МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра вычислительной техники и электроники

ИЗУЧЕНИЕ 8-РАЗРЯДНЫХ МИКРОКОНТРОЛЛЕРОВ MICROCHIP НА ПРИМЕРЕ PIC16F84A

Методические указания по выполнению лабораторных работ

Барнаул 2017

Составитель:

ст. преподаватель ***В.В. Белозерских***

Рецензент:

канд. физ-мат. наук, доцент ***В.В. Пашнев***

Представлена лабораторная работа по курсу «Микропроцессорные системы» для студентов направления «Информатика и вычислительная техника». Методические указания могут также быть использованы для студентов направления «Радиофизика».

Данный цикл создан на основании требований рабочей программы дисциплины «Микропроцессорные системы».

Тема: Изучение 8-разрядных микроконтроллеров Microchip на примере PIC16F84A.

Цель:

Изучение архитектуры и организации микроконтроллеров Microchip на примере PIC16F84A. Получение навыков программирования и разработки устройств с использованием пакета MPLAB.

Задачи:

Изучить особенности архитектуры и организации микроконтроллера PIC16F84A. В соответствии с заданием написать и отладить программы с помощью пакета MPLAB. Произвести программирование контроллера. Работоспособность созданного устройства проверить на лабораторном стенде. Предоставить отчет.

1.Теоретические сведения

1.1. *Микроконтроллер PIC16F84A*

На сегодняшний день микроконтроллеры используются во многих системах как центральное устройство управления. Основная область применения 8-разрядных контроллеров – встраиваемые устройства интеллектуального управления промышленной автоматики и бытовой аппаратуры. Специфика алгоритмов управления этих устройств не требует выполнения расчетов высокой точности в жестких условиях реального времени. Основная доля операций управления состоит в преобразовании логической информации. Следовательно, 8-разрядные МК могут с успехом реализовать эти задачи.

Одним из широко применяемых для этих задач является микроконтроллер фирмы Microchip PIC16F84A, имеющий Гарвардскую архитектуру и сокращенный набор команд (33), характерный для RISC – процессоров в отличие от CISC – процессоров с расширенным набором команд. PIC16F84A

имеет уникальную возможность электрического перепрограммирования памяти программ. Это позволяет легко вносить необходимые коррекции в программу на любом этапе проектирования и производства изделия. Кроме того, он имеет возможность внутрисхемного программирования. Система команд очень проста и может быть легко изучена.

PIC16F84A относится к семейству КМОП микроконтроллеров. Отличается тем, что имеет внутреннее 1Кх14 бит EEPROM для программ, 8-битовые данные и 64байт EEPROM памяти данных. При этом отличаются относительно низкой стоимостью и относительно высокой производительностью.

Высокая производительность микроконтроллеров PIC16F84A обусловлена большим числом архитектурных особенностей, характерных для RISC-микроконтроллеров. Архитектура основана на концепции раздельных шин и областей памяти для данных и для команд (Гарвардская архитектура) (рис. 1.1).

Шина данных и память данных (ОЗУ) - имеют ширину 8 бит, а программная шина и программная память (ПЗУ) имеют ширину 14 бит.

Такая концепция обеспечивает простую, но мощную систему команд, разработанную так, что битовые, байтовые и регистровые операции работают с высокой скоростью и с перекрытием по времени выборок команд и циклов выполнения. 14- битовая ширина программной памяти обеспечивает выборку 14-битовой команды в один цикл. Двухступенчатый конвейер обеспечивает одновременную выборку и исполнение команды. Все команды выполняются за один цикл, исключая команды переходов. В PIC16F84A программная память объемом 1Кх14 расположена внутри кристалла. Исполняемая программа может находиться только во встроенном ПЗУ.

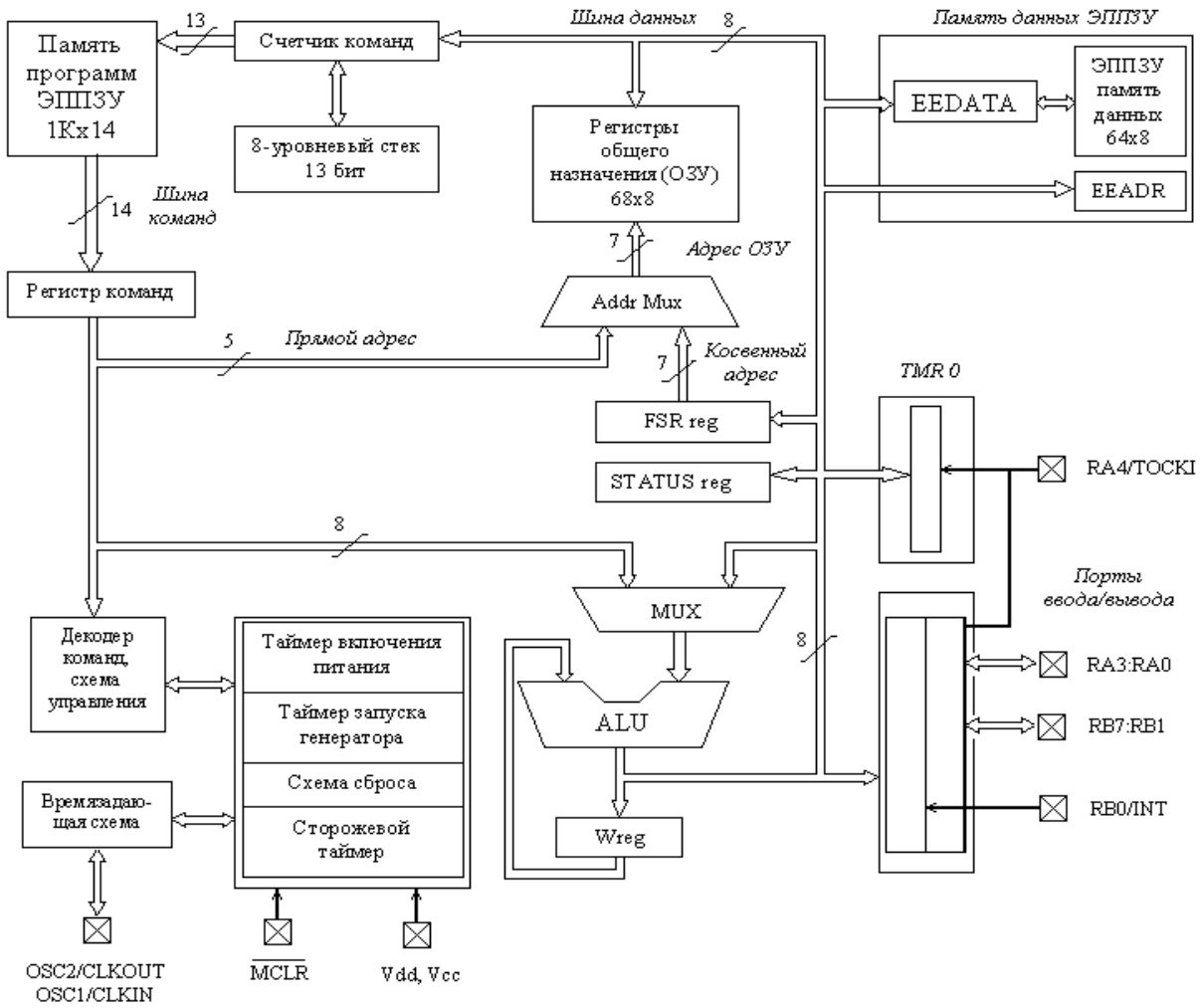


Рис. 1.1 Архитектура PIC16F84A.

Микроконтроллеры PIC используют прямую и косвенную адресацию всех регистров и ячеек памяти. Все специальные регистры и счетчик команд также отображаются на память данных. Ортогональная (симметричная) система команд позволяет выполнять любую операцию с любым регистром, используя любой метод адресации.

Регистры разделяются на две функциональные группы: специальные регистры и регистры общего назначения.

Специальные регистры включают в себя регистр таймера/счетчика (TMR0), счетчик команд (PC), регистр состояния (STATUS), регистры ввода/вывода (PORT), регистр косвенной адресации (FSR), регистры управления встроенным электрически перепрограммируемым ПЗУ (EEADR, EEDATA, EECON1, EECON2). Кроме того, специальные регистры управляют конфигурацией портов ввода/вывода и режимом последовательного делителя.

Регистры общего назначения используются программой для хранения переменных по усмотрению пользователя.

В микроконтроллерах PIC16F84A имеется 8-ми разрядное арифметико-логическое устройство (АЛУ) и рабочий регистр W. АЛУ выполняет сложение, вычитание, сдвиг, битовые и логические операции. В командах, имеющих два операнда, одним из операндов является рабочий регистр W. Вторым операндом может быть константа или содержимое любого регистра ОЗУ. В командах с одним операндом, операнд может быть содержимым рабочего регистра или любого регистра ОЗУ. Для выполнения всех операций АЛУ используется рабочий регистр W, который не может быть прямо адресован. В зависимости от результата выполнения операции изменяются значения битов переноса (C), десятичного переноса (DC) и нуля (Z) в регистре состояния STATUS. При вычитании биты C и DC рассматриваются, как биты заема и десятичного заема, соответственно.

Все команды микроконтроллера имеют, как было указано выше, одинаковую длину. Время выполнения команды – один машинный цикл. Каждый машинный цикл занимает 4 периода тактовой частоты. За это время последовательно выполняется дешифрация кода операции, его чтение, выполнение команды и запись результата. Такая последовательность действий позволяет организовать конвейерную обработку. Отдельно взятая команда выполняется за два машинных цикла. Но с учетом конвейерной обработки реальное время выполнения большинства команд (исключая команды, изменяющие состояние программного счетчика) составляет один машинный цикл. При тактовой частоте F=10 МГц время $t= 1/F*4 = 1/20*10^{-6}*4 = 400$нс.

Стек реализован аппаратно и его глубина равна 8. Стек не является частью памяти программ или данных и его указатель не может быть считан или записан. Биты, указывающие на переполнение и пустоту стека, отсутствуют.

Микроконтроллер PIC16F84A имеет два порта ввода/вывода – PORTA и PORTB. Некоторые разряды портов имеют дополнительные функции. Программа может считывать и записывать данные в регистры ввода/вывода ана-

логично регистрам общего назначения. При чтении всегда считывается действительное состояние выводов, независимо от того, запрограммированы они как входы или выходы. Конфигурация портов осуществляется с помощью регистров TRISA и TRISB. Микроконтроллер имеет встроенный 8-ми разрядный модуль таймера/счетчика. Вход счетчика совмещен с 4 выводом PORTA. В этом случае TMR0 увеличивается по каждому перепаду на входе TOCKI.

Микроконтроллер имеет 4 источника прерываний: внешнее прерывание от вывода RB0/INT; прерывание при переполнении таймера/счетчика TMR0; прерывание по окончании записи данных в EEPROM; прерывание при изменении сигналов на выводах порта RB<7:4>. Все прерывания имеют один и тот же вектор/адрес 0004h. Каждое прерывание может быть разрешено/запрещено индивидуально, кроме того, в регистре INTCON имеется бит общего разрешения/запрещения прерываний.

В микроконтроллерах PIC16F84A предусмотрены четыре типа генераторов: LP-микромощный резонатор; XT – керамический или кварцевый резонатор; HS – высокочастотный кварцевый резонатор; RC – RC цепочка. Кристаллы PIC16... могут также тактироваться и от внешних источников. Генератор, построенный на кварцевых или керамических резонаторах, требует периода стабилизации после включения питания. Кроме того, микроконтроллер имеет сторожевой таймер, работающий от внутреннего генератора (при написании программ задержки необходимо позаботиться о своевременном сбросе сторожевого таймера).

Выбор типа генератора, активация сторожевого таймера, таймера задержки при включении питания и установка бита защиты программы производится с помощью конфигурационного слова. Назначение битов:

FOSC0/FOSC - Биты выбора типа генератора.

FOSC1, FOSC0:

00 LP генератор

01 XT генератор

10 HS генератор

11 RC генератор

WDTE - Бит разрешения работы WDT.

WDTE = 1: WDT разрешен

WDTE = 0: WDT запрещен

PWRTE - Бит разрешения выдержки времени после детектирования включения питания

PWRTE = 1: Выдержка будет производиться

PWRTE = 0: Выдержки не будет

CP - Бит защиты кода.

CP = 1: Код защиты выключен

CP = 0: Код защиты включен

Программный код, который записан в кристалл, может быть защищен от считывания при помощи установки бита защиты (CP) в слове конфигурации в ноль. Содержимое программы не может быть прочитано так, что с ним можно было бы работать. Кроме того, при установленном бите защиты становится невозможным изменять программу. Тоже относится и к содержимому памяти данных EEPROM.

Если установлена защита, то бит CP можно стереть только вместе с содержимым кристалла. Сначала будет стерта EEPROM программная память и память данных и в последнюю очередь бит защиты кода CP.

Встроенные генераторы работоспособны при определенных номиналах питающего напряжения (табл. 1.1):

Таблица 1.1

Параметры встроенных генераторов МК PIC16F84A

Vdd (напр. Питания)	**OSC mode** (тип генератора)	**Max Freq** (макс. частота)
2..3 V	RC	2 MHz
	LP	200 kHz
3..6 V	RC	4 MHz
	XT,LP	200 kHz
4,5...5,5 V	HS	10 MHz

PIC16F84A-XT, -HS или -LP требуют подключения кварцевого или керамического резонатора к выводам OSC1 и OSC2. Маркировка следующая: XT - стандартный кварцевый генератор, HS - высокочастотный кварцевый генератор, LP - низкочастотный генератор для экономичных приложений.

Когда не предъявляются требования к быстродействию и к точности по времени, OTP кристалл, например PIC16F84A-RC, позволяет сэкономить деньги и реализовать простой RC генератор.

Нумерация и назначение выводов микроконтроллера в рабочем режиме и в режиме программирования приведено на рис. 1.2 и в табл. 1.2.

