, позволяющую получить на портах A поочередное совместное мигание лампочек, начиная с третьей, а на портах B – поочередное мигание лампочек, начиная со второй. Использовать для этого арифметические и логические операции соответственно.

3 Продемонстрировать результаты преподавателю.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Принципиальная схема тестового устройства.

4 Блок-схема алгоритма программы.

5 Листинг программы с комментариями.

6 Вывод.

Контрольные вопросы

1 Назовите функции линий порта A.

2 Назовите основные команды арифметических операций.

3 Назовите функции линий порта B.

4 Назовите основные команды логических операций.

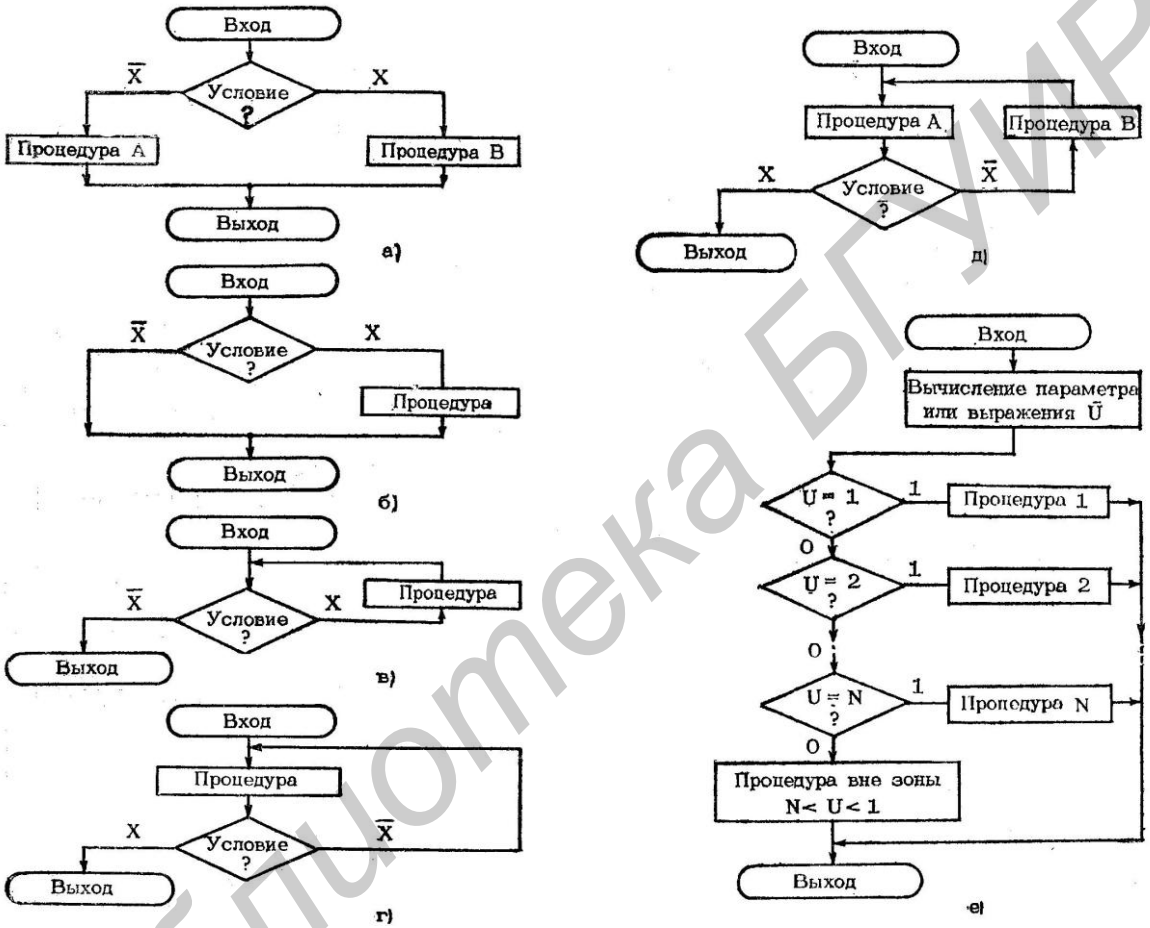
ЛАБОРАТОРНАЯ РАБОТА №3

КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Цель работы – научиться составлять простейшие ветвящиеся программы с использованием команд передачи управления.

Теоретическая часть

Типовые блок-схемы ветвящихся алгоритмов (БСА), широко встречающихся в программах обработки данных, приведены на рисунке 10.



а – структура ЕСЛИ-ТО-ИНАЧЕ; б – структура ЕСЛИ-ТО; в – структура ДЕЛАЙ-ПОКА; г – структура ПОВТОРЯЙ-ДО-ТОГО-КАК; д – структура ПРОЦЕСС-ПОКА; е – структура ДЕЛАЙ-В-ЗАВИСИМОСТИ-ОТ

Рисунок 10 – Типовые блок-схемы алгоритмических структур

Структура (а) применяется, когда нужно реализовать программный переход к одной из двух вычислительных процедур в зависимости от выполнения некоторого условия. Структура (б) используется, когда в зависимости от проверяемого условия нужно выполнить или не выполнить некоторую процедуру. Структуры (в) и (г) применяются для проверки окончания циклов, при этом процедура выполняется после (в) или до проверки условия (г). Структура (д) объединяет в себе две предыдущие структуры, а структура (е) осуществляет выбор действия при многозначных решениях и используется для замены цепо-

чек структур типа (а). Для организации подобных структур программ предназначены команды передачи управления.

В системе команд PIC-контроллера имеется относительно небольшой набор команд передачи управления (таблица 8). Команды проверки условий могут передать управление только при выполнении некоторого условия, заданного в коде операции (таблица 9).

Формирование и обработка массивов осуществляется циклическими программами, состоящими из четырех частей: инициализация, обращение к текущему элементу, переход к следующему элементу и проверка условия окончания цикла. В программах, как правило, используются два регистра: регистр, называемый указателем, где хранится адрес текущего элемента, и регистр-счетчик, фиксирующий окончание цикла после обработки последнего элемента массива, если известна его длина.

