ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
СРЕДНЕГО (ПОЛНОГО) ОБЩЕГО ОБРАЗОВАНИЯ
ЛИЦЕЙ ПРИ СПБГУТ

Вендор-ориентированный учебный курс в системе
«Старшая профильно-профессиональная школа-ВУЗ-Работодатель»:
«Программирование микроконтроллеров Microchip»

Богураев М.В.

«ОЗНАКОМЛЕНИЕ С ФУНКЦИЯМИ»

Методические указания к выполнению
лабораторной работы

Санкт - Петербург
2013

Богураев М.В. «ОЗНАКОМЛЕНИЕ С ФУНКЦИЯМИ». Методические указания к выполнению лабораторной работы № 3. СПб: ГОУ «Лицей при СПбГУТ», 2013.

ЛАБОРАТОРНАЯ РАБОТА №3
«ОЗНАКОМЛЕНИЕ С ФУНКЦИЯМИ»

Цель работы

Понять, что такое функции и как ими пользоваться. Научиться создавать свои функции, передавать внутрь них аргументы, и возвращать значение функции. Освоить создание программ для микроконтроллеров PIC на языке СИ с использованием компилятора HI-TECH C. Научиться симулировать работу программ на СИ для микроконтроллера в среде разработки MPLAB IDE.

Теоретические основы

Чтобы написать хорошую программу нужно сначала выяснить, что должно делать устройство и как оно должно работать. Затем составить алгоритм программы, чтобы осознать и уточнить работу устройства. Потом описать алгоритм на языке программирования. Программа – это набор текстовых файлов. В каждом таком файле при помощи операторов описывают переменные, константы и действия микроконтроллера. Операторы – это специальные символы в тексте программы, которыми обозначают операции. Операции – это «сказуемые» языка СИ. Термин операция – обозначает действие, которое нужно проделать над операндами. Операнды – это то, над чем проводятся операции. Операндами могут быть переменные или константы – это «подлежащие, приложения и дополнения» языка СИ. Переменная – это область памяти данных, которая имеет своё название, адрес, размер и определённым образом обрабатывается компилятором; программа может менять содержимое этой области памяти данных. Размер и способ обработки задаёт компилятору программист. Переменные характеризуются следующими параметрами: имя, тип, значение (текущее значение), адрес.

Имя переменной – задаётся программистом, с этим именем работает программист. Языки высокого уровня устроены так, что программист может работать с памятью компьютера посредством имён переменных, не задумываясь, по какому адресу расположены эти переменные.

Тип (или размер) переменной – этот параметр так же задаёт программист. Для компилятора тип переменной – это размер области памяти данных и то, как компилятор «понимает» содержимое этой области памяти данных. Тип – один из типов языка СИ (signed char, unsigned char, signed int и. д.).

Значение (текущее значение) – это то содержимое (комбинация единиц и нулей), что записано сейчас в области памяти данных, отведённой компилятором для переменной с заданным именем.

Адрес переменной – адрес области памяти данных, по которому расположена переменная с заданным именем. Этот адрес определяет компилятор.

Правила описания переменных следующие: описание переменной содержит тип переменной и имя переменной, можно сразу указать начальное значение переменной. Тип переменной – один из типов языка СИ (unsigned char, signed char, unsigned int, signed int, unsigned float и т. д.). Имя функции может состоять из латинских букв и цифр без пробелов. Например, так:

```
signed char a = 0;
```

Константа – это тоже область памяти данных, но, в отличие от переменной, содержимое этой области программой не меняется. В записи 3+2=5 числа 3 и 2 это операнды, 5 – это результат; символами + и = обозначают операторы сложения и вывода результата. При этом 3, 2 и 5 – это константы. В записи y=5x+8 «y» и «x» переменные, а 5 и 8 константы. При написании программы на СИ программист записывает в текстовый