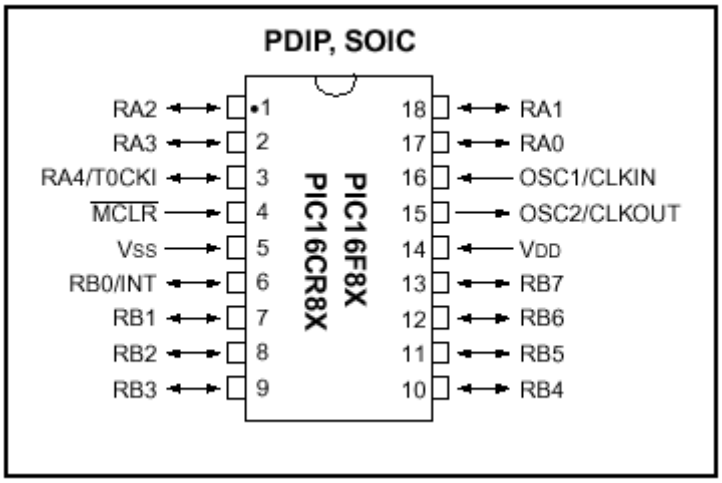


Рис. 1.2 Назначение выводов PIC16F84A.

Для изучения системы команд МК PIC16F84A имеется файл помощи **Piclab32.hlp**. В этом файле приведены сведения обо всех командах микроконтроллера.

Таблица 1.2

Назначение выводов МК PIC16F84A

Обозначение	**Нормальный режим**	**Режим записи EEPROM**
RA0 – RA3	Двунаправленные линии ввода/вывода. Входные уровни ТТЛ	
RA4/T0CKI	Вход через триггер. Линия порта ввода/вывода с открытым стоком или счетный вход для таймера/счетчика TMR0	
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 резонатора, RC или вход внешней тактовой частоты	
OSC2/CLKOUT	Генератор, выход тактовой частоты в режиме RC генератора, в остальных случаях – для кварцевого резонатора	
Vdd	Напряжение питания	Напряжение питания
Vss	Общий(земля)	Общий

1.2. *Разработка микропроцессорных систем*

В общем случае проектирование электронной системы с использованием микроконтроллера в качестве центрального узла состоит из нескольких этапов, основные из них:

- разработка аппаратной части;
- разработка программного обеспечения;
- тестирование созданного устройства.

Разработка аппаратной части подразумевает разработку и подключение периферийных устройств, исполнительных механизмов, а также устройств управления к МК. В зависимости от поставленной задачи выбирается микроконтроллер, способный реализовать необходимые функции. На данном этапе необходимо также рассматривать согласование уровней сигналов МК и уровней сигналов периферийных устройств. Составляется блок-схема аппаратной части на основе которой составляется логическая схема системы.

После разработки аппаратной части необходимо разработать программное обеспечение для управления всей системой, включая и подключенные периферийные устройства. Для начала составляется блок-схема процесса решения нужных задач (в случае сложных систем), обеспечивающих реализацию заданных требований. Реализуются блоки в отдельности и затем связываются воедино. После написания программы необходимо проверить правильность ее работы перед записью в память МК. Для этих целей используются специальные отладчики.

После разработки и отладки программной и аппаратной части производится запись данных в память МК и тестирование созданной системы на соответствие изначально предъявляемым требованиям.

2. Лабораторный комплекс

2.1. *Комплекс средств написания и отладки управляющей программы для МК*

В качестве среды разработки и отладки управляющей программы для МК используется фирменный пакет MPLAB IDE v8.76 или выше.

Пакет содержит компиляторы языка ассемблер, а также С-компилятор и поэтому программы можно писать на языке высокого уровня, если места в памяти МК достаточно.

После компиляции ассемблерных или С-файлов MPLAB строит файл, в котором располагаются данные для записи в память микроконтроллера.

В состав пакета входят несколько симуляторов, которые имитируют работу МК в реальном режиме времени. Это позволяет следить за ходом выполнения программы, создавать внешние воздействия, просматривать конечный результат выполнения программы.

Отладчик делает использование пакета очень удобным. Он позволяет проверить работоспособность программ, не занося данных в память контроллер. Во время проверки возможен просмотр содержимого всех доступных регистров контроллера.

Программная среда отладки предоставляет следующие возможности:

1. Отладка программы на уровне исходного текста ассемблера.
2. Контроль и модификация содержимого ячеек памяти, регистров и портов ввода/вывода.
3. Несколько режимов прогона программы: автоматический, пошаговый, с остановкой по контрольным точкам и прерываниям МК.
4. Производит конфигурирование адресного пространства для конкретного типа МК.
5. Контролирует обращение к несуществующим в адресном пространстве выбранного МК ячейкам памяти.

2.1.1. *Настройки MPLAB*

После запуска пакета MPLAB необходимо произвести некоторые настройки.

Для начала необходимо выбрать устройство (тип контроллера), для которого планируется разрабатывать программу. Это необходимо для того, чтобы сконфигурировать среду под определенный контроллер (настройка областей доступной памяти, порты ввода/вывода). Выбрать МК можно с помощью пункта *Select Device* в меню *Configure* (рис. 2.1).

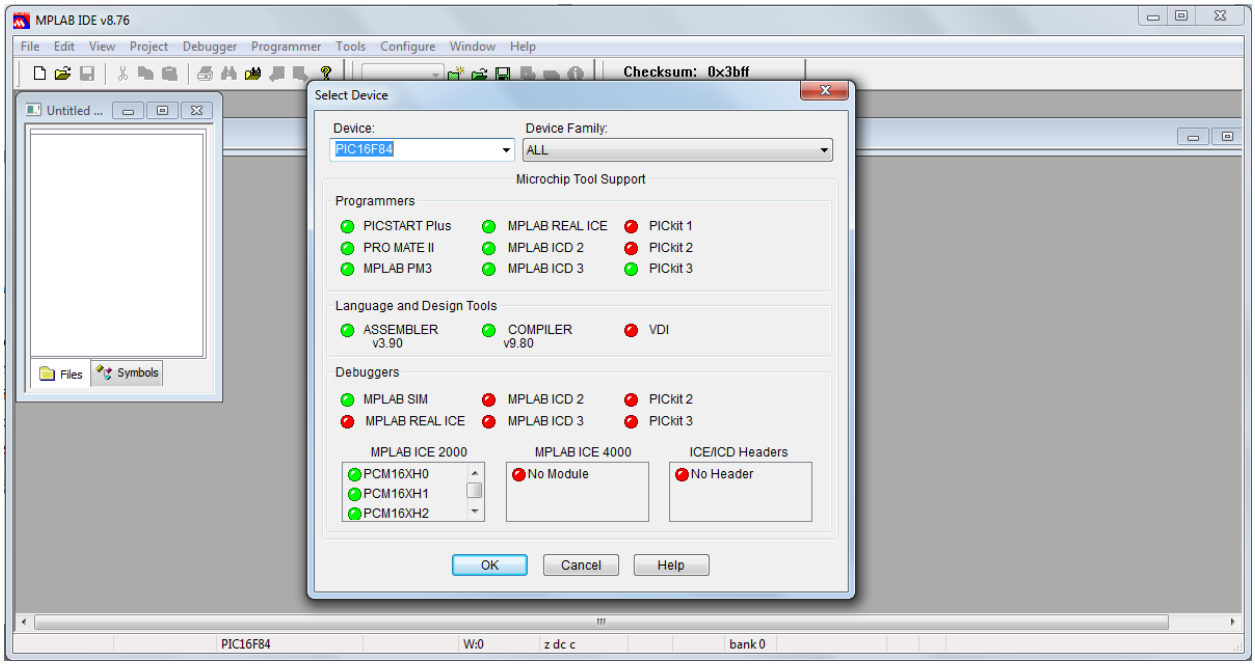


Рис. 2.1 Окно *Select Device.*

После выбора этого пункта появится список поддерживаемых типов МК. В данном списке необходимо выбрать нужный контроллер (для выполнения работ - PIC16F84A). Так же отображается список симуляторов и программаторов, которые поддерживают выбранный тип МК.

После выбора МК необходимо выбрать средство отладки (симулятор), с помощью которого в дальнейшем будет производиться отладка программы. Для этого в меню *Debugger* необходимо выбрать пункт *Select Tool* и из представленного списка доступных симуляторов выбрать нужный (для контроллера PIC16F84A используется симулятор MPLAB SIM).

MPLAB SIM-это встроенный в среду разработки MPLAB IDE симулятор для PIC-контроллеров. Средства отладки симулятора MPLAB SIM помогают пользователям при отладке программ для выбранного МК.

Возможности MPLAB SIM:

- модификация объектного кода и возможность перезапустить программу немедленно
- ввод внешних воздействий в имитированный процесс
- установка значений сигналов на выводах МК и значений регистров в определенные значения времени
- отслеживание хода выполнения программы

Для установки некоторых параметров симулятора необходимо выбрать пункт *Settings* в меню *Debugger*. Здесь можно выбрать частоту МК, сброс по таймеру, точки останова и т. д.

Для МК существует конфигурационное слово, которое изначально определят некоторые параметры. Биты конфигурации, указанные в исходном тексте программы, не будут настраивать параметры работы симулятора. Для установки конфигурации МК нужно использовать пункт *Configuration Bits* в меню *Configuration.* Это сделано для того, чтобы была возможность имитировать работу МК в различных режимах без изменения исходного текста программы. Здесь можно активировать или деактивировать сторожевой таймер(Watchdog Timer), выбрать степень защиты кода(Code Protected), таймер задержки по включению питания(Power Up Timer), колебательный контур(Oscillator).

Для правильной работы отладчика необходимо задать пути к исполняемым файлам ассемблера (mpasmwin.exe) и компоновщика (mplink.exe). Для этого в меню *Project* необходимо выбрать пункт *Set Language Tool Location*. В появившемся диалоговом окне (рис. 2.2) необходимо в списке используемых файлов определенного симулятора указать пути к данным файлам с помощью кнопки *Browse.* Обычно эти файлы находятся в папке ...\MPLAB IDE\MCHIP_Tools.

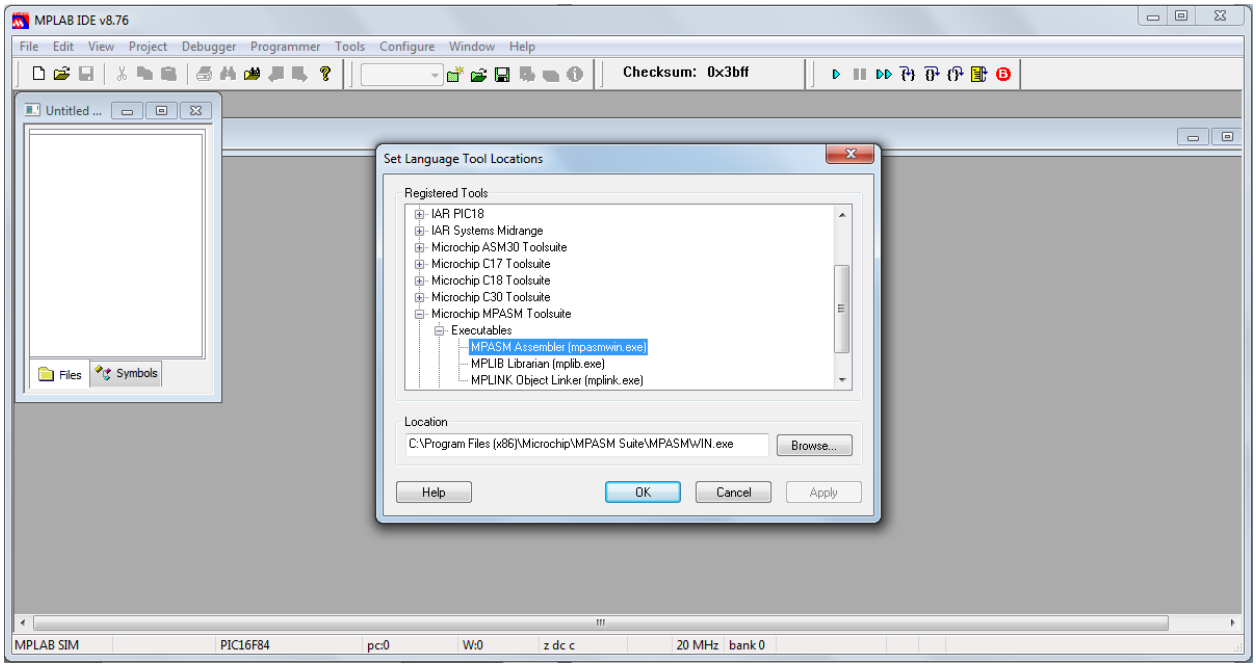


Рис. 2.2 Диалоговое окно *Set Language Tool Location.*

Представленное окно появляется только тогда, когда открыт проект. При желании можно выбрать компиляторы третьих фирм.

2.1.2. *Создание нового проекта*

Две основных характеристики MPLAB IDE – проекты(Project) и рабочие области(Workspace).

В программной среде MPLAB IDE используются проекты, чтобы упростить управление и отладку прикладных программ. Проект MPLAB IDE – это группа файлов, необходимых для работы различных инструментальных средств среды проектирования. Проект состоит из узла компиляции и одного или нескольких исходных узлов. Исходными узлами являются файлы, написанные на ассемблере или Си, объектные файлы и файлы сценария компоновщика (линкера).

Рабочие области подразумевают использование многочисленных проектов с одной и той же конфигурацией. Рабочая область содержит информацию о выбранном устройстве (МК), отладочном средстве и/или программаторе, открытых окнах и их позиции и других установочных параметрах конфигурации.

Если работать с единственным ассемблеровским файлом, можно использовать *Quickbuild*, чтобы откомпилировать один файл и не создавать проект.

Для того чтобы написать новый файл на языке ассемблер необходимо в меню *File* выбрать пункт *New*. После написания файла, его необходимо сохранить, иначе с ним невозможно будет работать.

Для создания нового проекта необходимо в меню *Project* выбрать пункт *New*. После этого необходимо указать имя проекта и его расположение.