Таблица 8 – Команды передачи управления

Мнемокод	Название команды	Примечание
CALL k	Вызов подпрограммы	
CLRWDT	Сброс Watchdog таймера	
GOTO k	Переход по адресу	
RETLW k	Возврат из подпрограммы с загрузкой константы в W	
RETFIE	Возврат из прерывания	
RETURN	Возврат из подпрограммы	
SLEEP	Переход в режим SLEEP	

Для создания условных переходов в PIC-контроллерах можно использовать сочетания команд проверки условий (см. таблицу 9) и безусловного перехода.

Таблица 9 – Команды проверки условий

Мнемокод	Название команды	Примечание
BTFSC f,b	Пропустить команду, если бит равен нулю	
BTFSS f,b	Пропустить команду, если бит равен единице	
DECFSZ f,d	Декремент f, пропустить команду, если 0	
INCFSZ f,d	Инкремент f, пропустить команду, если 0	

С целью изучения команд передачи управления и организации ветвлений создайте проект устройства, принципиальная схема которого представлена на рисунке 11.

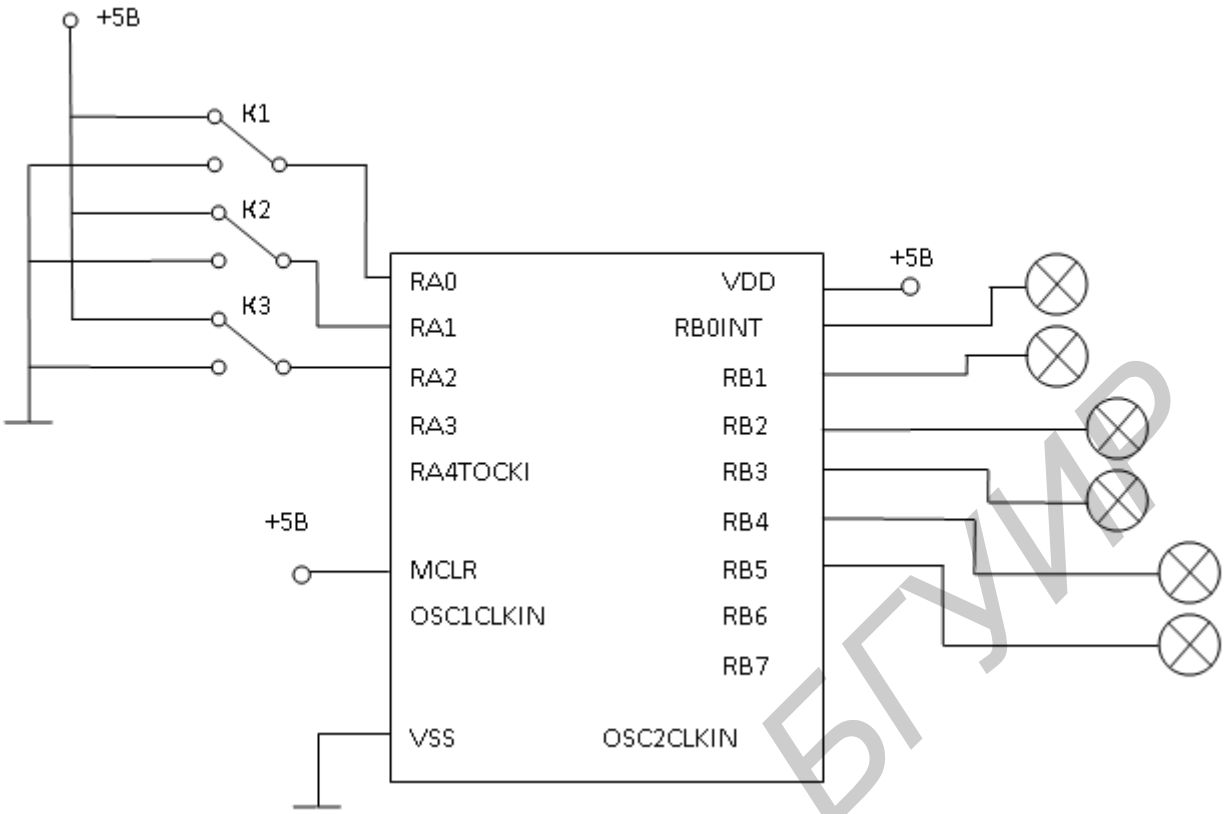


Рисунок 11 – Принципиальная схема тестового устройства

В схеме реализован несложный алгоритм приоритетного управления. Есть три ключа (кнопки K1, K2, K3). Каждый ключ управляет парой ламп: включённый ключ зажигает первую лампу пары, выключенный – вторую. Но нельзя использовать второй ключ, пока не включим первый (пока не зажжём вторую лампочку в первой паре). Аналогичная ситуация и со связью второго и третьего ключа. Такой алгоритм управления задается логикой программных переходов.

Листинг программы, отражающей алгоритм ветвления:

```
#include "p16f84.inc" ;Включение описания PIC16F84 для Ассемблера MPASM
MY_REG          EQU     0x0C; делаем все линии порта A входными
     BSF        STATUS,RP0              ; переходим в первый банк
     MOVLW              B'11111111'
     MOVWF              TRISA
     MOVLW              B'00000000'
     MOVWF              TRISB  ; делаем все линии порта B выходными
     BCF        STATUS,RP0      ; обратно в нулевой банк
START     MOVLW    0FFh          ; загружаем 0FFh в регистр MY_REG
          MOVWF    MY_REG        ; через рабочий регистр
LOOP      DECFSZ   MY_REG,1      ; уменьшаем MY_REG на 1
          GOTO     LOOP          ; если флаг нуля не выставился
          ;(не досчитали до нуля), то переходим обратно, иначе
          ;пропускается команда goto
     BTFSC    PORTA,0          ; если первый бит порта A не установлен,
```

```
        GOTO    LIGHT_0         ; то пропускаем LIGHT_0
        GOTO    BLOW_0          ; и запускаем BLOW_0, то есть выключаем
                                ;первую пару лампочек
CHECK_2
        BTFSC   PORTA,1 ; в случае если первый ключ разрешает
        GOTO    LIGHT_1  ;дальнейшую проверку
        GOTO    BLOW_1
CHECK_3                         ; в случае если второй ключ разрешает
        BTFSC   PORTA,2         ;заключительную проверку
        GOTO    LIGHT_2
        GOTO    BLOW_2
LIGHT_0                         ; зажигание второй лампы первой пары
        BCF     PORTB,0
        BSF     PORTB,1
        GOTOCHECK_2             ; и разрешение на дальнейшую проверку
BLOW_0                          ; с погашением первой пары
        BSF   PORTB,0
        BCF   PORTB,1
        GOTO        START       ; не даём возможности проверять другие ключи
LIGHT_1
        BCF     PORTB,2
        BSF     PORTB,3
        GOTOCHECK_3
BLOW_1
        BSF    PORTB,2
        BCF    PORTB,3
        GOTO        START
LIGHT_2
        BCF     PORTB,4
        BSF     PORTB,5
        GOTO        START
BLOW_2
        BSF     PORTB,4
        BCF     PORTB,5
        GOTO        START
        END
```