файл константы, переменные и операторы с операндами, в соответствии с правилами языка и алгоритмом программы. При этом операнды и операторы помещают внутрь функций. Функции – удобный способ свести в одно место вычислительные операции, а затем обращаться к этим операциям по много раз. Получается, что функция в СИ – это группа команд, объединённых одним именем – названием функции, то есть функция – это способ сокращения записи последовательностей операторов и операндов. Функции могут быть написаны одними, а использоваться другими программистами это является большим достоинством языка СИ – это ускоряет и упрощает программирование. Правила написания функции следующие: описание функции содержит заголовок, круглые скобки и фигурные скобки. В заголовке описывают тип возвращаемого значения и имя функции. Возвращаемое значение – это результат выполнения функции. В круглых скобках объявляют аргументы (или параметры), то есть то, что попадёт внутрь функции. В фигурных скобках пишут исполняемые операторы, которые называют телом функции. Перед написанием функции необходимо учесть, что все переменные и константы должны быть объявлены до первого использования. В наиболее общем виде функции описывают так:

<тип возвращаемого значения> <имя функции> (<аргументы>) {<тело функции>}

Например:

unsigned int myfunction (unsigned char x, signed int y) {…};

где unsigned int – тип возвращаемого значения, myfunction – имя функции: эти две части составляют заголовок функции; в круглых скобках указаны unsigned char x, signed int y – аргументы функции, в фигурных скобках вместо многоточия указывают операторы и операнды. Для аргументов компилятор выделяет место в памяти данных и создаёт локальные переменные – то есть переменные доступные только внутри самой функции. При выполнении функции значения из внешних, по отношению к функции, переменных копируются в локальные переменные внутрь функции. Поэтому тип аргумента (signed char, unsigned char, signed int и т. д.) должен совпадать с типом копируемой переменной, чтобы внутри функции правильно разместить данные. После выполнения функции локальные переменные исчезают, скопированные значения аргументов теряются и этот участок памяти данных может быть использован другой функцией под свои переменные. Значение, которое возвратит функция, задают оператором return <значение>. Если функция должна возвращать какое-либо значение из множества значений, то создают локальную переменную внутри функции, затем присваивают этой переменной значение и возвращают эту переменную (в нашей программе это переменная key). Функция может не возвращать никакого значения. Такие функции иногда называют подпрограммами. Во всех функциях, которые ничего не возвращают, указывается ключевое слово void, которое обозначает, что функция не возвращает значения. Функция void main (void){} запускается первой в начале программы. Программа для микроконтроллера – это, в принципе, бесконечный цикл. Функция void main (void){} не может вернуть никакого значения, потому что некуда возвращать. То есть, нет функции, которая вызывает main( ). Так же при написании функций важно учесть, следующее: если функция используется внутри функции main(){}, то эта функция должна быть описана до main(){}. Структура программ на языке СИ следующая:

Заголовки функций (директива #include).

Описание своих типов.

Описание глобальных переменных – единых переменных для всех функций.

Описание своих функций.

Функция void main (void){}.

Каждая программа начинается с директивы #include < >. Этот фрагмент при компиляции заменяется на текст включаемого файла, название которого заключено в скобки. Если скобки треугольные (#include < >), то это указывает компилятору путь к стандартным библиотекам в папке include – чаще всего на диске С. Или в том месте, куда

был установлен компилятор. Стандартные библиотеки содержат часто используемые функции и устанавливаются вместе с компилятором. Если программист ставит кавычки (#include " "), то это указывает компилятору на файлы в текущем каталоге проекта. Часто ещё можно встретить конструкцию #define. Например, #define N 6000 приведёт к замене N на 6000 (как поиск и замена в текстовом файле).