Чтобы добавить файлы в проект необходимо в меню *Project* выбрать пункт *Add Files to Project* и указать, какие файлы необходимо добавить. Для удаления файлов из проекта необходимо выбрать пункт *Remove File From Project.*

После создания нового проекта в окне рабочей области появится информация об исходных файлах, файлах библиотек, файлах компоновщика и т. д.

В меню *Project* необходимо выбрать пункт *Select Language Toolsuite* для выбора блока средств разработки (для контроллера PIC16F84A нужно выбрать Microchip MPASM Toolsuite).

Необходимо также указать пути для исходных и конечных файлов, файлов библиотек, файлов сценариев компоновщиков, файлов-заголовков. Это делается в пункте *Build Options* меню *Project* (рис. 2.3).

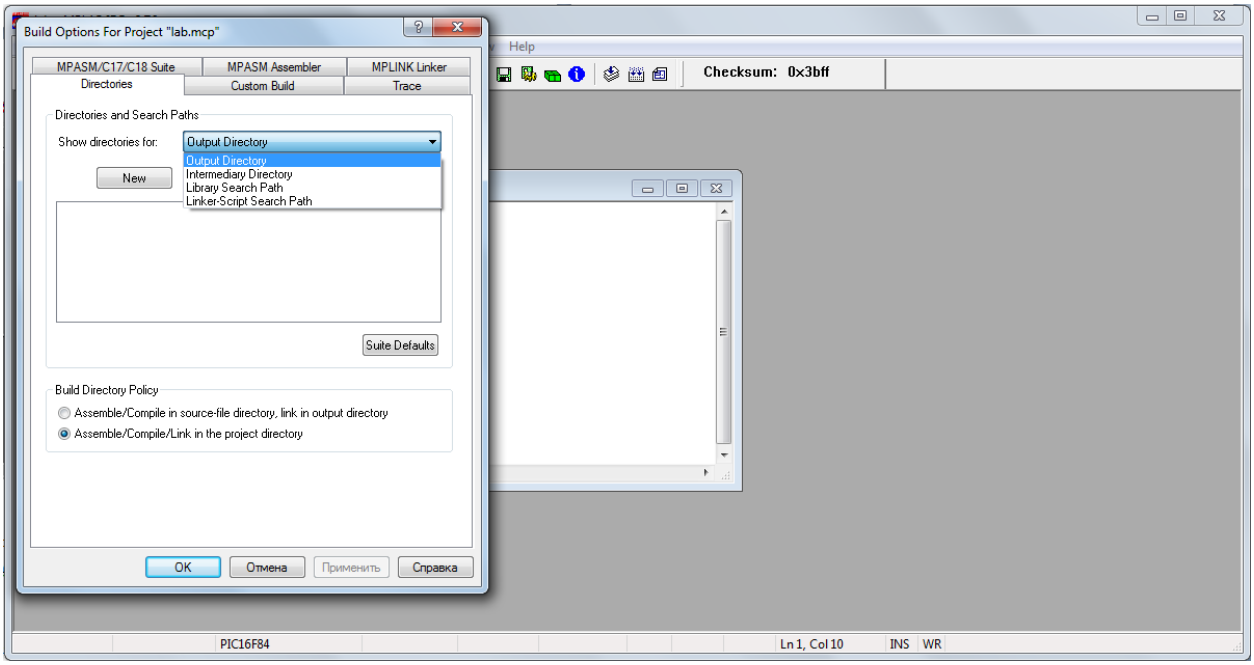


Рис. 2.3 Окно *Build Options.*

Здесь же можно установить параметры выходного HEX-файла. Файлы-заголовки(.inc) находятся в папке ...\MPLAB IDE\MCHIP_Tools, файлы сценариев(.lkr)- в папке ...\MPLAB IDE\MCHIP_Tools\LKR.

2.1.3. *Компиляция и устранение ошибок*

Для того чтобы убедиться, что ошибок нет, необходимо откомпилировать созданный файл.

Сформировать все файлы в активном проекте можно с помощью меню правой кнопки мыши. Можно также в меню *Project* выбрать пункт *Build All.*

Для того чтобы сформировать только те файлы, которые изменились в проекте, необходимо выбрать пункт *Make* в меню *Project.*

После компиляции на экране появляется окно *Output,* в котором показан результат. Наличие ошибок указывается строкой с обозначением *Error().* В скобках указывается номер строки исходного файла, в которой обнаружена ошибка. При двойном нажатии кнопки мыши на строке ошибки происходит автоматический переход на указанную строку исходного файла.

При появлении сообщения *Error* в первую очередь необходимо проверить настройки отладчика. Для начала нужно убедиться в правильности выбора типа контроллера, иначе могут возникать ошибки записи в память по несуществующим адресам для выбранного МК. Затем необходимо проверить

правильность указания путей для исходных файлов, файлов компоновщиков. Также необходимо убедиться, что в качестве блока средств отладки выбран Microchip MPASM Toolsuite.

Если вышеперечисленные настройки безошибочны, то необходимо искать ошибки в исходном тексте программы. Часто это могут быть синтаксические ошибки (неправильно написано имя команды), использование неопределенных ранее переменных или неправильный выбор страницы определенных адресов памяти.

В окне *Output* указываются также некритические ошибки *Warning* (предупреждение) и *Message* (сообщение). Эти ошибки позволяют дальше отлаживать программу.

После компиляции автоматически создается файл с указанием наличия или отсутствия ошибок.

После того как проект откомпилирован и в окне *Output* нет сообщений об ошибках, можно запускать программу на исполнение и проверять ее работоспособность с помощью различных функций встроенного отладчика.

2.1.4. *Использование отладчика*

В состав пакета MPLAB входит отладчик, позволяющий проверить работоспособность программы перед записью в память контроллера.

Стандартные окна отладчика позволяют рассматривать содержимое памяти программ, данные или другие типы памяти выбранного МК, чтобы помочь в отладке приложения. Дополнительные окна отладчика могут появиться в меню в зависимости от выбранного симулятора. Выбор необходимых окон отладчика производится в меню *View*.

В меню *View* входят следующие окна:

- *File Registers*-отображает содержимое памяти. Можно выбрать вид отображения с помощью кнопок, расположенных внизу окна. Содержимое некоторых ячеек памяти можно изменять вручную (если эти ячейки не являются недоступными для изменения)
- *Program Memory*-отображает содержимое памяти программ МК. Путем выбора соответствующих значков в нижней части окна можно просматривать содержимое в различном виде: HEX-кодировка, машинные коды. С помощью нажатия правой кнопки мыши в окне можно вызвать меню, содержащее различные функции для управления этим окном.
- *EEPROM*-отображает содержимое EEPROM памяти МК.
- *Special Function Registers*-отображает специальные регистры МК и их содержимое. Содержимое некоторых регистров можно менять, если они доступны для изменения.
- *Watch*-окно, позволяющее отслеживать изменения определенных переменных, содержимого регистров, портов. Для добавления регистра необходимо выбрать его из списка и затем нажать на Add SFR, а для того, чтобы добавить переменную необходимо использовать кнопку Add Symbol и список, находящийся справа от нее. В окне отображается адрес переменной или регистра, его имя и значение.

- *Hardware Stack*-показывает содержимое стека. Здесь: **TOS** (если доступно) – указатель вершины стека; **Stack Level**- номер уровня стека (общее число уровней зависит от выбранного МК); **Return Address**-возвращает адрес в стек. С помощью правой кнопки мыши в окне *Hardware Stack* можно войти в меню настроек.

Внизу основного окна оболочки MPLAB IDE располагается линейка состояния. Она отображает состояние выполнения программы. Назначение полей линейки состояния (слева направо):

- отображается выбранное средство отладки;
- отображается тип выбранного МК;
- текущее значение счетчика команд;
- значение регистра W;
- состояние битов статуса. Верхний регистр-бит установлен, нижний регистр-бит сброшен.

Отладчик позволяет запускать программу целиком, а также имеется возможность пошагового выполнения. Это очень удобно для отладки программ. Во время пошагового выполнения программы можно наблюдать за изменением состояния нужных регистров, памяти, переменных. Запустить программу на выполнение можно путем выбора пункта *Run* в меню *Debugger* или нажать *F9*. Запустить программу в пошаговом режиме можно с помощью *Step Into* (выполняется одна инструкция МК, а затем программа останавливается, обновляя значения регистров) или *Step Over* (“выполнить текущую инструкцию”, по команде Step Over инструкции CALL выполняются за один шаг (полностью выполнив подпрограмму), затем программа останавливается в следующем за инструкцией CALL адресе) в меню *Debugger*. Также это можно сделать с помощью клавиш *F7* и *F8* соответственно.

Перед повторным запуском программы необходимо сбросить контроллер с помощью *Processor Reset* меню *Debugger* или путем нажатия *F6.* Также необходимо очистить память с помощью пункта *Clear All Memory* в меню *Debugger*.

Меню *Debugger* позволяет останавливать программу в выбранных контрольных точках. Для этого используется пункт *Breakpoints*. После выбора этого пункта появится диалоговое окно, где нужно задать точки останова путем задания номеров строк программы. Появится список всех заданных точек. В этом списке можно активировать/деактивировать точки останова. Точки останова также можно задать в окне памяти программ (*View>Program Memory*) с помощью правой кнопки мыши. После каждого обновления с помощью команды *Make* необходимо заново указывать точки останова.

Для того чтобы наблюдать за количеством машинных циклов и за временем выполнения программы используется пункт *Stopwatch* меню *Debugger*. Окно *Stopwatch* необходимо при написании программ временной задержки.

Отладчик также позволяет задавать внешние воздействия на определенные выводы контроллера. Описать эти воздействия можно выбрав пункт *Stimulus* в меню *Debugger*. На вкладке *File Stimulus* с помощью кнопки *Add* в области *Input Files* можно загрузить файл воздействий, сохраненный ранее. На вкладке *Pin Stimulus* можно задать новые воздействия на выводы МК с помощью кнопки *Add Row*. Здесь можно выбрать синхронные/асинхронные воздействия, номер вывода уровень сигнала. Здесь же можно сохранить созданный список в файл, который при необходимости можно будет загрузить.

Необходимо учитывать, что после любого изменения в проекте (файле) необходимо обязательно произвести компиляцию с помощью пункта *Make* в меню *Project*. Это нужно для того, чтобы все новые изменения в проекте (файле) изменяли ход программы.

2.2. *Комплекс средств программирования МК*

Для прошивки и отладки МК используется программный пакет MPLAB.

Для начала необходимо подключить адаптер к одному из портов USB (рис.2.4).

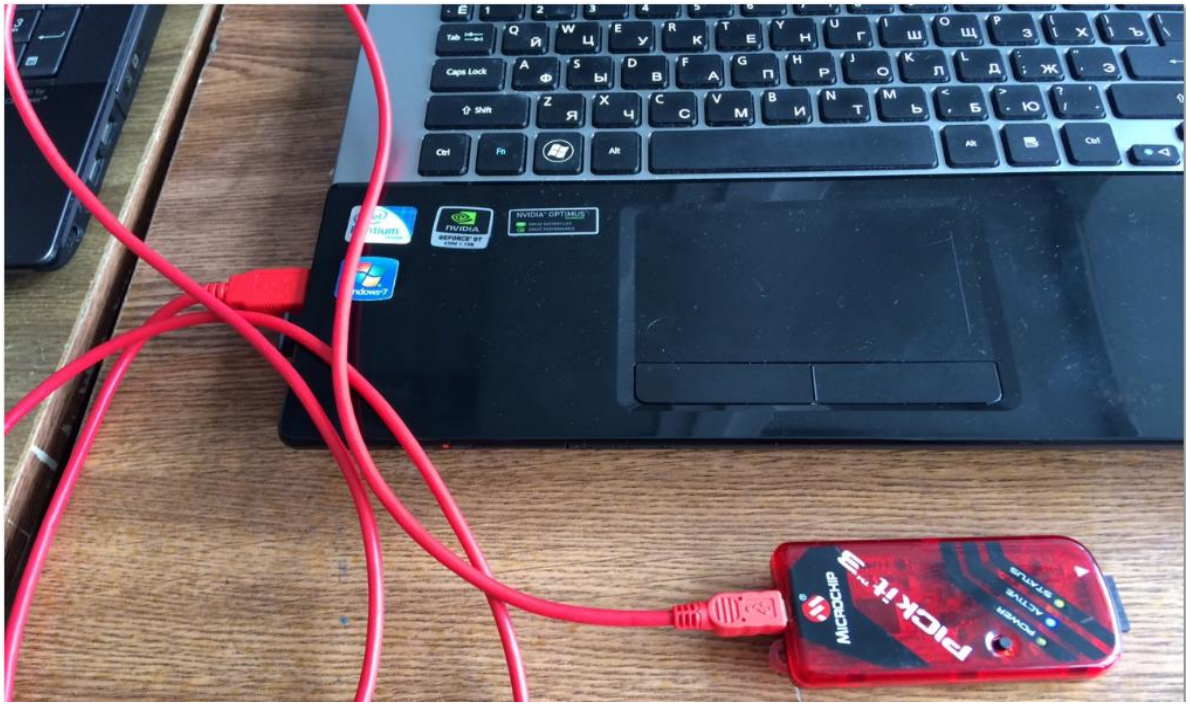


Рис. 2.4 Подключение адаптера.

Затем необходимо выбрать пункт *PICkit 3* в меню *Programmer* (рис. 2.5).