Задания

1 Создать проект со схемой, представленной на рисунке 11, и написать программу, позволяющую получить зажигание i-й лампы j-й пары, если ключ Km разомкнут, а Kn замкнут. Числа i, j, m, n задает преподаватель.

2 Создать проект со схемой, представленной на рисунке 11, и написать программу, реализующую заданную преподавателем логическую функцию.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Принципиальная схема тестового устройства.

4 Блок-схема алгоритма программы.

5 Листинг программы с комментариями.

6 Вывод.

Контрольные вопросы

1 Приведите примеры типовых ветвящихся алгоритмов.

2 Каким образом можно реализовать условные переходы в PIC-контроллере?

3 Каким образом в программе реализована задержка выдачи управляющих сигналов?

4 Какую операцию выполняет команда BTFSS?

ЛАБОРАТОРНАЯ РАБОТА №4

ПРОГРАММИРОВАНИЕ ВРЕМЕННЫХ ЗАДЕРЖЕК

Цель работы – изучение программных средств управления временем выполнения программы, приобретение навыков программирования циклических алгоритмов в кодах микроконтроллера.

Теоретическая часть

Одним из наиболее простых методов формирования временных задержек при работе в реальном масштабе времени является создание программного кода, который будет выполняться определенное, достаточно длительное время. Чаще всего этот метод реализуется с помощью циклических алгоритмов с выходом из цикла по обнулению счетчика, а время выполнения наращивается за счет создания нескольких вложенных циклов. С учетом возможностей стека МК PIC16F84 уровень вложенности циклов и подпрограмм не должен превышать 8. Программа, приведенная ниже, иллюстрирует создание задержки 1 с при тактовой частоте 4 МГц (100 000 циклов длительностью 1 мкс каждый) с помощью трех вложенных циклов. Для задержки в счетчики загружаются значения 0x08, 0x2F и 0x03. Точная подгонка времени достигается помещением в конце еще нескольких команд. Вычисление времени производится, если мы знаем время выполнения команд (один или два цикла) и тактовую частоту (определяет длительность цикла.

```
        CBLOCK
        D1
        D2
        D3
        ENDC
                    ;999997 циклов
        MOVLW       0x08
        MOVWF       D1
        MOVLW       0x2F
        MOVWF       D2
        MOVLW       0x03
        MOVWF       D3
DELAY_0
        DECFSZ      D1, F
        GOTO   $+2
        DECFSZ      D2, F
        GOTO$+2
        DECFSZ      D3, F
        GOTO DELAY_0
                    ;3 цикла
        GOTO   $+1
        NOP
```

Следующая программа иллюстрирует другой вариант реализации того же метода для получения временного интервала в 5 с:

```
PAUSE05
      MOVLW 0x7E
      MOVWF REG_1
      MOVLW 0x89
      MOVWF REG_2
      MOVLW 0x03
      MOVWF REG_3
WR    DECFSZ REG_1, F
      GOTO WR
      CLRWDT
      DECFSZ REG_2, F
      GOTO WR
      DECFSZ REG_3, F
      GOTO WR
   RETURN
```

Для выполнения этого фрагмента программы необходимо в ее начале не забыть определить адреса регистров REG_1-REG_3 директивами EQU.

При выполнении циклов задержки не должно возникать прерываний, поскольку на их обработку тратится время, и в результате время задержки возрастает по сравнению с расчетным.

Для изучения способов формирования программных задержек и создания циклических алгоритмов может быть использована принципиальная схема, представленная на рисунке 12.