Например:

```
#include <stdio.h>
#define text "Hello, World!"
void main (void)
   {
      printf (text);
   }
```

Текст программы обязательно должен содержать описание функции void main (void){}. Если программа предназначена для микроконтроллеров PIC и будет обработана компилятором HI-TECH C, то обязательно подключается библиотечный файл htc.h, и задаётся слово конфигурации. Слово конфигурации настраивает микроконтроллер и задаётся директивой __CONFIG(). Если компилятор был установлен на диск по умолчанию, тогда подключаемые файлы доступны по адресу C:\Program Files\HI-TECH Software\PICC\LITE\9.xx\include. Следует помнить, что компилятор СИ проверяет только правильность синтаксиса программы. Компилятор не определяет логических ошибок программы, поэтому синтаксически верная программа может не работать или работать неправильно из-за нарушения логики. Поэтому перед написанием любой программы нужно изобразить алгоритм работы микроконтроллера. Ответственность за логику работы программы лежит на программисте.

Задание

На первом этапе откомпилируйте проект с файлом Project3_1.c, сначала проделайте работу с исходным текстом программы, выполните пошагово программу, при этом наблюдайте за изменением переменных b и a в окне Watch. Проанализируйте содержимое переменных a, b. Затем измените исходный текст программы: поменяйте значение переменной b на другое, положительное. Откомпилируйте программу. Выполните пошагово программу, при этом наблюдайте за изменением переменной b и переменной a в окне Watch. Проанализируйте содержимое переменных a, b. На втором этапе разберитесь, как функция возвращает одно из четырёх возможных значений. Проанализируйте значение переменной key в окне Watch.

Порядок выполнения

Первый этап. На диске C в папке Projects создайте папку Project3. В эту папку скопируйте файлы Project3_1.c и Project3_2.c. Запустите программу MPLAB IDE и создайте проект на СИ в этой папке (на диске C по адресу – C:\Projects\Project3). Присоедините к проекту файл Project3_1.c (рис. 1).

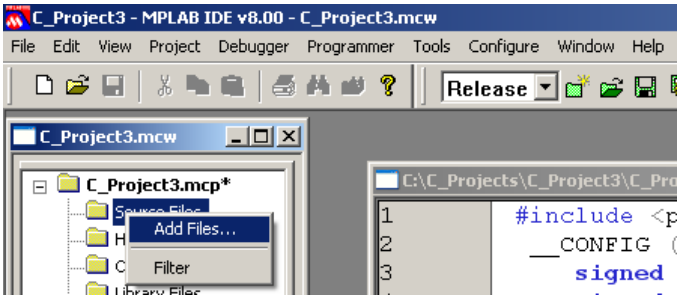


Рис. 1. Добавление файла в проект.

Откомпилируйте проект, нажав на красную кнопку (рис. 2).

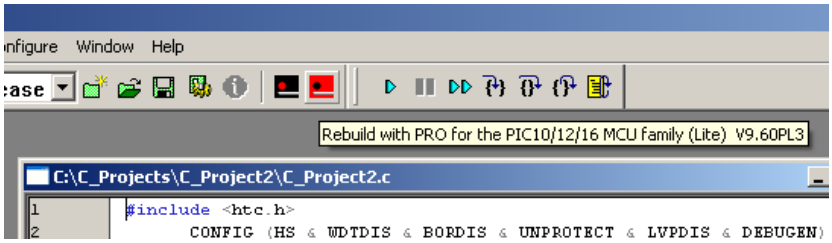


Рис. 2. Компиляция проекта.

В качестве отладчика выберите симулятор MPLAB SIM. Для этого в главном меню выберите Debugger/Select Tool/MPLAB SIM (рис. 3).

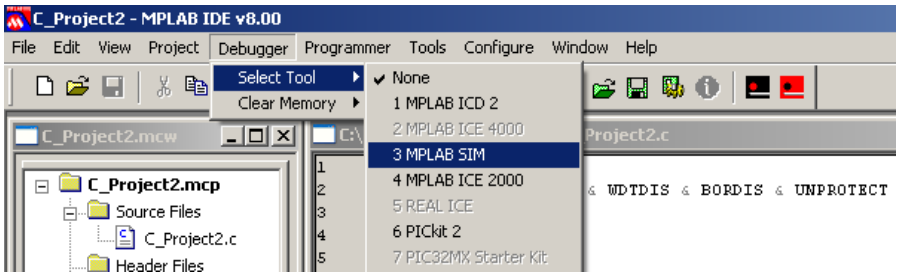


Рис. 3. Выбор симулятора MPLAB SIM.