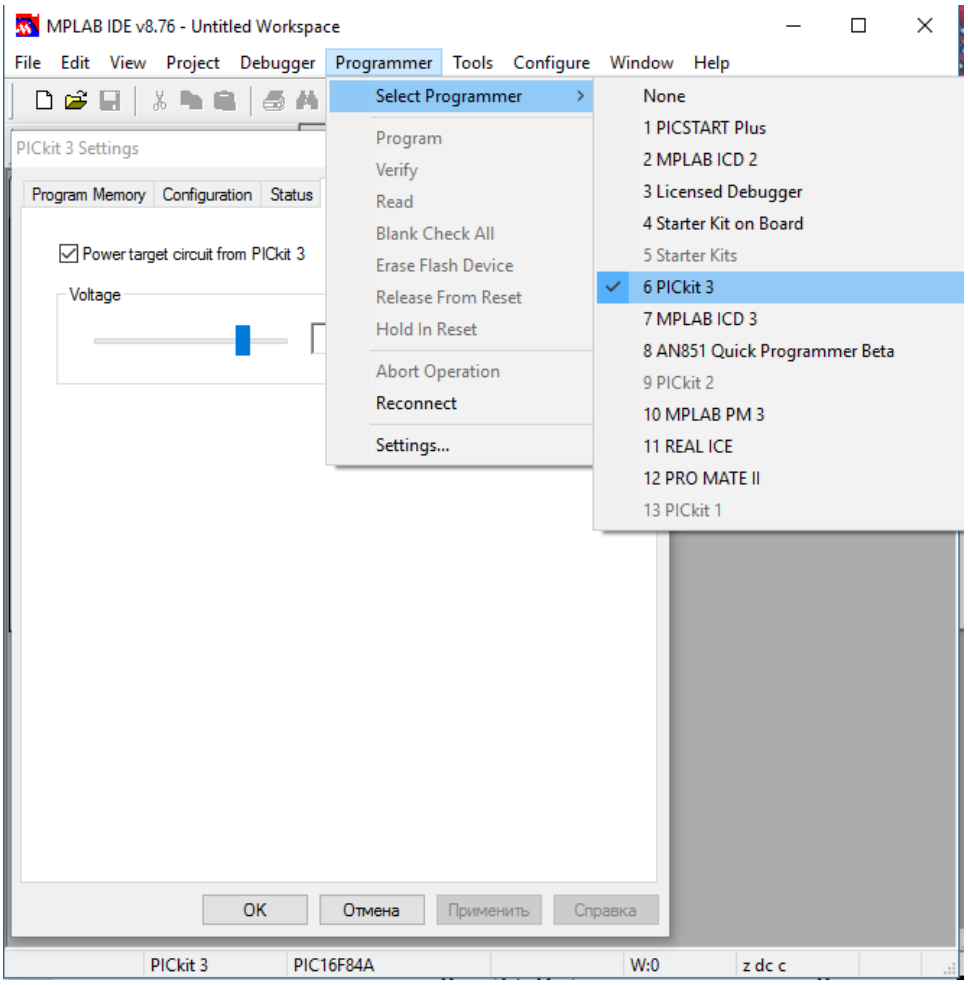


Рис. 2.5 Выбор программатора – *PICkit 3*.

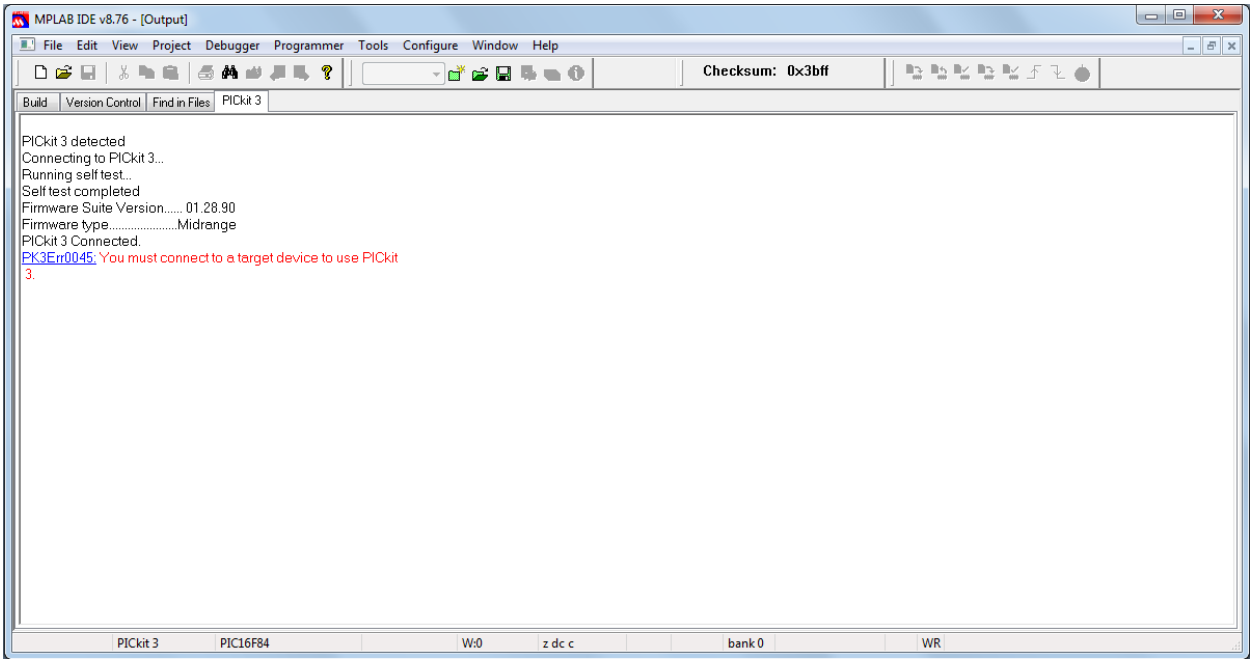


Рис. 2.6 Подключение *PICkit 3*

Далее необходимо указать в меню *Programmer* > *Settings*... в закладке *Power*, что напряжение питание +5 вольт на отладочную плату будет подаваться от *PICkit 3*

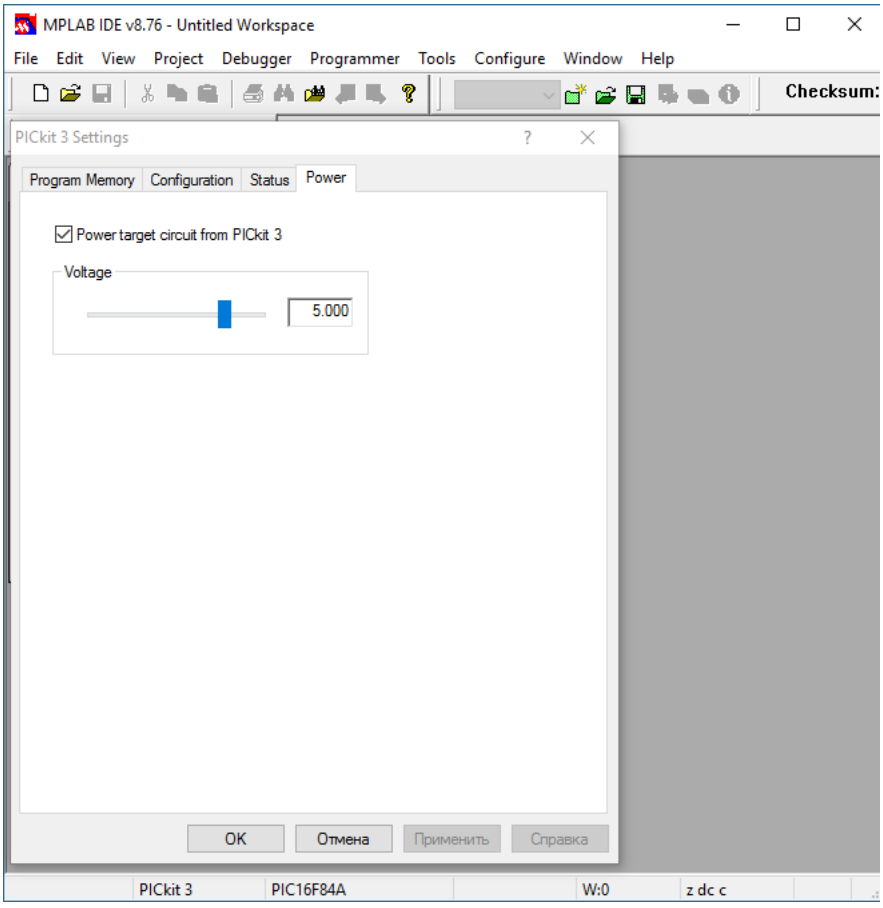


Рис. 2.7 Выбор источника подачи напряжения питания на отладочную плату.

Далее к адаптеру следует подключить плату (рис. 2.8), при подключении, MPLAB сразу распознает устройство (рис. 2.8).

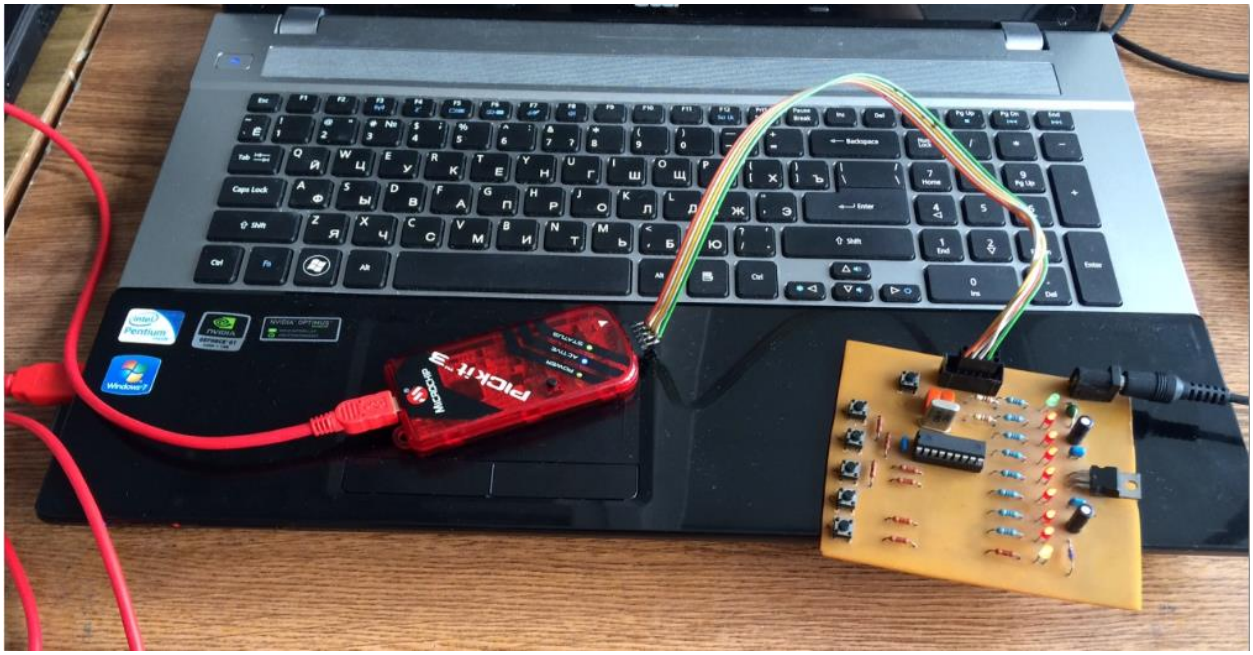


Рис. 2.8 Подключение отладочной платы.

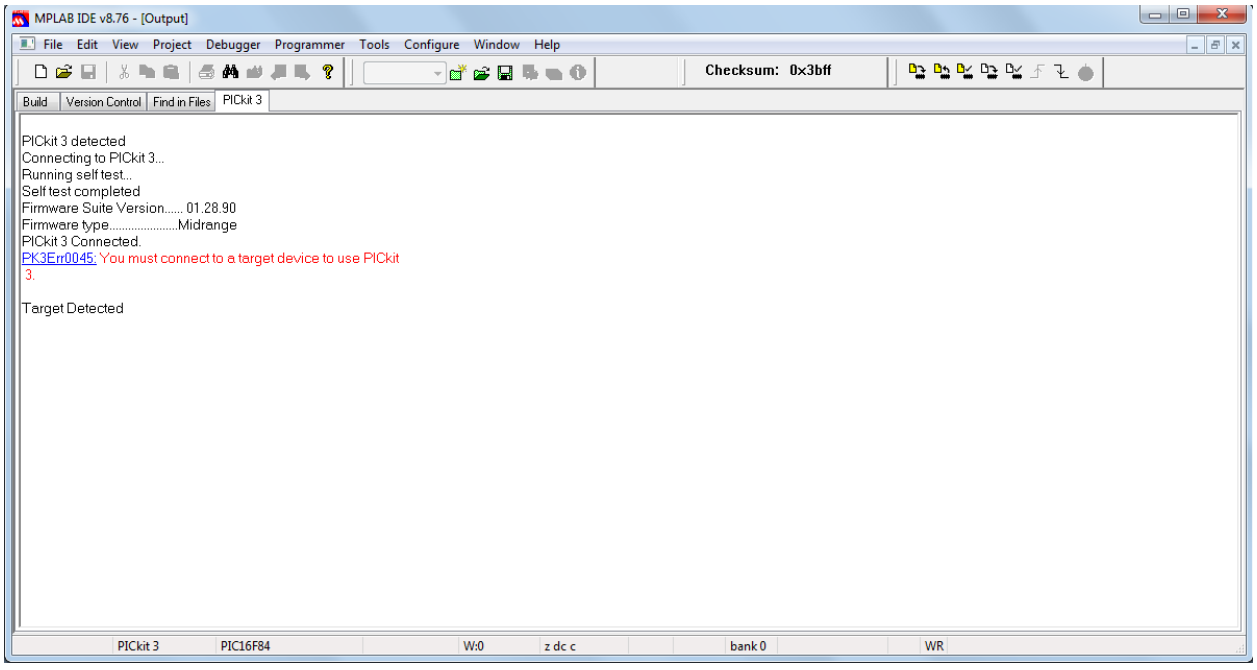


Рис. 2.9 Распознавание устройства при подключении отладочной платы.

Что бы начать программирование нужно нажать *Program* на панели управления (рис. 2.10).