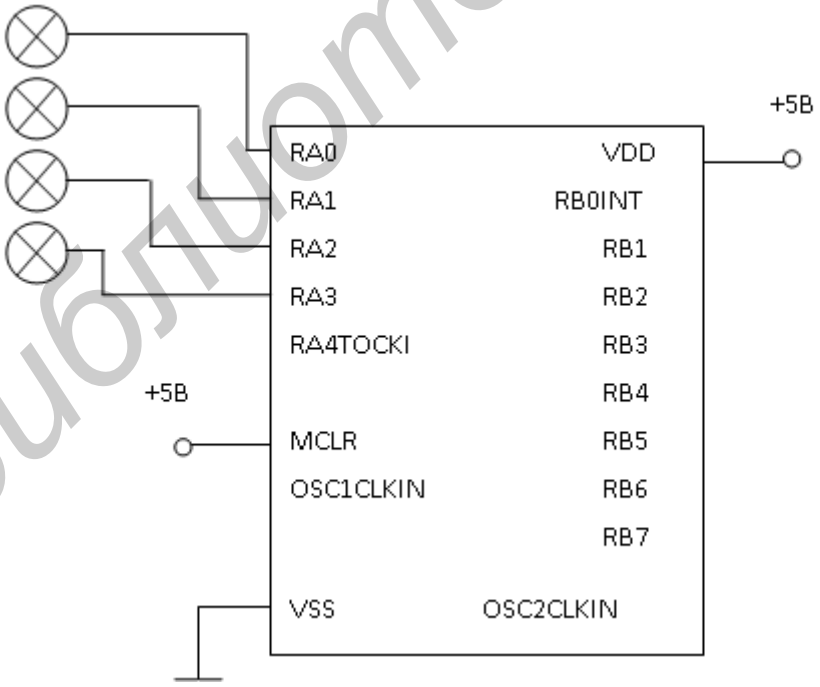


Рисунок 12 – Принципиальная схема тестового устройства

Листинг программы, позволяющей получать одновременно мигающие лампы, используя для задержки вложенный цикл:

```
#include "p16f84.inc" ;Включение описания PIC16F84 для Ассемблера MPASM
MY_REG          EQU     0Ch
        BSF             STATUS,RP0      ; переходим в первый банк
        MOVLW           0x00
        MOVWF           TRISA           ; делаем все линии порта A выходными
        BCF             STATUS,RP0      ; обратно в нулевой банк
START
        MOVLW           0               ; зажигаем все лампы
        MOVWF           PORTA           ; через рабочий регистр
        MOVLW   0FFh            ; загружаем 0FFh в регистр MY_REG
        MOVWF   MY_REG          ; через рабочий
LOOP1
        DECFSZ  MY_REG,1        ; уменьшаем MY_REG на 1
        GOTO    LOOP1           ; если флаг нуля не выставился (не досчитали
до нуля), то переходим обратно, иначе пропускается команда goto
        MOVLW           B'11111'        ; зажигаем
        MOVWF           PORTA           ; через рабочий регистр
        MOVLW   0FFh            ; загружаем 0FFh в регистр MY_REG
        MOVWF   MY_REG          ; через рабочий
LOOP2
        DECFSZ  MY_REG,1        ; уменьшаем MY_REG на 1
        GOTO    LOOP2           ; если флаг нуля не выставился (не досчитали
до нуля), то переходим обратно, иначе пропускается команда goto
        GOTO    START           ; и снова на начало
        END
```

Задания

1 Проанализировать тексты программ временных задержек и вывести общую формулу (для одной из программ), позволяющую определить значения констант, загружаемых в счетчики.

2 Создать схему, представленную на рисунке 12, и написать программу, позволяющую получить одновременно мигающие лампочки с задержкой N секунд. Количество и номера лампочек, а также число N в диапазоне 1–5 с задает преподаватель.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Формула для вычисления времени задержки, с пояснениями.

4 Принципиальная схема тестового устройства.

5 Блок-схема алгоритма.

6 Листинг программы.

7 Вывод.

Контрольные вопросы

1 Какую операцию выполняет команда MOVWF?

2 Какую операцию выполняет команда DECFSZ?

3 Какую операцию выполняет команда BSF?

4 Как получить задержку 0,7 с при частоте 10 МГц?

5 Какой минимальный уровень вложения циклов необходим для получения временной задержки 0,1 с при тактовой частоте 4 МГц?

ЛАБОРАТОРНАЯ РАБОТА №5

ПРОГРАММИРОВАНИЕ ВРЕМЕННЫХ ЗАДЕРЖЕК С ПОМОЩЬЮ ТАЙМЕРА

Цель работы – изучение программных средств управления временем выполнения программы с помощью таймера.

Теоретическая часть

Одним из удобных способов формирования временных задержек является использование встроенного таймера. Достоинство этого способа – одновременно с формированием задержки МК может выполнять и другие операции, при этом их выполнение может не влиять на длительность задержки. Обычно в таких случаях программа задержки включает формирование какого-либо интервала с помощью таймера, а также программный счетчик, отсчитывающий определенное количество таких интервалов. Для получения некоторого интервала в таймер можно загрузить число, с которого и начнется счет до переполнения. В нижеприведенном примере таймер перезапускается после переполнения по истечении времени 872 мкс:

```
RESTART_TMR0
        BCF   INTCON, T0IF      ; Очистить флаг переполнения
        MOVLW     D'37'         ; 255 - 37 = 218 x 4 = 872 uSec
        MOVWF     TMR0          ;Запуск таймера
        RETURN
```

Далее можно организовать программный счетчик для подсчета числа переполнений таймера. Идентификацию переполнения можно осуществлять либо с помощью программного опроса бита переполнения таймера, либо с помощью процедуры прерывания от таймера. С целью изучения способов формирования задержек с применением таймера может быть использована принципиальная схема, представленная на рисунке 12.

Задания

1 Написать программу временной задержки, включающей N переполнений таймера (число N задает преподаватель).

2 Создать проект, включив в него схему, представленную на рисунке 12, и написать программу, позволяющую получить одновременно мигающие лампы с задержкой N секунд, применив программу задержки по переполнению таймера. Количество и номера ламп, а также число N в диапазоне 0,3–5 с задает преподаватель.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Формула для вычисления времени задержки, с пояснениями.

4 Принципиальная схема тестового устройства.

5 Листинг программы с комментариями.

6 Вывод.

Контрольные вопросы

1 Как определить момент переполнения таймера?

2 Как можно запустить и остановить таймер?

3 Как получить задержку 0,01 с при частоте 10 МГц?

4 Каково максимальное время переполнения таймера при тактовой частоте 4 МГц?

ЛАБОРАТОРНАЯ РАБОТА №6

БИТОВЫЕ ОПЕРАЦИИ В PIC-КОНТРОЛЛЕРЕ

Цель работы – изучение программных средств обработки бит и особенностей программирования битовых операций в кодах PIC-контроллера.

Теоретическая часть

Набор команд, используемых при работе с битами, представлен в таблице 10.

Таблица 10 – Бит-ориентированные команды

Мнемокод	Название команды
BCF	Сброс бита в регистре f
BSF	Установка бита в регистре f
BTFSC	Пропустить команду, если бит равен 0
BTFSS	Пропустить команду, если бит равен 1

С целью изучения битового пространства памяти PIC-контроллера разработана принципиальная схема, представленная на рисунке 13.