Откройте окно Watch. Для этого в главном меню выберите View/Watch (рис. 4).

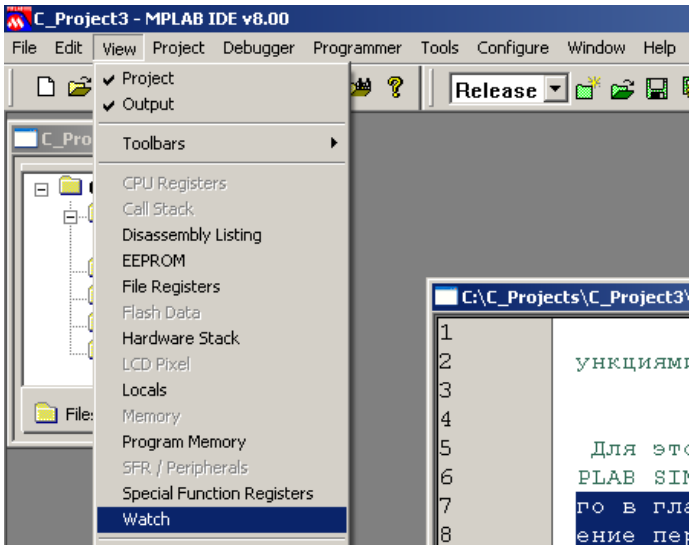


Рис. 4. Открытие окна Watch.

В окне Watch откройте переменные a и b. Для этого нажмите чёрный треугольник напротив Add Symbol, выберите нужную переменную и нажмите Add Symbol (рис. 5).

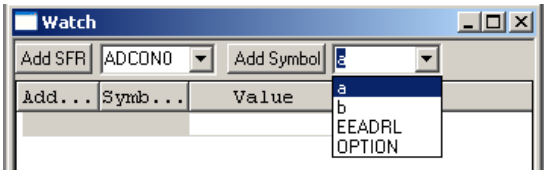


Рис. 5. Выбор переменной для наблюдения.

Теперь нужно выполнять пошагово программу и наблюдать в окне Watch изменение содержимого переменных a и b. Для этого нажимайте кнопку "стрелка в фигурных скобках" (рис. 6) или клавишу F7 на клавиатуре.

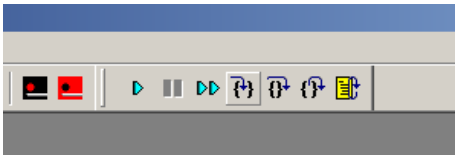


Рис. 6. Кнопка пошагового выполнения программы.

В программе Project3_1.c функция abs при объявлении имеет абстрактный аргумент x. Когда функция abs вызывается из функции main, тогда в функцию abs вместо абстрактного x подставляется конкретное значение из переменной, которую программист укажет в круглых скобках. В нашем случае это переменная b. Далее внутри функции main все действия, которые происходят для переменной x функции abs, выполняются со значением, скопированным из переменной b. При таком вызове функции говорят о передаче параметра по значению. Т.е. передаётся только значение переменной b, это значение подставляется вместо x. Тип переменной b это signed char. Предполагается, что функция abs будет работать с переменной b, поэтому аргумент функции abs (в круглых скобках) объявлен как signed char. Предполагается, что значение из функции abs возвращается в вызывающую её функцию main, при этом возвращённое значение присваивается переменной a. Переменная a имеет тип signed char, значит, возвращаемое значение тоже должно быть signed char. Поэтому возвращаемое значение функции объявлено как signed char.

Поменяйте значение переменной b на другое, положительное. Откомпилируйте программу. Выполните пошагово программу, при этом наблюдайте за изменением переменной b и переменной a в окне Watch.