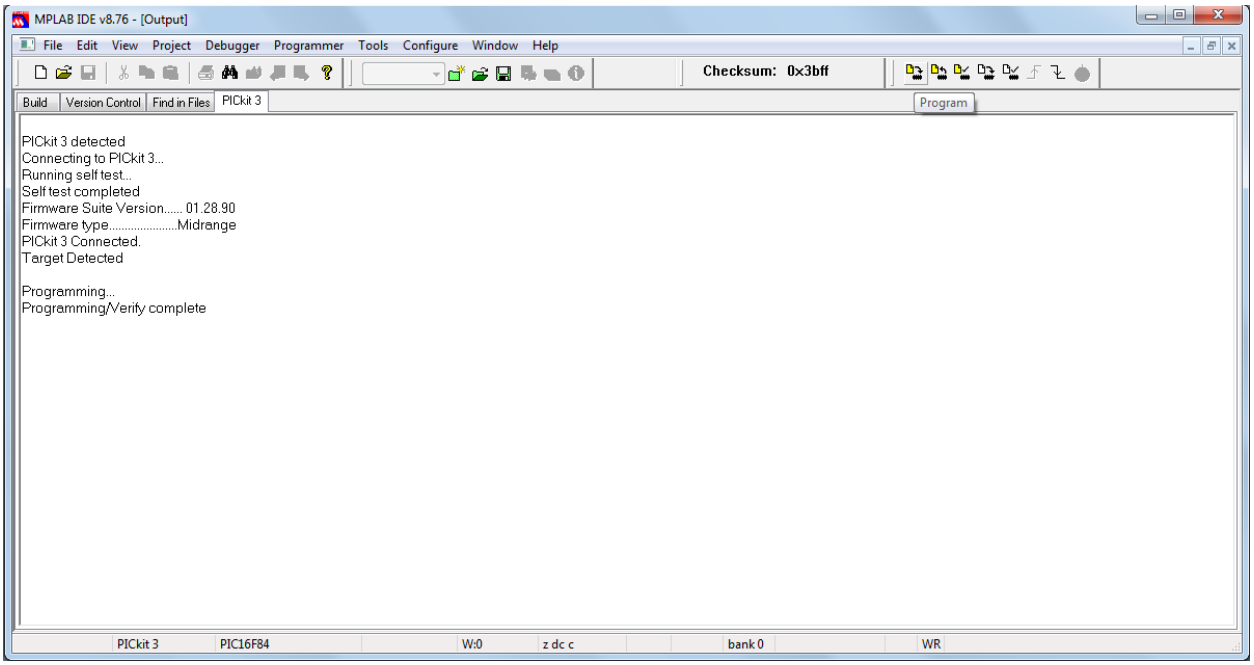


Рис. 2.10 Программирование отладочной платы.

Для считывания программы из памяти МК необходимо нажать *Read* (рис. 2.11).

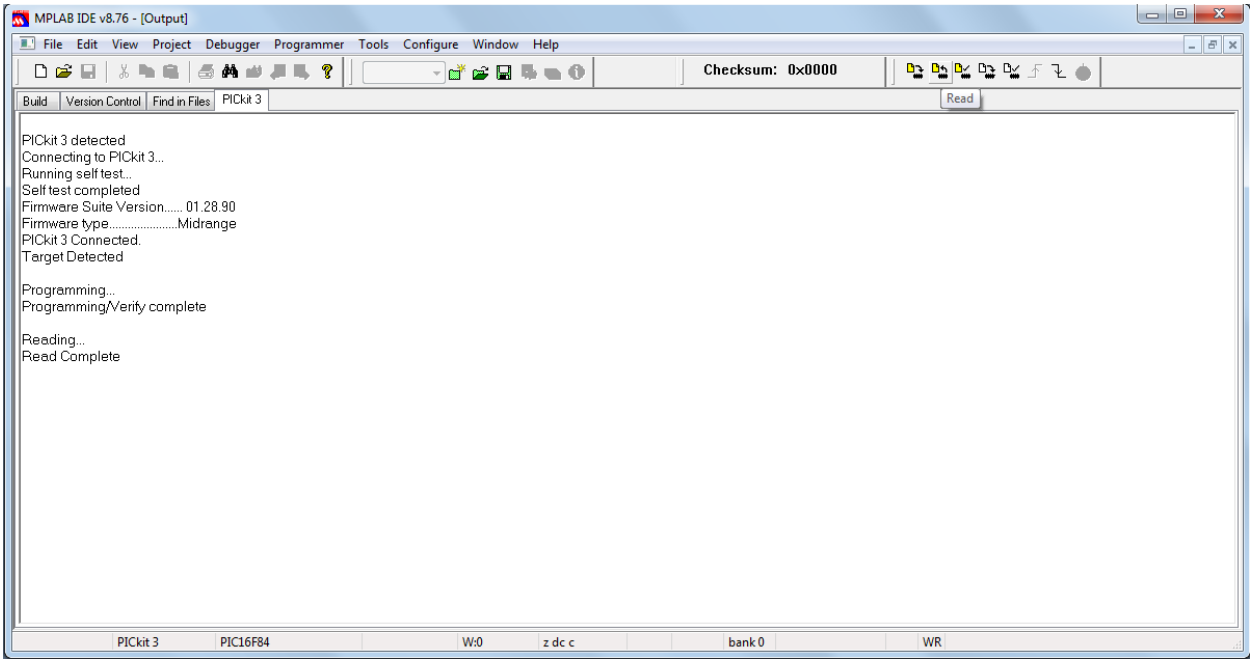


Рис. 2.11 Считывание.

Для проверки на соответствие записанной программы - *Verify* (рис. 2.12).

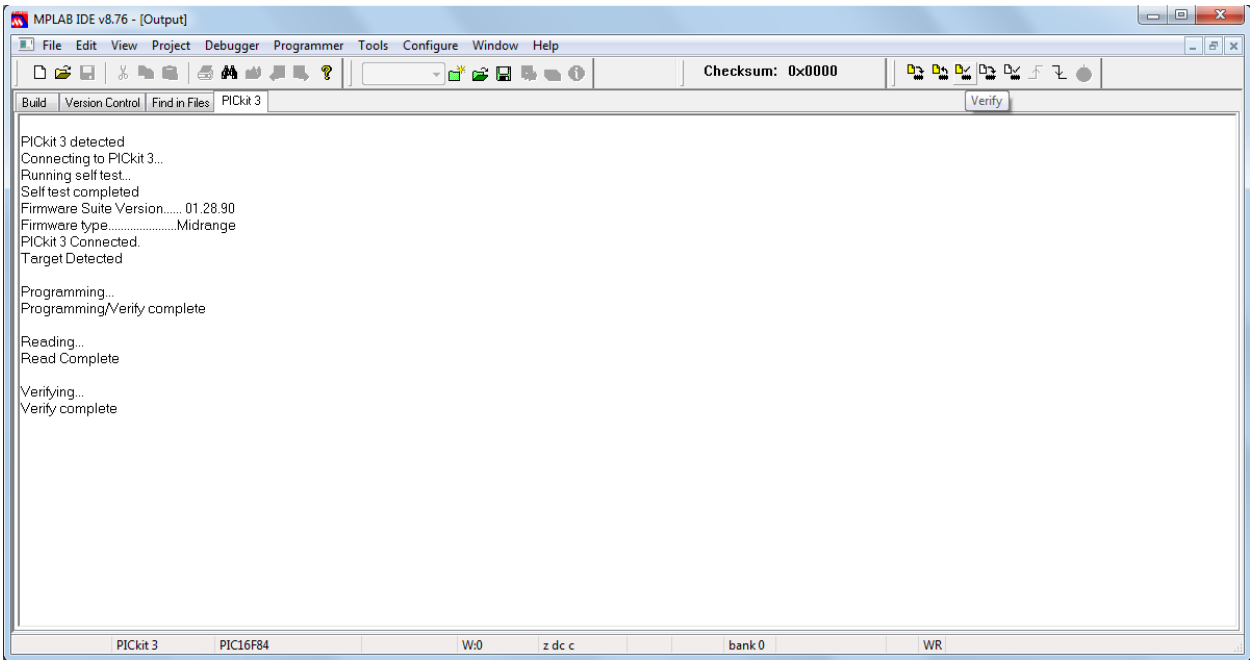


Рис. 2.12 Проверка на соответствие записанной программы.

2.3. *Лабораторный стенд*

Для проверки работоспособности программ, создаваемых в процессе выполнения лабораторных работ, используется отладочная плата. Ее схема приведена на рис. 2.13.

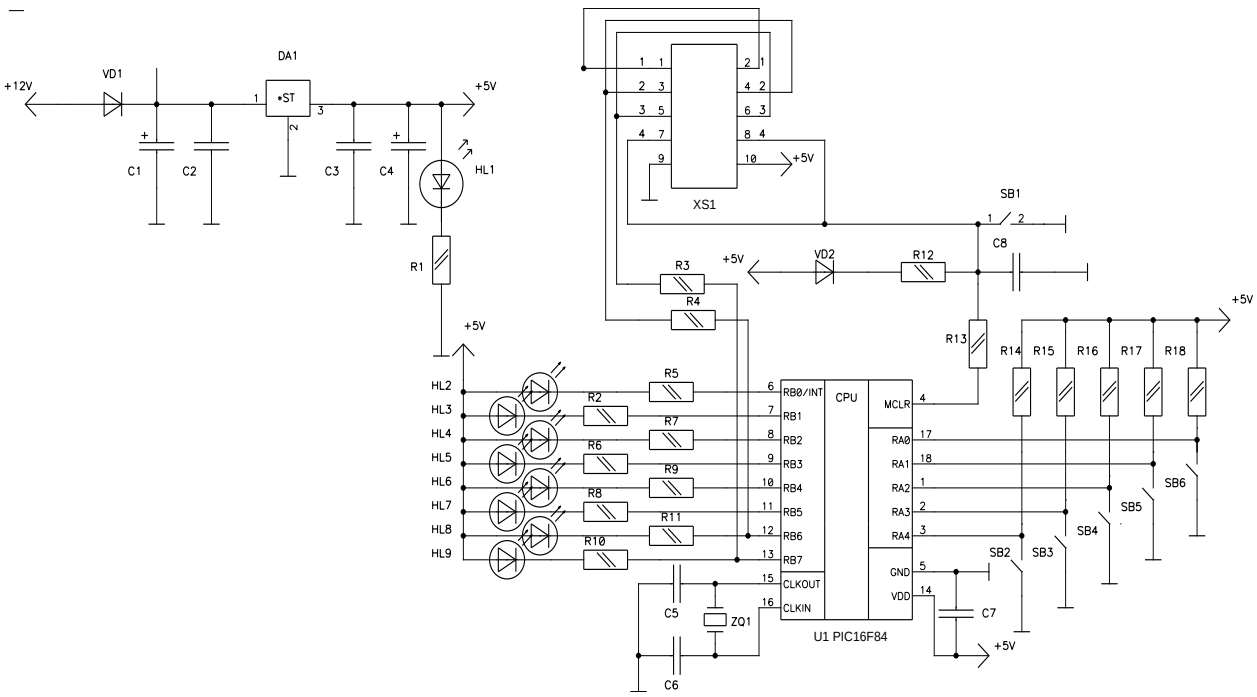


Рис. 2.13 Схема электрическая принципиальная отладочной платы.

Светодиоды HL2-HL9 предназначены для индикации состояния линий порта B. Кнопочные выключатели SB2-SB6 подключены к линиям порта A и являются элементами управления.

Размещение элементов на отладочной плате показано на рис. 2.14.

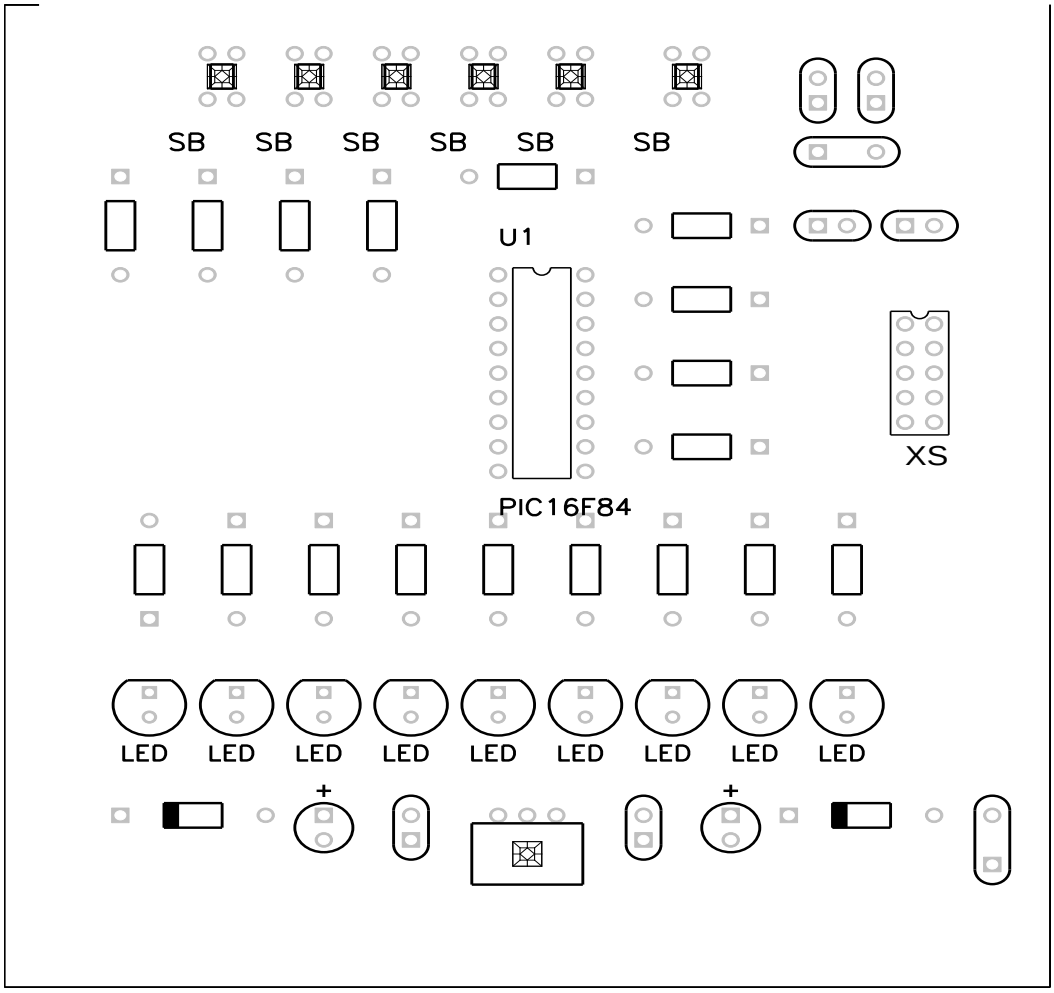


Рис. 2.14 Размещение элементов на отладочной плате.