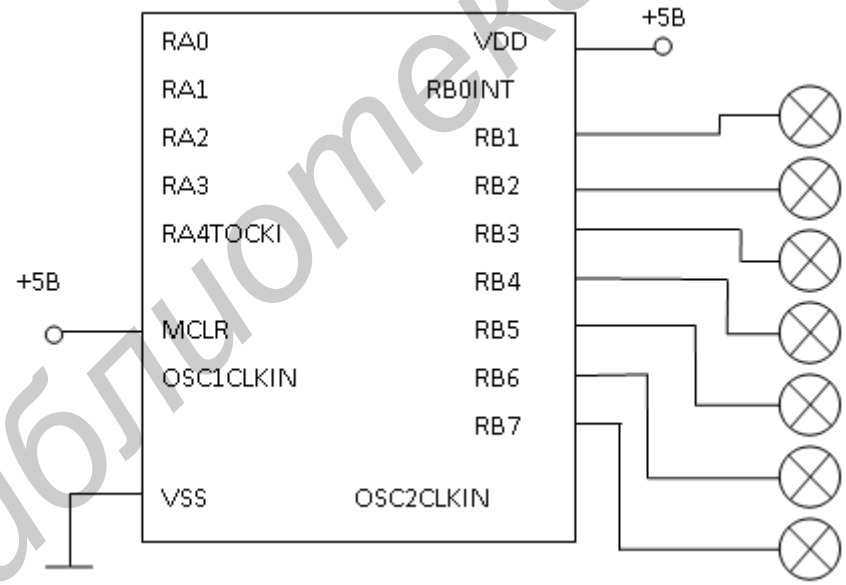


Рисунок 13 – Принципиальная схема тестового устройства

Листинг программы, позволяющей получать последовательное мигание лампочек:

```
; работа с битами заключается в следующем:
; – выставляем/сбрасываем в регистре STATUS бит RP0
; – очищаем бит переноса C в регистре STATUS
; – сдвиг бит регистра PORTB
```

```
; – команда BTFSS, проверяющая состояние бита регистра
#include "p16f84.inc" ;Включение описания PIC16F84 для Ассемблера MPASM
; Команды BCF STATUS,RP и BSF STATUS,RP нужны для переключения
; между банками памяти.
; Не забываем, что память данных микроконтроллера разбита на два банка.
; Банку 0 соответствуют адреса 0x00..0x7F, банку 1 адреса 0x8F..0xFF.
; Выбор банка определяется состоянием бита 5 в регистре STATUS.
; Когда этот бит установлен в 1, выбран банк 1, иначе – банк 0.
      MSB    EQU    7          ; номер старшего бита
      BSF    STATUS,RP0        ; выбрать регистровый банк 1
      MOVLW      0x00
      MOVWF      TRISB         ; линии порта B на вывод
      BCF    STATUS,RP0        ; выбрать регистровый банк 0
      MOVLW      0x01          ; через рабочий регистр
      MOVWF      PORTB         ; зажигаем первую лампку (порт B)
      BCF    STATUS,C          ; очистить флаг C для того, чтобы при
                               ; сдвиге его значение на записалось в
                               ; начало регистра PORTB
LEFT       RLF    PORTB,F     ; сдвинуть содержимое индикатора влево
     BTFSS  PORTB,MSB         ; достигли крайней левой позиции?
     GOTO   LEFT              ; если нет – цикл
RIGHT      RRF    PORTB,F     ; сдвинуть содержимое индикатора вправо
     BTFSS  PORTB,0           ; достигли крайней левой позиции?
     GOTO   RIGHT             ; если нет – цикл
     GOTO   LEFT              ;начать новый цикл
     END
```

Задания (по вариантам)

Создать проект, включив в него схему, представленную на рисунке 13, и:

1) написать программу, позволяющую получить «бегущий огонёк»;

2) написать программу, позволяющую получить «бегущий огонёк», начинающийся с i-й лампочки (число i указывает преподаватель);

3) написать программу, выдающую последовательность зажигания ламп, а также периоды свечения и паузы между ними, указанные преподавателем.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Принципиальная схема тестового устройства.

4 Листинг программы.

5 Вывод.

Контрольные вопросы

1 Какую операцию выполняет команда BTFSS?

2 Для чего нужны команды BCF STATUS,RP и BSF STATUS,RP?

3 Объяснить работу цикла в программе.

4 Каким образом можно изменить очередность начала мигания ламп?

ЛАБОРАТОРНАЯ РАБОТА №7

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ФУНКЦИЙ ДИНАМИЧЕСКОГО ВЫВОДА ЗНАКОВОЙ И СИМВОЛЬНОЙ ИНФОРМАЦИИ

Цель работы – ознакомиться с некоторыми особенностями организации вывода знаковой и символьной информации и программной реализации функций динамического вывода такой информации.

Теоретическая часть

В лабораторной работе используется семисегментный индикатор, позволяющий отображать информацию, поступающую с портов PIC-контроллера (рисунок 14).

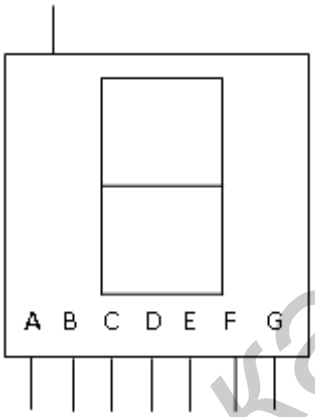


Рисунок 14 – Вид семисегментного индикатора

С целью изучения программной реализации функций динамического вывода знаковой и символьной информации необходимо создать проект с принципиальной схемой, представленной на рисунке 15.

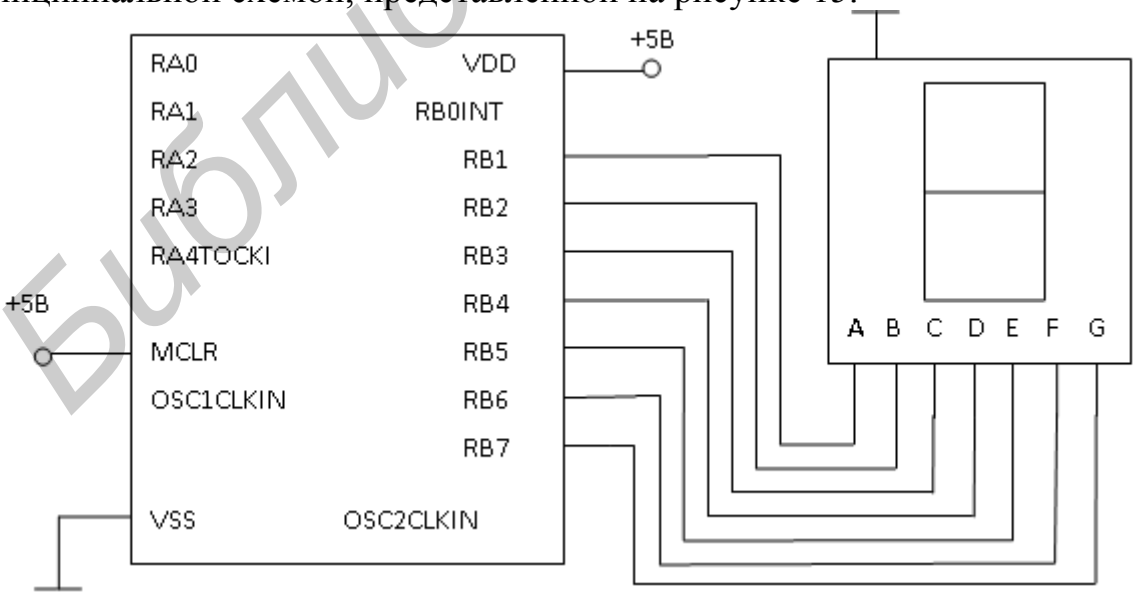


Рисунок 15 – Принципиальная схема тестового устройства

Листинг программы, позволяющей получать последовательное отображение цифр и букв на индикаторе:

```
#include "p16f84.inc" ;Включение описания PIC16F84 для Ассемблера MPASM
DELAYCOUNT1   EQU         0x1A  ; используется для задержки
DELAYCOUNT2   EQU         0x1B  ; используется также для задержки
Counter              EQU         0x1C  ; счётчик (тоже в задержке)
; инициализация портов (стандартная)
; порты B для выхода
      BSF         STATUS,RP0              ; банк 1
      MOVLW       0x00
      MOVWF       TRISB
      BCF         STATUS,RP0              ; банк 0
; выключаем дисплей (инвертированная логика) и счётчик обнуляем
      MOVLW       0xFF
      MOVWF       PORTB
      MOVLW       0x00
      MOVWF       Counter
; основной цикл
Loop
      CALL  DELAY               ; основная задержка (вызов функции)

      INCF  Counter,1         ; увеличиваем регистр счётчика на 1
      BTFSS       Counter,4             ; проверяем на переполнение (больше F)
      GOTO        Display               ; если меньше либо равно F
      MOVLW       0x00                  ; если больше, то обнуляем счётчик
      MOVWF       Counter
Display                                 ;отображение значения счётчика на
                                        ;семисегментный индикатор
      CALL  Segmente            ;вызов функции, которая запишет в
                                        ;рабочий регистр текущее значение счётчика
      MOVWF       PORTB         ;и потом его передаем на выход порта B
      GOTO        Loop          ;снова в начало цикла
; таблица под 7-сегментный индикатор с инверсной логикой
Segmente
      MOVF  Counter,0           ; значение счётчика в рабочий регистр
      ADDWF       PCL, f        ; прибавляет к младшему байту программного
                                  ; счётчика (PCL) значение рабочего регистра,
                                  ; что фактически означает переход на ту
                                  ; команду, которая соответствует значению
                                  ; счётчика. Команда возвращает управление
                                  ; из подпрограммы, устанавливая необходимое
                                  ; для отображения значения в рабочий регистр
      RETLW       B'11000000' ; 0 0xC0
      RETLW       B'11111001' ; 1 0xF9
      RETLW       B'10100100' ; 2 0xA4
      RETLW       B'10110000' ; 3 0xB0
      RETLW       B'10011001' ; 4 0x99
      RETLW       B'10010010' ; 5 0x92
      RETLW       B'10000010' ; 6 0x82
```

```
        RETLW   B'11111000' ; 7 0xF8
        RETLW   B'10000000' ; 8 0x80
        RETLW   B'10010000' ; 9 0x90
        RETLW   B'10001000' ; A 0x88
        RETLW   B'10000011' ; B 0x83
        RETLW   B'11000110' ; C 0xC6
        RETLW   B'10100001' ; D 0xA1
        RETLW   B'10000110' ; E 0x86
        RETLW   B'10001110' ; F 0x8E
; задержка  необходима достаточная для записи в EEPROM:
DELAY
        MOVLW   0xF7                ; установка/сброс счётчика
        MOVWF   DELAYCOUNT1
        MOVLW   0xAA        ; DELAYCOUNT2 устанавливаем в 0
DELAYLOOP1
        MOVWF   DELAYCOUNT2
DELAYLOOP2
        INCFSZ  DELAYCOUNT2,1
        GOTO    DELAYLOOP2
        INCFSZ  DELAYCOUNT1,1
        GOTO    DELAYLOOP1
        RETURN
        END
```

Задание

Создать проект, содержащий схему, представленную на рисунке 15, и написать программу, позволяющую получить на индикаторе последовательное отображение цифр и букв, указанных преподавателем.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Принципиальная схема тестового устройства.

4 Блок-схема алгоритма.

5 Листинг программы.

6 Вывод.

Контрольные вопросы

1 Объясните работу семисегментного индикатора.

2 Какую функцию выполняет команда Display?

3 Для чего используется команда DELAYCOUNT1 EQU 0x1A?

4 Для чего используется команда INCF?

ЛАБОРАТОРНАЯ РАБОТА №8

КОМАНДЫ ВВОДА/ВЫВОДА И ОБРАЩЕНИЯ К ПОДПРОГРАММАМ. РАБОТА СО СТЕКОМ

Цель работы – изучение команд ввода/вывода, обращения к подпрограммам, работы со стеком.

Теоретическая часть

Обращение процессора к внешним устройствам может осуществляться так же, как обращение к ячейкам памяти с использованием прямой или косвенной адресации.

1 Прямая адресация в кодах PIC-контроллера

Когда производится прямая 9-битная адресация, младшие 7 бит берутся как прямой адрес из кода операции, а два бита указателя страниц (RP1,RP0) – из регистра статуса (03h) (рисунок 16).

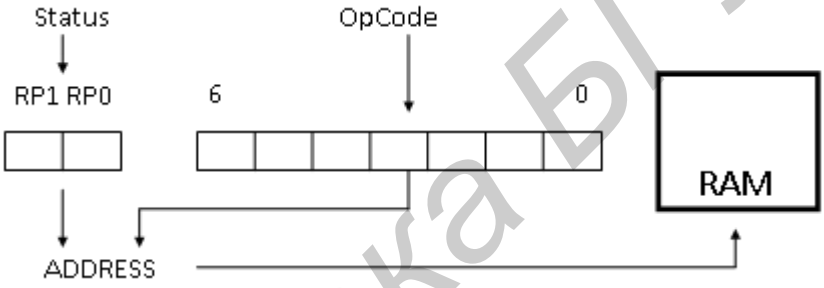


Рисунок 16 – Прямая адресация в кодах PIC-контроллера

2 Косвенная адресация в кодах PIC-контроллера

Любая команда, которая использует f0 (адрес 00) в качестве регистра, фактически обращается к указателю, который хранится в FSR (04h). Чтение косвенным образом самого регистра f0 даст результат 00h. Запись в регистр f0 косвенным образом будет выглядеть как Nop, но биты статуса могут быть изменены. Необходимый 9-битный адрес формируется объединением содержимого 8-битного FSR регистра и бита IRP из регистра статуса (рисунок 17).

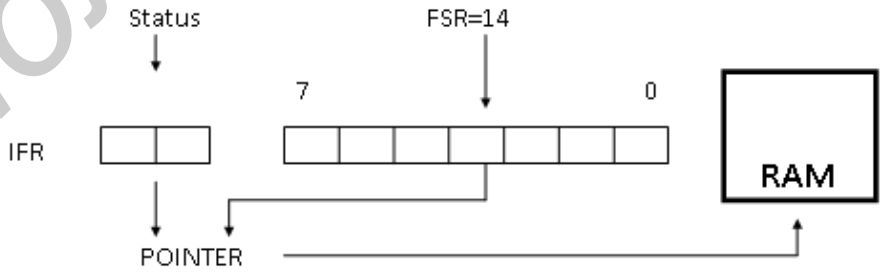


Рисунок 17 – Косвенная адресация в кодах PIC-контроллера

3 Стек и возвраты из подпрограмм

Контроллер PIC16F84 имеет восьмиуровневый аппаратный стек шириной 13 бит. Область стека не принадлежит ни к программной области, ни к области данных, а указатель стека пользователю недоступен. Текущее значение

программного счетчика посылается в стек, когда выполняется команда CALL или производится обработка прерывания. При выполнении процедуры возврата из подпрограммы (команды RETLW , RETFIE или RETURN) в программный счетчик выгружается содержимое стека. Отсюда ясен максимально допустимый уровень вложенности подпрограмм. Регистр PCLATH (0Ah) не изменяется при операциях со стеком.

4 Команды безусловного и условного вызова подпрограмм

В состав команд микропроцессора входят команды безусловного и условного вызова подпрограмм, представленные в таблице 11.

Таблица 11 – Команды вызова подпрограмм

Мнемокод	Название команды	Флаги
CALL	Вызов подпрограммы	
CLRWDT	Сброс Watchdog таймера	TO,PD
GOTO	Переход по адресу	
RETLW	Возврат из подпрограммы с загрузкой константы в W	
RETFIE	Возврат из прерывания	
RETURN	Возврат из подпрограммы	
SLEEP	Переход в режим SLEEP	TO,PD

Область памяти, отводимая для размещения подпрограмм, не должна пересекаться с областями памяти данных и областью основной программы.

С целью изучения безусловного и условного вызова подпрограмм можно создать проект с принципиальной схемой, представленной на рисунке 18.

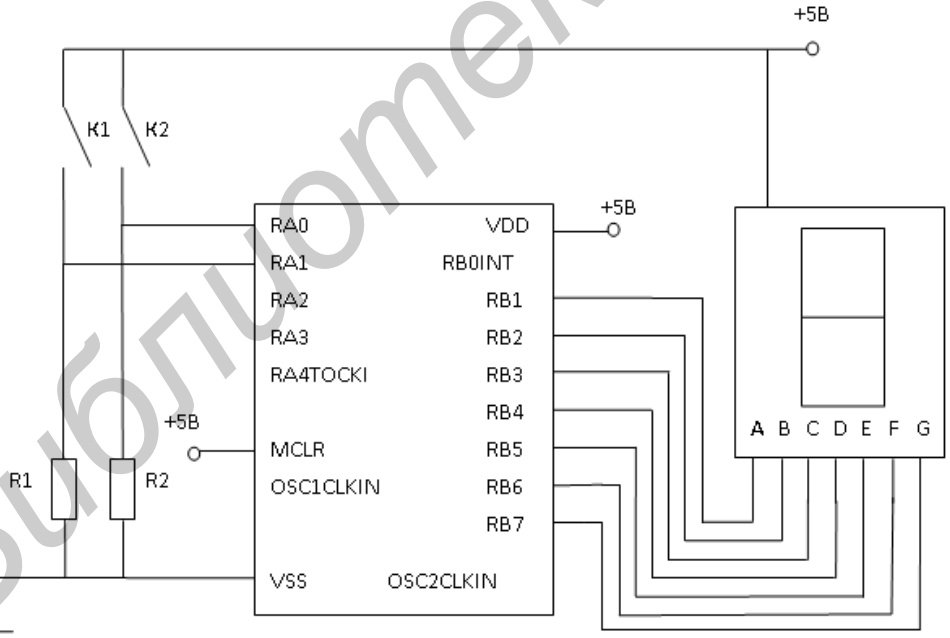


Рисунок 18 – Принципиальная схема тестового устройства

В данной схеме вывод осуществляется на матричный 7-сегментный индикатор. Для ввода используются две кнопки, регулирующие работу счётчика. Одна из них включает/выключает счёт, а другая определяет направление. Стек используется при вызове подпрограмм для сохранения значения программного счётчика (неявное использование).

Листинг программы, отражающей работу команд вызова подпрограмм:

```
#include "p16f84.inc" ;Включение описания PIC16F84 для Ассемблера MPASM
DELAYCOUNT1         EQU  0x1A  ;
DELAYCOUNT2         EQU  0x1B  ;
Counter             EQU        0x1C
; инициализация портов (стандартная)
; порты B – на вывод и порты A – на ввод (от кнопок)
      BSF       STATUS,RP0          ; банк 1
      MOVLW     0x00
      MOVWF     TRISB
      MOVLW     0x1F
      MOVWF     TRISA
      BCF       STATUS,RP0          ; банк 0
; выключаем индикатор (инверсная логика) и обнуляем счётчик
      MOVLW     0xFF
      MOVWF     PORTB
      MOVLW     0x00
      MOVWF     Counter
; основной цикл
Loop
      CALL  DELAY               ; основная задержка (вызов функции)
      BTFSC   PORTA, 0          ; читаем значение RA0
   GOTO    Display_OFF     ; если кнопка не нажата, то выключаем индикатор
   GOTO    Counter_Start       ; если же нажата, то начинаем счёт
Display_OFF                     ; выключаем индикатор
      MOVLW     0xFF
      MOVWF     PORTB
      MOVLW     0x00
      MOVWF     Counter   ; сбрасываем счётчик
      GOTO      Loop      ; и снова на задержку и определение
                          ; нажатия кнопки пуска
Counter_Start               ; начинаем счёт
      BTFSC   PORTA, 1      ; считываем RA1 (кнопка направления)
      GOTOCountDown         ; если она не нажата
      GOTO      CountUp     ; если нажата

CountUp                     ; счёт на увеличение
      INCF  Counter,1       ; увеличиваем регистр счётчика на 1
      BTFSS     Counter,4   ; проверяем на переполнение (больше F)
      GOTODisplay           ; если меньше либо равно F
      GOTOReset_INC         ; если больше, то обнуляем счётчик
CountDown                   ; счёт на уменьшение
      DECF      Counter,1   ; уменьшаем регистр счётчика на 1
      BTFSS     Counter,5   ; проверяем на переполнение (меньше 0)
      GOTODisplay           ; если счётчик больше либо равен 0
      GOTOReset_DEC         ; если меньше 0
Reset_INC       ; сброс счётчика в 0 после переполнения при увеличении
      MOVLW     0x00
      MOVWF     Counter
      GOTO      Display
Reset_DEC       ; сброс счётчика в F после переполнения при уменьшении
```

```
        MOVLW   0x0F
        MOVWF   Counter
    Display     ; отображение значения счётчика на 7-сегментный индикатор
        CALL Segmente       ; вызов функции, которая запишет в рабочий регистр текущее значение счётчика
        MOVWF   PORTB       ; и потом его отправляем на выход порта B
        GOTO    Loop            ; и снова на цикл
    ; таблица под семисегментный дисплей с инверсной логикой
    Segmente
        MOVFCounter,0   ; значение счётчика в рабочий регистр
        ADDWF   PCL, f
        RETLW   B'11000000' ; 0 0xC0
        RETLW   B'11111001' ; 1 0xF9
        RETLW   B'10100100' ; 2 0xA4
        RETLW   B'10110000' ; 3 0xB0
        RETLW   B'10011001' ; 4 0x99
        RETLW   B'10010010' ; 5 0x92
        RETLW   B'10000010' ; 6 0x82
        RETLW   B'11111000' ; 7 0xF8
        RETLW   B'10000000' ; 8 0x80
        RETLW   B'10010000' ; 9 0x90
        RETLW   B'10001000' ; A 0x88
        RETLW   B'10000011' ; B 0x83
        RETLW   B'11000110' ; C 0xC6
        RETLW   B'10100001' ; D 0xA1
        RETLW   B'10000110' ; E 0x86
        RETLW   B'10001110' ; F 0x8E
    DELAY
        MOVLW   0xF7
        MOVWF   DELAYCOUNT1
        MOVLW   0xAA        ;
    DELAYLOOP1
        MOVWF   DELAYCOUNT2
    DELAYLOOP2
        INCFSZ  DELAYCOUNT2,1
        GOTO    DELAYLOOP2
        INCFSZ  DELAYCOUNT1,1
        GOTO    DELAYLOOP1
        RETURN
        END
```

Задание

Создать модель устройства по схеме рисунка 18 и написать программу, позволяющую получить на индикаторе прямое и обратное отображение цифр и букв: в последовательности, темпе и составе, указанном преподавателем.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Принципиальная схема тестового устройства.

4 Блок-схема алгоритма программы.

5 Листинг программы с комментариями.

6 Вывод.

Контрольные вопросы

1 Какую операцию выполняет команда BCF?

2 Какую операцию выполняет команда CALL?

3 Какую операцию выполняет команда RETFIE?

4 Какую операцию выполняет команда RETURN?

ЛИТЕРАТУРА

1 Гурский, А. Л. Цифровые и микропроцессорные устройства средств измерений. Лабораторный практикум : учеб.-метод. пособие. В 2 ч. Ч.1 : Цифровые устройства / А. Л. Гурский, Н. А. Певнева. – Минск : БГУИР, 2010 – 52 с.

2 Катцен, С. PIC-микроконтроллеры. Полное руководство / С. Катцен ; пер. с англ.; под ред. А. В. Евстифеева. – М. : Додека-XXI, 2010. – 656 с.

3 Тавернье, К. PIC-микроконтроллеры. Практика применения / К. Тавернье. – М. : ДМК-Пресс, 2004. – 272 с.

4 Шестеркин, А. Н. Система моделирования и исследования радиоэлектронных устройств Multisim 10 / А. Н. Шестеркин. – М. : ДМК-Пресс, 2012. – 316 с.

Св. план 2012, поз. 54

Учебное издание

Гурский Александр Леонидович
Певнева Наталья Алексеевна

ЦИФРОВЫЕ И МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА СРЕДСТВ ИЗМЕРЕНИЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 2-х частях

Часть 2

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *И. П. Острикова, Е. С. Чайковская*
Корректор *Е. Н. Батурчик*
Компьютерная верстка *А. А. Лысеня*

Подписано в печать 04.01.2013. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 2,91. Уч.-изд. л. 3,0. Тираж 100 экз. Заказ 262.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛИ №02330/0494175 от 03.04.2009.
220013, Минск, П.Бровки, 6