Второй этап. К вашему проекту, расположенному на диске C по адресу – C:\Projects\Project3, присоедините файл Project3_2.c. Для этого отсоедините от проекта файл Project3_1.c. Чтобы отсоединить файл нажмите правой кнопкой мыши на название файла в окошке C_Project3.mcw, в выпадающем списке выберите Remove (рис. 7)

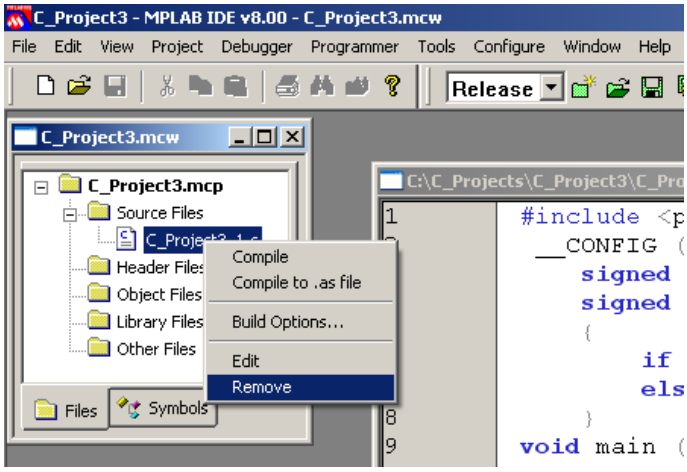


Рис. 7. Отсоединение файла Project3_1.c

Теперь добавьте к проекту файл Project3_2.c для чего нажмите правой кнопкой мыши на папку с названием Source Files в окошке Project3.mcw. Затем в выпадающем списке выберите Add Files…(рис. 1) нажмите левую кнопку мыши и в папке C:\Projects\Project3 выберите файл Project3_2.c. Этот файл вы должны были поместить

туда в начале работы. Откомпилируйте проект. В окне Watch нужно добавить представление переменных Char. Для этого в окне Watch наведите курсор на надпись Address, щёлкните правой кнопкой мыши, в появившемся списке поставьте галочку напротив Char. В окне Watch откройте переменные a и b. Для этого нажмите чёрный треугольник напротив Add Symbol, выберите нужную переменную, нажмите Add Symbol (рис. 5).

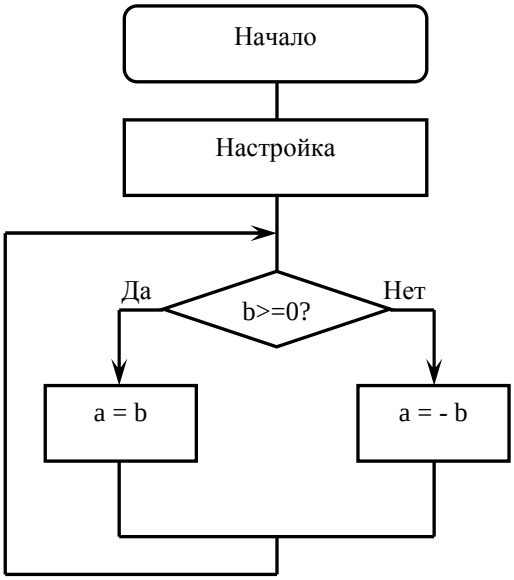
Выполняйте пошагово программу и наблюдайте в окне Watch изменение содержимого переменных a, b и myfunction_key. Для этого нажимайте кнопку "стрелка в фигурных скобках" или клавишу F7 на клавиатуре (рис. 6). В этой программе функция возвращает одно значение из множества возможных. Возвращаемое значение определяется аргументом функции. Для такого возврата создаём локальную переменную внутри функции, затем присваиваем этой переменной значение и возвращаем эту переменную. В программе Project3_2 создаём и возвращаем переменную key. Выполните пошагово программу для других значений переменной a. Две косые черты – оператор комментирования строки. Для разных значений переменной a раскомментируйте одно значение (удалите две косые черты), остальные значения закомментируйте. Откомпилируйте программу.