На представленном рисунке подписаны элементы, необходимые для работы с отладочной платой. Здесь: SB- выключатели, предназначенные для управления устройством; LED- светодиоды, предназначенные для индикации; U1- микроконтроллер; XS- гнездо для программирования. Остальные элементы – элементы для работы МК и согласования сигналов.

Основная литература

1. В. Я. Хартов Микропроцессорные системы: учеб. пособие для вузов. – М.: Академия, 2010. – 352 с.
2. Однокристальные микроконтроллеры Microchip: PIC16C8X./ Пер. с англ.// Под ред. А.Н. Владимирова.- Рига.: ORMIX, 1996.- 120с.: ил.

Дополнительная литература

1. Микроконтроллеры. Выпуск 5. Справочник. – М.: Издательский дом ДОДЭКА, 2000. – 272 с.
2. Современные микроконтроллеры: Архитектура, средства проектирования, примеры применения, ресурсы сети Internet. © «Телесистемы». Под ред. Коршуна И.В.: Составление, пер. с англ. и литературная обработка Горбунова Б.Б., -М.: Издательство «Аким», 1998.- 272с.,ил.

3. Выполнение лабораторной работы

Выполнение лабораторной работы можно разделить на следующие этапы:

1. Ознакомление с теорией в ходе которого производится знакомство со структурой МК, его возможностями и системой команд. Также необходимо ознакомиться со средствами отладки и программирования.
2. Выполнение приведенного примера.
3. Создание программных средств для МК:

   а) Создание алгоритма программы в соответствии с поставленной задачей.

   б) Создание управляющей программы для МК с помощью пакета MPLAB IDE. Написание программы на ассемблере.

   в) Проверка правильности работы на программном симуляторе с использованием необходимых возможностей отладчика.

   г) Исправление ошибок, выявленных на этапе проверки.

3. Запись данных в память микроконтроллера с помощью средств программирования.
4. Проверка работоспособности созданного устройства на лабораторном стенде и соответствие алгоритма его работы поставленной цели.

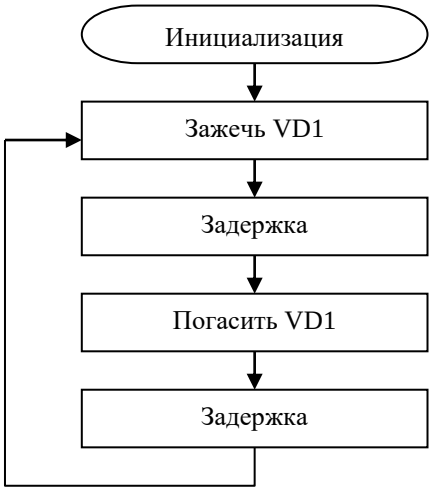
3.1. Пример разработки микропроцессорного устройства

Задание: Разработать генератор прямоугольных импульсов с частотой 0,5 Гц. Индикация высокого и низкого уровней производится светодиодом.

Выполнение задания: Микроконтроллер PIC16F84A имеет 13 портов ввода/вывода, из них на отладочной плате светодиоды подключены к порту B. В нашем случае требуется один разряд порта, настроенный на выход. Можно использовать, например, RB0.

Так как на лабораторном стенде уже имеются все необходимые элементы управления и индикации, можно считать, что аппаратная часть устройства готова. Осталось лишь разработать программную часть.

Структура программы:



Для написания и отладки программы используем пакет MPLAB.

После проведения настроек, указанных в 2.1 в нижней части экрана в линейке состояния должно появиться название симулятора (MPLAB SIM) и тип выбранного устройства (PIC16F84A). Если указанные названия не появились, то обратитесь к 2.1 данных методических указаний, если все прошло успешно, то теперь необходимо создать новый проект. Для этого воспользуемся меню *Project* и выберем там *New*. Появляется окно (рис. 3.1).

В появившемся диалоговом окне указываем название проекта и каталог, где будет храниться этот проект. В окне рабочей области должно появиться название созданного проекта и список включенных в него файлов.

Для создания нового файла необходимо выбрать *New* в меню *File*. Появляется окно (рис. 3.2).

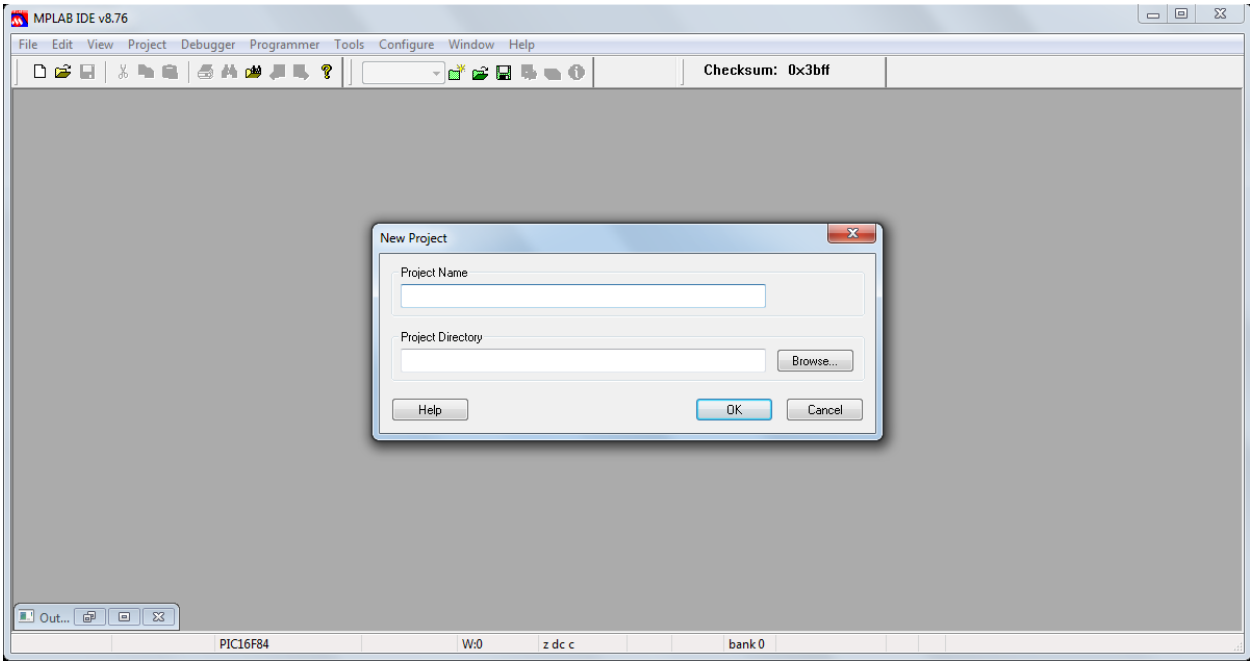


Рис. 3.1 Диалоговое окно *New Project*.

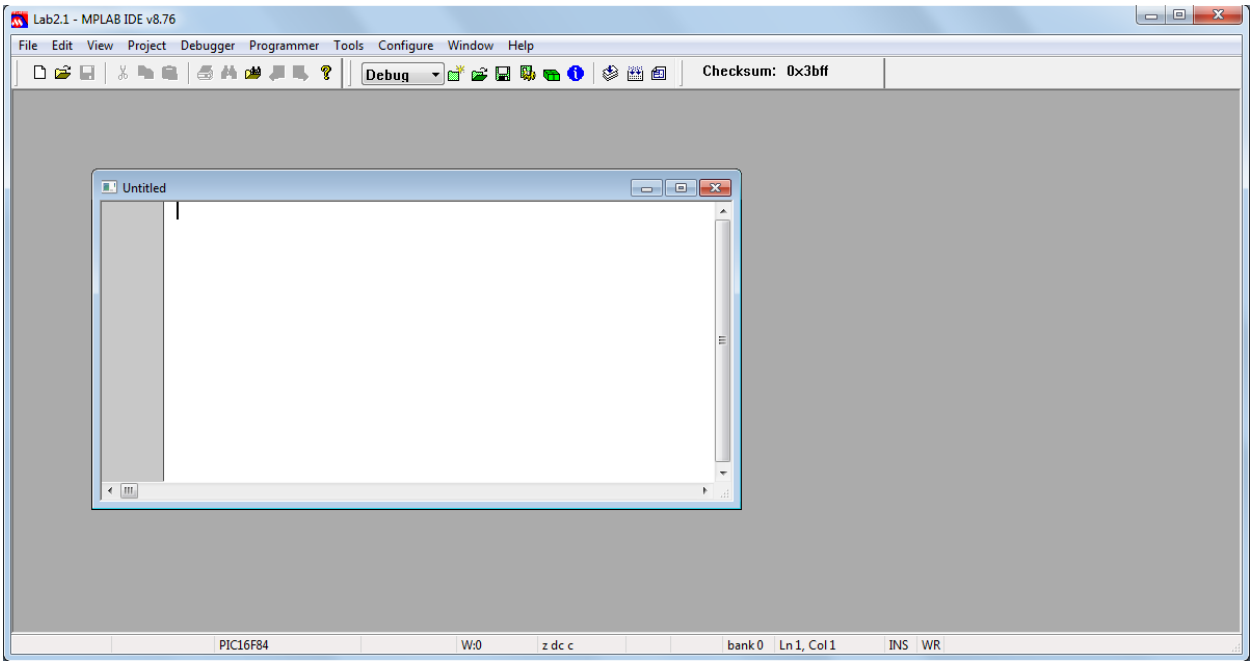


Рис. 3.2 Создание нового файла.

В появившемся окне необходимо написать текст исходной программы. После написания файла программы его необходимо обязательно сохранить с помощью пункта *Save* меню *File*.

Текст программы:

```
; используем процессор PIC16F84A, система исчисления десятичная
include "p16f84.inc"                    ; подключаем файл с описанием регистров
; Переменные
wt1             equ     0x0f
wt2             equ     0x10
wt3             equ     0x11
```

```
RB0             equ     0
;Код программы Устанавливаем вектор сброса
org     0
goto    Start
; Основная программа
Start
; Инициализируем переменные начальными значениями
bcf     STATUS,IRP          ; устанавливаем использование банка только 0 и1
bcf     STATUS,RP1          ; устанавливаем использование банка 0 и 1
bsf     STATUS,RP0          ; установка бита выбирает банк 1, сброс - банк 0
movlw 0x1E                  ; вывод RB0 как выход
movwf TRISB                 ; установили
bcf     STATUS,RP0          ; установили банк 0
Diode bsf       PORTB,RB0 ; установили RB0 в 1
call    Wait                ; подпрограмма задержки
bcf     PORTB,RB0           ; установили RB0 в 0
call    Wait                ; подпрограмма задержки
goto    Diode               ; Повторяем программу пока работаем
;------ конец основной программы----------------------------------------
;-------это подпрограмма-----------------------------------------------
Wait    movlw 0x05
movwf wt3
We3     movlw 0x00
movwf wt2
We2     movlw 0x00
movwf wt1
We1     decfsz  wt1,1   ;768 uS
goto    We1
decfsz wt2,1            ;198 mS
goto    We2
decfsz wt3,1            ;1 S
goto    We3
return
;-------конец подпрограммы-----------------------------------------------
end     ;конец всей программы
```

Теперь созданный файл программы необходимо добавить в проект. Для этого воспользуемся также меню *Project* и выберем там пункт *Add Files to Project* (рис. 3.3).

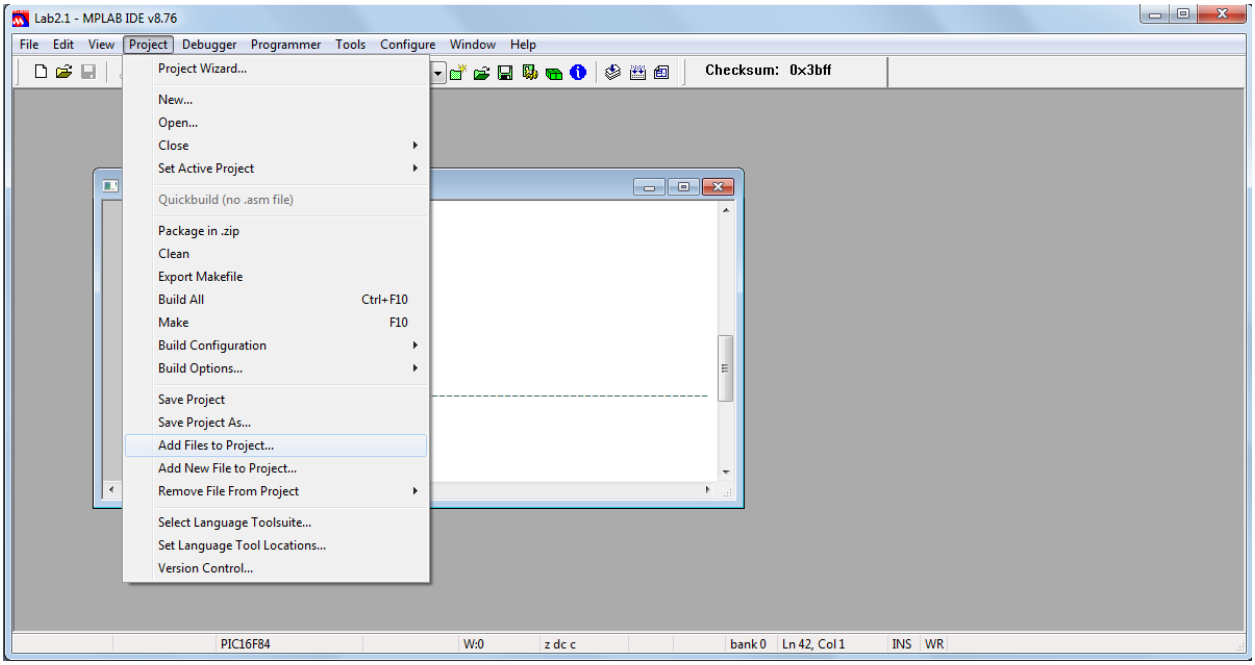


Рис. 3.3 Добавление файла в проект.

После этого в рабочей области проекта в списке включенных файлов в пункте *Source Files* должен появиться созданный файл.

Далее проект нужно откомпилировать. Выбираем пункт *Build All* в меню *Project*. В процессе компиляции (рис. 3.4) создаются файлы, необходимые для работы отладчика, а также файл для прошивки МК. После компиляции появляется окно результата, в котором отображается наличие ошибок, при появлении которых необходимо проверить настройки отладчика и текст программы.

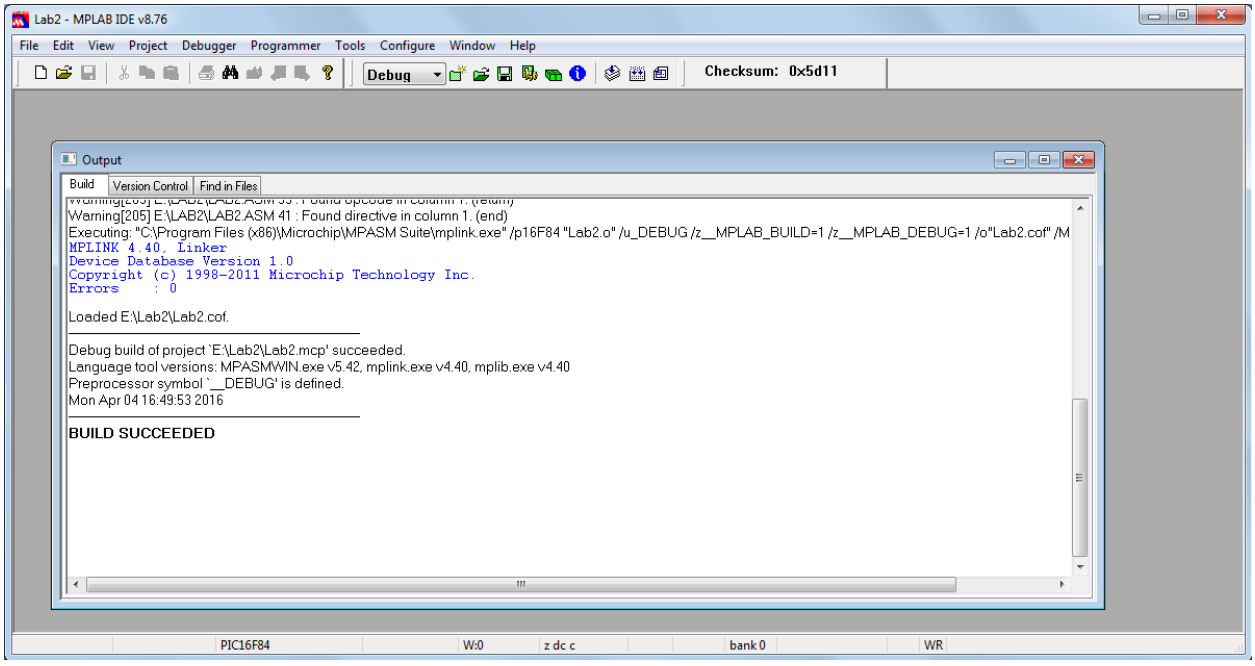


Рис. 3.4 Компиляция проекта.

Если критических ошибок нет, то можно приступать к просмотру хода выполнения программы. Для просмотра состояния портов и соответственно результатов выполнения программы необходимо открыть окно *Watch* в меню *View*.

В появившемся окне возле кнопки *Add SFR* необходимо выбрать *PORTB*, затем нажать кнопку. В окне появится адрес порта *PORTB* и его текущее значение (рис. 3.5).

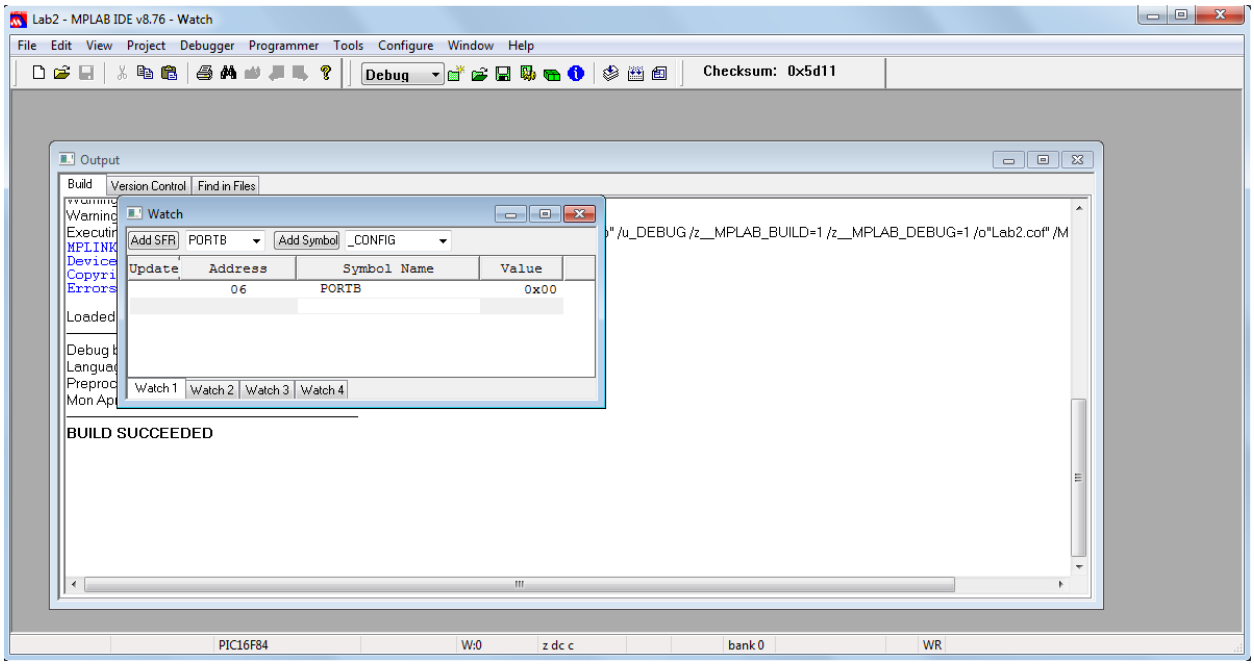


Рис. 3.5 Использование окна *Watch*.

Для просмотра хода и результатов выполнения программы удобнее всего воспользоваться пошаговым выполнением программы. Для этого необходимо перейти в режим отладчика (Рис. 3.6).

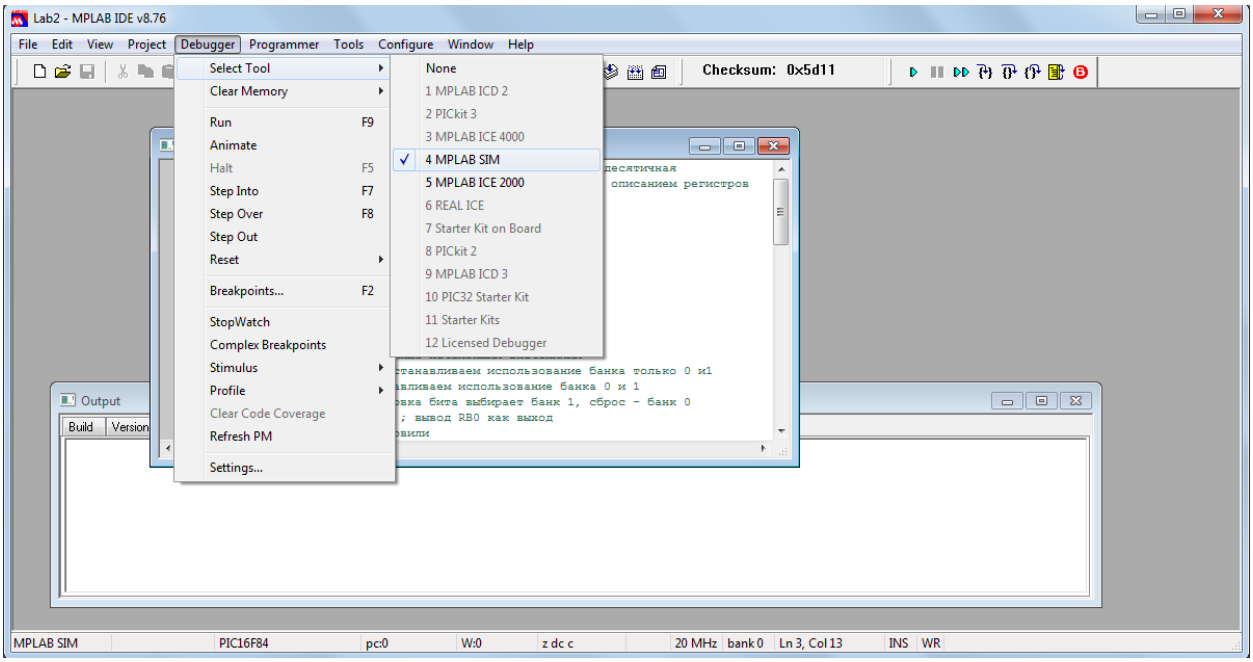


Рис. 3.6 Выбор отладчика.

Для начала необходимо сбросить контроллер с помощью *F6* или *Debugger>Reset*.(Рис. 3.7)

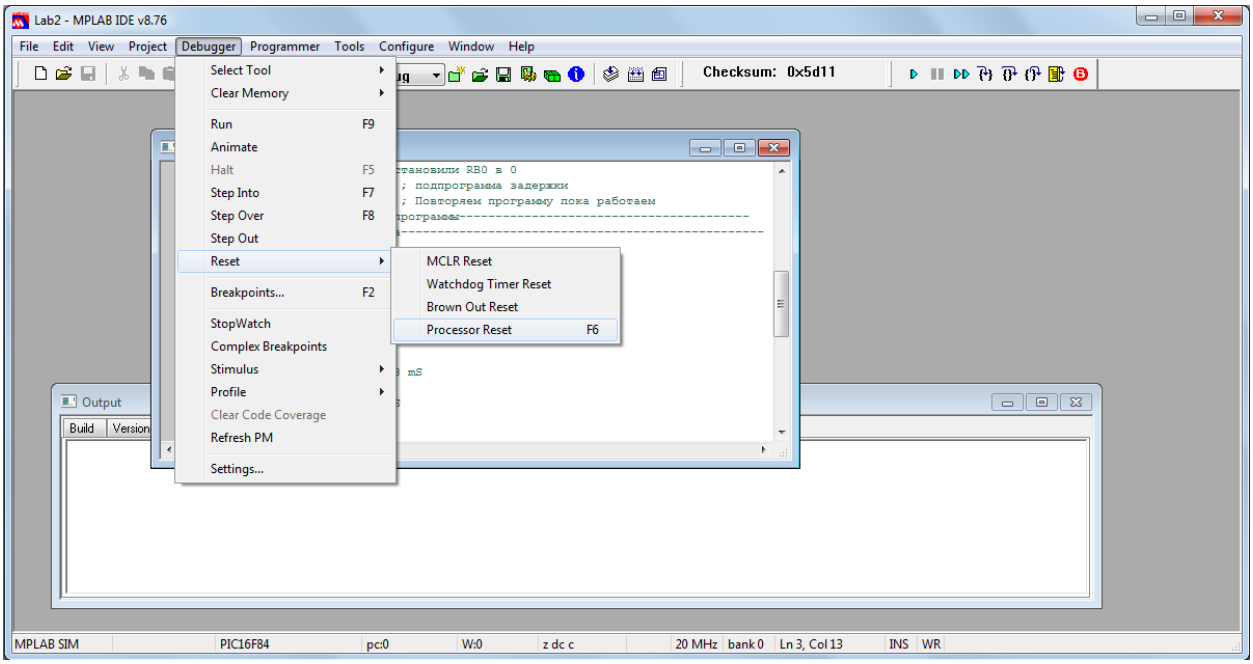


Рис. 3.7 Сброс

Эта команда переводит все регистры контроллера в исходное состояние, сбрасывает состояние счетчика команд. Пошаговое выполнение можно выполнять с помощью *F7* и *F8* или из меню *Debugger* (Рис. 3.8).

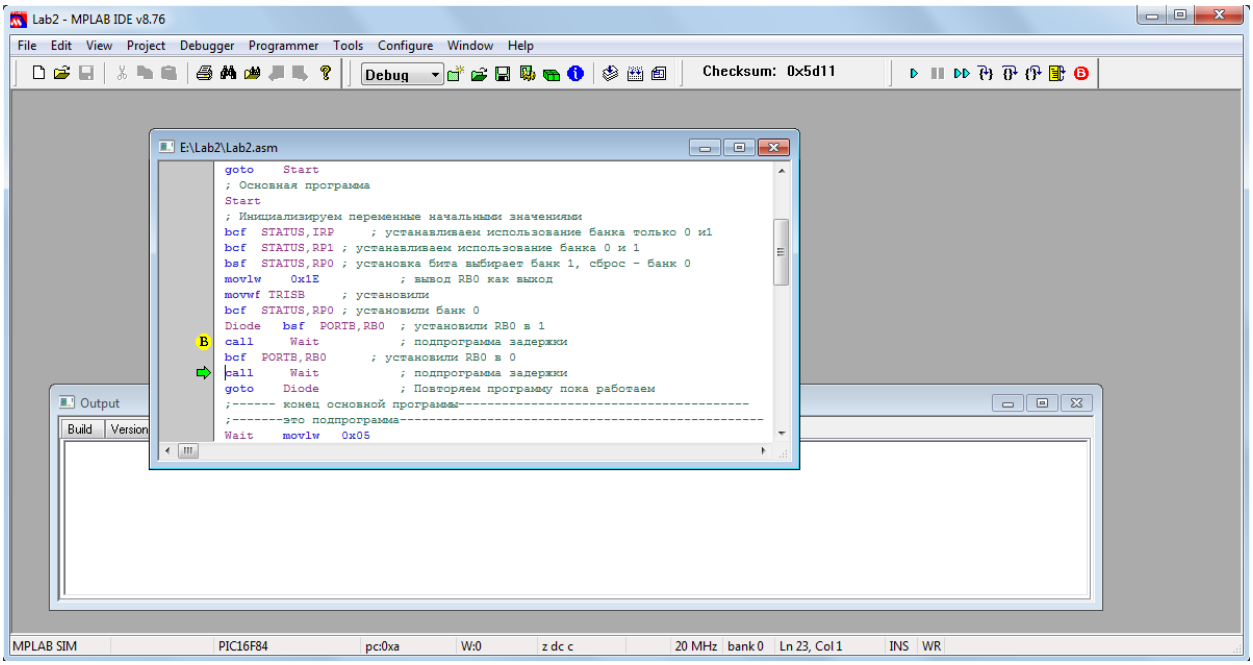


Рис. 3.8 Окно отладчика в пошаговом режиме с точкой останова

Если программа работает так, как нам нужно, можно приступать к прошивке МК с помощью адаптера для PIC16F84A. Для этого подключаем адаптер к одному из свободных USB-портов компьютера.

После подключения необходимо проверить состояние битов конфигурации (рис. 3.8.).

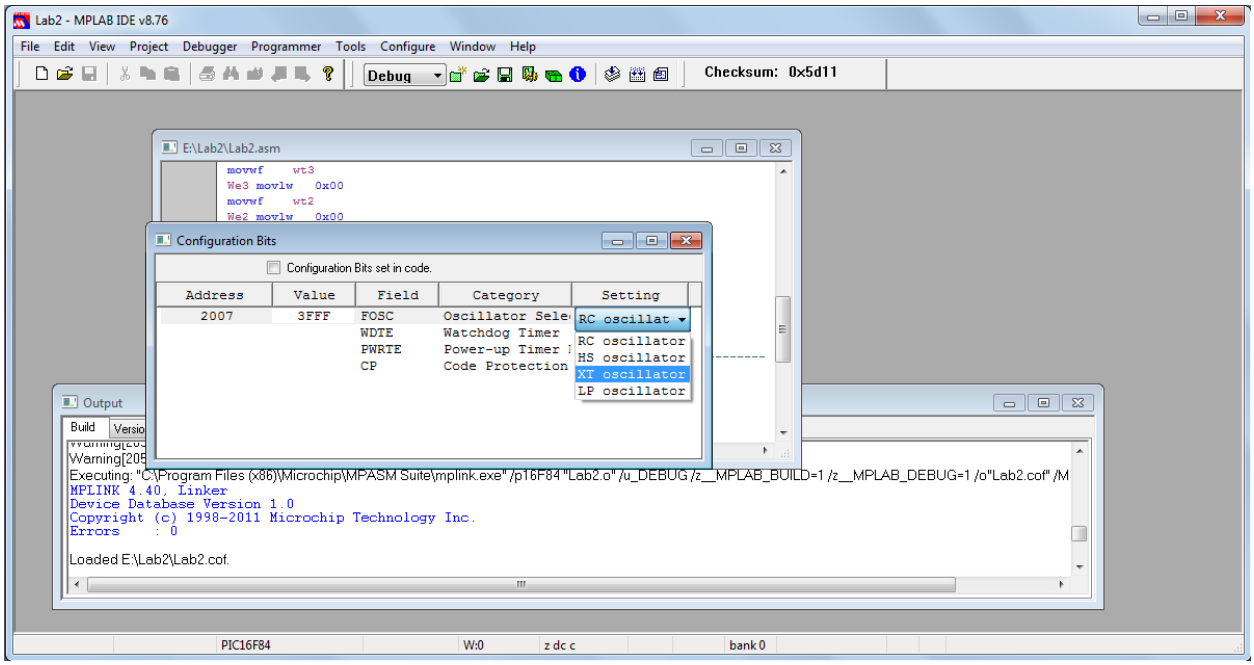


Рис. 3.8 Окно настройки конфигурации.

После выбора битов конфигурации необходимо подключить отладочную плату (Рис. 3.9).

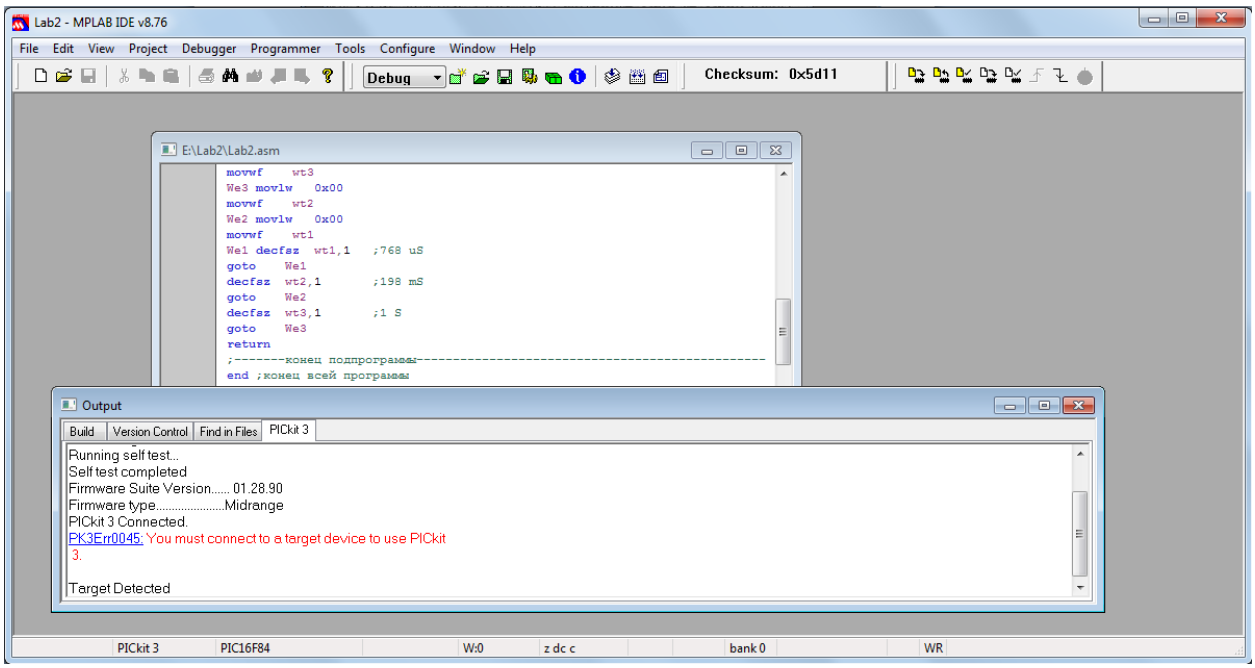


Рис. 3.9 Подключение отладочной платы.

После этого с помощью значков в строке управления необходимо произвести соответствующее действие (Program, Read, Verify). (Рис. 3.10).

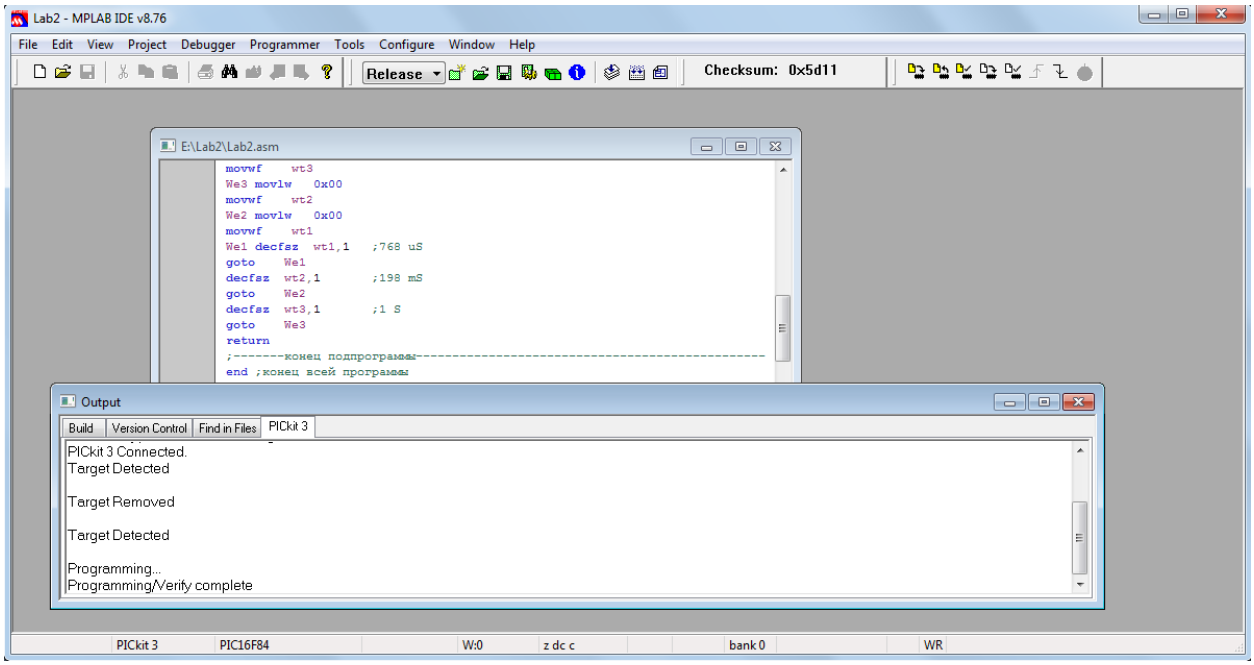


Рис. 3.10 Программирование.

После записи слова конфигурации и прошивки МК необходимо проверить работоспособность созданного устройства на лабораторном стенде. При правильной работе устройства должен мигать светодиод, контролирующий состояние линии 0 порта B. Период мигания должен составлять 2 с (светодиод светится в течение 1с. и погашен в течение 1с.).

3.2. Задания для выполнения работы

1. Написать программу сложения 8-разрядного числа, вводимого с линий порта A МК, и числа 5. Использовать Stimulus. Вывод результата осуществить в порт B.

2. Написать программу сложения, умножения, вычитания или деления (по заданию преподавателя) двух чисел. Разрядность чисел, их местонахождение в памяти и область вывода результатов получить у преподавателя.

3. Написать программу временной задержки, используя программный алгоритм, основанный на применении циклов. Время задержки и точность по заданию преподавателя.

4. Написать программу временной задержки, используя таймер/счетчик МК. Время задержки и точность по заданию преподавателя.

5. Разработать индикатор количества нажатий кнопки в двоичном коде (двоичный счетчик). (Номер кнопки и направления счета получить у преподавателя)

6. Разработать «бегущие огни» на не менее чем на 8 эффектов (изменение номера эффекта циклическое, по нажатию на указанную преподавателем кнопку) с изменением направления движения при нажатии на кнопку. О необходимости использовать при этом прерывания уточнить у преподавателя.

7. Построить пешеходный светофор. Светофор управляется кнопкой, которая подключена к одной из линий порта A. В исходном состоянии горит зеленый для автомобилей и красный для пешеходов. После нажатия кнопки зеленый мигает 10 секунд, загорается желтый + красный для автомобилей, горит 10 секунд. Затем 20 секунд для автомобилей горит красный, а для пешеходов зеленый. Для пешеходов зеленый мигает 10 секунд, для автомобилей горит желтый. Затем для автомобилей загорается зеленый, а для пешеходов красный. После выполнения данного цикла светофор 10 секунд не реагирует на кнопку.

3.3. Контрольные вопросы

1. Охарактеризуете Гарвардскую архитектуру.
2. Что означает аббревиатура RISC и CISC?
3. Расскажите об архитектуре PIC16F84A. (Шина команд и данных, адресация, регистры, АЛУ, конвейер).
4. Расскажите о портах ввода/вывода. Сколько разрядов имеет каждый? Какие дополнительные функции имеют некоторые разряды?
5. Сколько типов генераторов поддерживает PIC16F84A?
6. Расскажите о таймере/счетчике.
7. Сколько источников прерываний имеет PIC16F84A?
8. Каковы электрические характеристики PIC16F84A?
9. Что такое слово конфигурации, его расположение в памяти?
10. Какие средства необходимы для разработки микропроцессорных систем?
11. Можно ли с помощью отладчика пакета MPLAB просматривать время выполнения программы?
12. Как с помощью отладчика MPLAB запустить на выполнение только часть программы?
13. Как с помощью отладчика задать внешние воздействия? Какого типа воздействия можно задать?

Требования к защите работы:

1. Демонстрация написанных и полностью отлаженных программ на компьютере (пример и задания 5-7 – на макетной плате).

2. Отчет по лабораторной работе, оформленный в соответствии с правилами.

3. Готовность отвечать на вопросы преподавателя по теме работы.

Правила оформления отчета к работе:

Отчет, является документом, отражающим результаты и ход выполнения лабораторной работы. Отчет должен содержать следующие пункты:

1. Титульный лист, содержащий тему и номер лабораторной работы, фамилии выполнявшего студента и проверявшего преподавателя.

2. Цель работы – указываются цели, которые необходимо достигнуть в результате выполнения лабораторной работы.

3. Задачи – указываются задачи, решаемые в ходе выполнения лабораторной работы.

4. Выполнение работы – указывается последовательность действий или алгоритм работы программы, приводятся выполненные расчеты.

5. Результаты работы – приводятся алгоритм, блок-схема алгоритма работы программы и текст работы программы с комментариями на русском языке.

6. Вывод по каждому пункту, в соответствии с заданием.

6. Общий вывод по работе в соответствии с заявленными целями. – пункт содержит перечень решенных в ходе работы задач и выполненных целей, а также выводы по достигнутым результатам.

Обратить внимание:

1. Отчет принимается только в печатном виде на бумажном носителе. Т.е. отчет должен быть полностью набран на компьютере и распечатан. Отчеты в электронном виде рассматриваться не будут.

2. Листы отчета должны быть пронумерованы, скреплены между собой и помещены в папку.