Проделайте программу пошагово для четырёх значений переменной a, проанализируйте содержимое переменных a, b и myfunction_key.

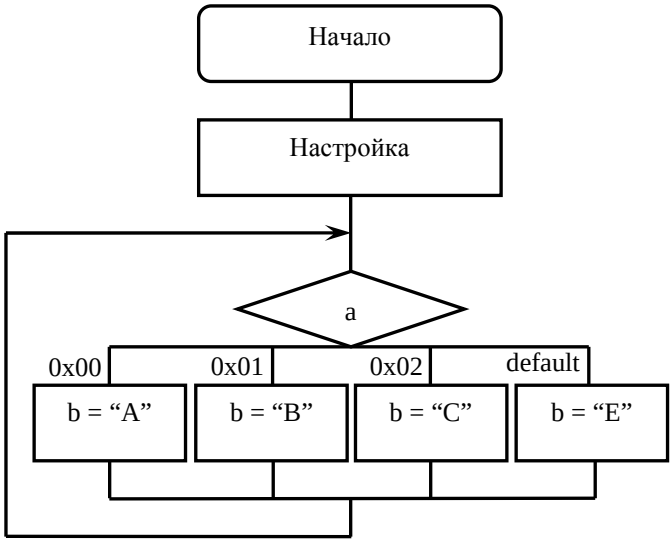
Аппаратное обеспечение

Эта работа выполняется в симуляторе среды разработки MPLAB IDE, аппаратное обеспечение не требуется.

Программное обеспечение



Алгоритм программы C_Project3_1



Алгоритм программы C_Project3_2

Текст файла Project3_1.c

```
#include <pic.h>
__CONFIG (HS & WDTDIS & LVPDIS & DEBUGEN);
   signed char    a,b = 0;
signed char abs (signed char x)
   {
      if (x>=0) return x;
      else return -x;
   }
void main (void)
{
   while (1)
      {
         b = - 5;
         a = 123;
         a = abs (b);
      }
}
```

Текст файла Project3_2.c

```
 #include <pic.h>
 __CONFIG (HS & WDTDIS & LVPDIS & DEBUGEN);
unsigned char a,b = 0;
unsigned char myfunction (unsigned char x)
{
   unsigned char key = 0;
   switch (x)
   {
   case 0x00: key = 'A'; break;
   case 0x01: key = 'B'; break;
   case 0x02: key = 'C'; break;
   default: key = 'E';
   }
   return key;
}
void main (void)
{
   while (1)
      {
         //a = 0x00;
         a = 0x01;
         //a = 0x02;
         //a = 156;
         b = 0;
         b = myfunction (a);
      }
}
```

Индивидуальные задания

Первый этап. Почему переменная b отображается в окне Watch как 0xFB но при этом имеет значение – 5? Если будет нужно, чтобы переменные a и b были типа signed int, что тогда нужно будет изменить в программе?

Второй этап. Просмотрите локальную переменную myfunction_key. Объясните поведение переменной в окне Watch при выходе из функции myfunction (a).

Контрольные вопросы

1. Для чего составляется алгоритм программы?
2. Что такое программа?
3. Что такое операторы и операции?
4. Что такое операнды?
5. Что такое переменные?
6. Какими параметрами можно охарактеризовать переменную?
7. Каковы правила описания переменных?
8. Что такое константы?
9. Что такое функции?
10. Каковы правила написания функций?
11. Что такое возвращаемое значение?
12. Что такое аргументы (или параметры) функции?
13. Что такое тело функции?
14. Напишите функцию в общем виде.
15. Каким оператором задают возвращаемое значение?
16. Может ли функция не возвращать значения?
17. Что обозначает ключевое слово void?
18. Что происходит при компиляции, если встречается директива #include?
19. Что значат для компилятора треугольные скобки в директиве #include?
20. Что значат для компилятора кавычки в директиве #include?
21. Что значит для компилятора директива #define?

Оглавление:

Список рисунков: