Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

Э. М. Виноградов

ПРОГРАММИРОВАНИЕ PIC-КОНТРОЛЛЕРОВ НА ЯЗЫКЕ СИ

ПРАКТИКУМ
по выполнению лабораторных работ
по дисциплине «Микропроцессоры в системах управления» для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения

Электронный аналог печатного издания

Гомель 2018

УДК 681(075.8)
ББК 32.81я73
В49

Рекомендовано к изданию научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 5 от 04.12.2017 г.*)

Рецензент: доц. каф. «Автоматизированный электропривод» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *В. В. Тодарев*

Виноградов, Э. М.

В49 Программирование PIC-контроллеров на языке Си : практикум по выполнению лаборатор. работ по дисциплине «Микропроцессоры в системах управления» для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» днев. формы обучения / Э. М. Виноградов. – Гомель : ГГТУ им. П. О. Сухого, 2018. – 56 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

ISBN 978-985-535-383-7.

Представлены четыре лабораторные работы с порядком их выполнения, основными теоретическими сведениями, заданиями для самостоятельной работы и контрольными вопросами.

Для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения.

УДК 681(075.8)
ББК 32.81я73

ISBN 978-985-535-383-7

Лабораторная работа № 1

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ ДЛЯ PIC-МИКРОКОНТРОЛЛЕРОВ

1 Цель работы

Изучить и практически исследовать методы разработки и отладки программ на языке Си для PIC-микроконтроллеров с помощью интегрированной среды.

2 Основные теоретические сведения

Для создания программного обеспечения микроконтроллерных систем широко используются средства вычислительной техники, в том числе персональные компьютеры, позволяющие разработчику выполнить весь цикл проектирования, включая отладку целевой программы. В настоящее время самым мощным средством разработки программного обеспечения для микроконтроллеров являются интегрированные среды разработки IDE (Integrated Development Environment).

Одним из таких программных инструментов является MPLAB IDE – интегрированная среда разработки для микроконтроллеров PICmicro компании Microchip. MPLAB IDE содержит все инструментальные средства, необходимые для того, чтобы написать программу на языке Ассемблер, получить ее машинные коды, выполнить на симуляторе ее тестирование и загрузить машинные коды в программатор. IDE имеет встроенные и сменные модули, позволяющие сконфигурировать среду с различными инструментальными средствами, программным обеспечением и аппаратными средствами. Для создания программ на языке Си в MPLAB IDE могут быть добавлены компиляторы C18 и C30 разработки компании Microchip. Могут также использоваться компиляторы фирм HI-TECH, IAR, CSS, Byte Craft. Пакет программ MPLAB IDE можно бесплатно загрузить с сайта www.microchip.com. С сайта www.microchip.ru можно также бесплатно загрузить демо-версии некоторых компиляторов.

Сербская компания MikroElektronika разработала IDE, которая позволяет быстро создавать эффективные программы на языке Си. Среда имеет удобный интерфейс пользователя со встроенным редактором текста и мощным отладчиком программ. Встроенный мастер проектов позволяет в считанные минуты создать заготовку рабочей программы для любого микроконтроллера из целого семейства. Биб-

лиотека готовых функций обеспечивает программиста поддержкой для быстрого и безошибочного создания программы. Компания MikroElektronika создала среду разработки для таких популярных и известных микроконтроллеров, как семейство PIC компании Microchip, AVR компании Atmel и семейство MCS-51. С сайта www.mikroe.com компании MikroElektronika можно бесплатно скачать демонстрационную версию среды для PIC-контроллеров, которая позволяет создавать программы с объемом исполняемого кода до 2 Кбайт.

Разработка программного обеспечения при использовании среды разработки IDE состоит из следующих основных этапов:

1 Создание проекта.

2 Создание исходных файлов на языке Си.

3 Построение проекта.

4 Тестирование программы и ее отладка.

Среда IDE организует программное обеспечение в виде проектов, состоящих из одного файла проекта (файл с расширением *.mcppi) и одного или нескольких исходных файлов на языке Си (файлов с расширением *.c). Исходные файлы могут компилироваться только в том случае, если они включены в проект.

Файл проекта содержит:

– имя проекта;

– тип микроконтроллера;

– тактовую частоту его работы;

– слово конфигурации микроконтроллера;

– список исходных файлов для проекта;

– другие (вспомогательные) файлы.

3 Порядок выполнения работы

3.1 Создание папки для работы в среде IDE

3.1.1 Для выполнения лабораторных работ по дисциплине «Микропроцессоры в системах управления» Вам необходимо создать на компьютере свою рабочую папку. Сделать это можно следующим образом.

На панели TotCom выберите диск F. Затем выберите папку MPinCS (сокращение от Microprocessors in Control Systems) и раскройте ее. Внутри нее создайте новую папку (с помощью клавиши F7) с именем, соответствующим Вашей фамилии (буквы обязательно латинские), например: Ivanov.

Примечание. При использовании в именах папок русских букв возможна неправильная работа исследуемых программ.

В дальнейшем Вы будете записывать и хранить в Вашей папке все файлы в процессе выполнения лабораторных работ. Полный путь к ней:

F:\MPinCS\Ivanov

3.1.2 При выполнении лабораторных работ Вы будете создавать много различных файлов. Чтобы было легче ориентироваться в них, желательно для каждой лабораторной работы иметь свою отдельную папку. Допустим, что для выполнения лабораторной работы № 1 мы будем использовать папку с именем Lab1, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку (с именем, соответствующим Вашей фамилии) и создайте в ней (с помощью клавиши F7) новую папку с именем Lab1.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы в процессе выполнения лабораторной работы № 1. Полный путь к ней:

F:\MPinCS\Ivanov\Lab1

3.2 Запуск среды разработки

Интегрированная среда разработки IDE запускается из стартового меню Windows подобно остальным приложениям, либо, что удобнее, с помощью ярлычка на рабочем столе компьютера.

После запуска программы на экране компьютера Вы увидите рабочий стол среды IDE. В верхней части экрана находится строка заголовка, в которой выводится имя проекта. Под ней расположена строка главного меню. Главное меню содержит обширный набор команд для доступа к функциям среды.

Если по умолчанию установлена опция Show Page On Startup, которая открывает Start Page (стартовую страницу), то на рабочем столе среды разработки будут открыты следующие окна.

В центре экрана располагается окно Get Started с различными сервисными функциями для помощи начинающим пользователям. В левой части экрана расположены команды для создания нового проекта (**New Project**…), открытия существующего проекта (**Open Project…)** и открытия папки примеров (**Open Examples Folder…**). В нижней части экрана находится окно сообщений. Стартовую страницу можно открыть также с помощью команды меню **View > Start Page.**

Рассмотрим методику разработки программного обеспечения в интегрированной среде на примерах простейших программ для микроконтроллера PIC16F84A.

3.3 Первая программа вывода данных в порт МК

3.3.1 *Создание нового проекта*. Процесс создания нового проекта в среде IDE очень прост. Подведите курсор мыши к строке команды для создания нового проекта **New Project …** и щелкните левой кнопкой.

На экране появится окно мастера создания нового проекта – New Project Wizard. Начальное окно содержит список действий, которые должны быть выполнены для создания нового проекта. Этот процесс разбит на несколько шагов (Steps), которые Вы должны последовательно проделать.

Для продолжения щелкните по кнопке [Next].

Шаг 1 Установка настроек проекта

На этом шаге необходимо определить общую информацию для проекта: выбрать микроконтроллер, его тактовую частоту и, конечно, дать проекту название. Это очень важный этап, так как компилятор создает внутренние настройки на основе этой информации. По умолчанию мастер проекта предлагает определенные настройки, но в нашем случае их требуется изменить. С этой целью проделайте следующее.

1 Очистите строку Project Name и введите новое имя проекта. При выборе имени проекта желательно, чтобы оно имело какую-то смысловую нагрузку. В нашей первой программе будет выполняться вывод данных в порт МК, поэтому можно дать имя этому проекту как out (вывод). Напоминаем, что среда использует только латинские буквы. Использование кириллицы **недопустимо!**

Итак, введите имя проекта – out (расширение .mcppi набирать не нужно!). В строке Project Name должно быть:

Project Name: out

2 Для выбора папки хранения проекта щелкните по кнопке [Browse] – обзор. Откроется окно обзора папок. Последовательно выберите и раскройте нужные папки в соответствие с установленном в п. 3.1.2 пути f:\MPinCS\Ivanov\Lab1. Последней должна быть папка с именем Lab1. Щелкните по ее имени для раскрытия. В заключение

щелкните по кнопке [OK] для подтверждения своего выбора и установке нового пути. В строке Project Folder должно появиться имя папки Lab1 и полный путь к ней:

Project Folder: f:\MPinCS\Ivanov\Lab1\

3 Введите новое имя для микроконтроллера – PIC16F84A. С этой целью щелкните по стрелке в правой части строки Device Name. В раскрывшемся списке PIC-микроконтроллеров выберите PIC16F84A и щелкните по этой строке. В строке Device Name должно появиться:

Device Name: PIC16F84A

4 И, наконец, необходимо установить в строке Device Clock тактовую частоту работы микроконтроллера. По умолчанию она равна 8 МГц. Для нашего первого проекта можно оставить это значение.

5 В заключение шага 1 щелкните по кнопке [Next] для продолжения.

Шаг 2 Добавление файлов

Этот шаг позволяет Вам включить дополнительные файлы, которые будут необходимы для Вашего проекта, например, некоторые заголовочные файлы. В нашем случае нет необходимости подключать какие-либо дополнительные файлы. Поэтому просто щелкните по кнопке [Next] для продолжения.

Шаг 3 Подключение библиотек

Этот шаг позволяет Вам определить, желаете ли Вы включить все библиотеки в Ваш проект или нет. Следует знать, что, даже если все библиотеки включены, они не требуют какой-либо дополнительной памяти, если они не используются в Вашей программе. Главное достоинство опции подключения всех библиотек в том, что Вы получаете возможность использования в Вашей программе свыше 1000 готовых функций. Поэтому следует оставить эту конфигурацию по умолчанию. Однако убедитесь, что флажок на кнопке [Include All (Default)] установлен.

Щелкните по кнопке [Next] для продолжения.

Шаг 4 Окончание построения проекта

После того как все необходимые настройки для проекта выполнены, заключительный шаг позволяет Вам сделать еще кое-что.

Имеется возможность разрешить автоматическое открытие окна редактирования проекта после завершения работы мастера проекта. С помощью этого окна можно изменять биты конфигурации микроконтроллера. Это окно также может быть открыто в любой момент разработки программы с помощью команды меню **Edit > Project**.

Следует отметить, что в результате предыдущих шагов мастером проекта были установлены для микроконтроллера PIC16F84A следующие биты конфигурации:

– HS – высокочастотный генератор (тактовая частота от 4 до 20 МГц);

– WDT off – сторожевой таймер WDT отключен;

– Code Protection off – отключена защита кода от чтения, т. е. можно читать и записывать код в микроконтроллер.

Для выполнения лабораторных работ можно оставить эту конфигурацию МК по умолчанию. Поэтому для окончания построения проекта просто щелкните по кнопке [Finish].

Результатом работы мастера проектов будет новый проект out.mcppi, имя которого вместе с указанием полного пути к нему появится в строке заголовка в верхней части рабочего стола среды.

В центре рабочего стола откроется окно редактора кода с пустым исходным файлом для текста программы на языке Си:

```
void  main( ) {


}
```

Исходный файл по умолчанию имеет то же имя, что и проект, т. е. out.c. Следует отметить, что среда не требует, чтобы исходный файл имел такое же название, как проект, поэтому его можно переименовать.

3.3.2 *Создание исходного файла*. Наберите в окне редактора кода текст программы вывода данных в порт B микроконтроллера. Текст комментария для экономии времени можно не набирать.

Примечание. При наборе текста программы обязательно делайте отступы (3–4 пробела), как это выполнено в примере. Отступы облегчают чтение программы.

```
/*************************************************
out.c – программа вывода данных в порт B
*************************************************/
```

```
void  main( )
{
    TRISB = 0;              // настроить все линии порта B на вывод
    PORTB = 0x00;           // вывод данных (все нули) в порт B
    PORTB = 0xFF;           // вывод данных (все единицы) в порт B
}
```

После набора сохраните файл out.c, выполнив команду меню **File > Save**.

3.3.4 *Построение проекта*. Построение проекта в mikroC PRO – это процесс компиляции исходных файлов, их компоновки и создание hex-файла, предназначенного для загрузки в программную память микроконтроллера. Для построения проекта надо выполнить команду меню **Build > Build**. После процесса компиляции и компоновки в окне Messages (в нижней части экрана) выдается сообщение об итогах построения. Сообщение содержит информацию о количестве затраченных ресурсов памяти, наличии ошибок (Errors) и предупреждений (Warnings).

Возникающие ошибки (обычно синтаксические) не позволяют компилятору сгенерировать машинные коды программы на основе исходного текста, поэтому такие ошибки обязательно должны быть исправлены. При наличии ошибок выводится сообщение “Finished (with errors)”.

Предупреждения выдаются компилятором в том случае, когда он может сгенерировать машинные коды, но в процессе их формирования возникли обстоятельства, требующие внимания разработчика. К таким обстоятельствам относятся, например, использование переменной, которой не присвоено начальное значение, потеря точности при преобразовании типов переменных и т. п. Как правило, предупреждения выдаются в потенциально опасных ситуациях и позволяют избежать трудно выявляемых логических ошибок в процессе работы программы, вследствие этого игнорировать их не следует.

Если в результате построения проекта возникли сообщения об ошибках или предупреждения, то их перечень с указанием номера строки и типа ошибки отобразится (красным цветом) в окне Messages. Место ошибки в исходном тексте можно легко найти, дважды щелкнув мышью на строке соответствующего сообщения.

Если ошибок нет, то в окне Messages появится сообщение (голубого цвета) с текстом “Finished successfully”.

После успешной компиляции среда генерирует выходные файлы, которые помещаются в папку с файлом проекта:

– ассемблерный файл (с расширением *.asm). Содержит команды Ассемблера, сгенерированные компилятором;

– файл листинга (с расширением *.lst). Это общая картина распределения памяти микроконтроллера: адреса команд, регистры, программы и метки;

– HEX-файл (с расширением *.hex). Это основной выходной файл, который используется для загрузки кодов программы в память микроконтроллера.

Выходные файлы по умолчанию имеют то же имя, что и файл проекта. Ассемблерный файл и листинг могут быть просмотрены с использованием команд меню **View**.

Задание. Выполните построение проекта out.mcppi с помощью команды меню **Build > Build**. В случае успешной компиляции выведите на экран ассемблерный файл out.asm, используя команду меню **View > Assembly**. В этом файле в правой части находятся строки исходной программы на языке Си. В средней части размещены команды Ассемблера, сформированные компилятором. Обратите внимание, что при окончании функции main( ) – (правая фигурная скобка }) компилятор сгенерировал команду GOTO $. Эта команда формирует бесконечный цикл повторения самой себя, выход из которого возможен только путем сброса микроконтроллера. Таким образом, после окончания главной функции main( ) дальнейшее выполнение программы микроконтроллером фактически прекращается (программа зацикливается).

Для закрытия окна ассемблерного файла надо щелкнуть мышью по значку “x“ в правой части строки out.asm.

3.3.5 *Тестирование программы с помощью симулятора-отладчика*. Среда IDE имеет средства отладки, позволяющие проверить работоспособность программы и исправить при необходимости ошибки в исходном тексте. Отладку можно выполнять как с помощью внутреннего симулятора работы микроконтроллера, так и с помощью аппаратного отладчика, подключив к нему целевую микроконтроллерную систему. По умолчанию отладчик IDE настроен для работы в режиме симулятора.

Для запуска отладчика надо выполнить команду из меню **Run > Start Debugger**. В результате IDE перейдет в режим отладки (в нем текущая строка исходного кода обозначается зеленой стрелкой

и по умолчанию выделена синим цветом), и откроется окно наблюдения Watch Values, позволяющее отслеживать в ходе выполнения программы содержимое различных переменных и регистров. Значения обновляются в процессе симуляции работы программы в отладчике. Последние измененные элементы в окне наблюдения выделяются красным цветом.

Примечание. Если после запуска отладчика на экране будет отсутствовать окно наблюдения Watch Values, то его можно открыть с помощью команды меню **View > Debug Window > Watch Window** или нажатием комбинации клавиш shift + F5. Формат отображения данных в окне наблюдения будет по умолчанию десятичный.

Для внесения в список окна наблюдения Watch Values той или иной переменной программы, значение которой необходимо отслеживать, ее следует выбрать в раскрывающемся списке строки Select variable from list и нажать кнопку [Add].

После того как переменная добавлена в список окна Watch Values, можно настроить параметры отображения ее в столбце Value. Для этого можно воспользоваться одним из способов:

– дважды щелкнуть мышью на соответствующей строке списка Watch Values в столбце Name или Address;

– щелкнуть в столбце Value и затем нажать расположенную справа кнопку […].

В любом случае на экране появится диалоговое окно Edit Value, позволяющее отредактировать значение переменной прямо в ходе выполнения программы или же изменить формат ее отображения.

Для выбора формата отображения необходимо установить флажок в соответствующем окошке:

– Dec – десятичный;

– Hex – шестнадцатеричный;

– Bin – двоичный;

– Float – с плавающей точкой;

– Char – символьный.

– Установка флажка Signed означает активизацию отображения знака.

Для того чтобы подтвердить изменения, внесенные в значение или формат отображения переменной, в диалоговом окне Edit Value следует нажать кнопку [OK]. Нажатие кнопки [Cancel] отменяет все внесенные изменения.

Для удаления текущего элемента в списке Watch Values служит кнопка [Remove], а для полной очистки этого списка – кнопка [Remove All].

В режиме отладки открывается также окно хронометража Watch Clock. В нем отображается текущий счетчик Current Count командных (машинных) циклов и времени в микросекундах (us) от момента запуска отладчика. Секундомер (Stopwatch) измеряет время исполнения в количестве командных циклов и микросекундах от момента запуска отладчика и может быть обнулен в любой момент времени. Разность (Delta) представляет фактическое время выполнения участка программы от предыдущей точки останова до текущей, отображаемое в количестве командных циклов и микросекундах (при пошаговом исполнении отображается время выполнения одной строки кода программы на языке Си).

Также в окне Watch Clock отображается текущая тактовая частота микроконтроллера (Clock). Тактовую частоту в окне хронометража можно изменять, что приведет к пересчету времени в микросекундах. Однако это изменение не влияет на текущие установки проекта, где также задана тактовая частота микроконтроллера, а влияет только на расчет времени симуляции.

Управлять процессом отладки можно тремя способами:

1) с помощью пунктов меню **Run**;

2) при помощи «горячих клавиш» клавиатуры;

3) с использованием кнопок (значков) в окне Watch Values.

Практика показывает, что наиболее удобно для отладки использовать именно кнопки окна Watch Values.

Примечание. Чтобы узнать функцию кнопки (значка), надо поместить курсор мыши на эту кнопку. На экране появится описание этой функции.

Название и функция основных кнопок управления отладкой (а также соответствующих клавиш клавиатуры) следующие:

– Start Debugger (F9) – запуск отладчика.

– Run/Pause Debugger (F6) – запуск/останов программы в непрерывном (автоматическом) режиме.

– Stop Debugger (Ctrl + F2) – прекращение работы отладчика.

– Step Into (F7) – шаг на одну строку программы.

– Step Over (F8) – шаг через одну строку программы.

– Step Out (Ctrl + F8) – шаг из функции.

– Run To Cursor (F4) – выполнять до курсора.

– Toggle Breakpoint (F5) – поставить/удалить точку останова.

– Команда Step Into позволяет «шагать» по каждой строке исходного текста. При помощи ее можно войти в вызываемую функцию.

– Команда Step Over позволяет «перешагнуть» через вызываемую функцию, не входя внутрь, а выполнив ее как единое целое.

Для завершения работы с отладчиком в любой момент времени следует выбрать из меню команду **Run > Stop Debugger**. Произойдет возврат в режим редактирования.

Задание. Выполните тестирование и отладку в IDE первой программы вывода в порт out.c. С этой целью перейдите в режим отладки с помощью команды меню **Run > Start Debugger**.

Чтобы проверить работу программы out.c, нужно следить за состоянием выводов порта B, которые отображаются переменной PORTB. Для занесения в окно наблюдения переменной подведите курсор мыши к строке Select variable from list (выбрать переменную из списка) и щелкните по стрелке в правой части. В раскрывшемся списке выберите строку PORTB и щелкните по ней. Для добавления выбранной переменной PORTB в список окна наблюдения щелкните по кнопке [Add]. Строка с переменной PORTB (желтого цвета) появится в окне наблюдения.

Выполните программу out.c в пошаговом режиме Step Over, используя соответствующую кнопку управления отладкой или клавишу F8. При каждом шаге синяя полоса будет перемещаться по тексту программы. Наблюдайте за значением переменной PORTB на каждом шаге.

Выполнение программы out.c прекращается при достижении правой фигурной скобки функции main( ). Программа зацикливается с помощью команды GOTO $, которую сгенерировал компилятор (смотреть файл out.asm).

Для возможности нового выполнения программы нужно произвести сброс микроконтроллера. Это можно сделать путем повторного запуска отладчика командой **Run > Start Debugger** или щелчком по соответствующей кнопке управления отладкой в окне Watch Values.

Значения переменных в окне наблюдения по умолчанию отображаются в десятичном формате. В программе out.c значения PORTB записаны в шестнадцатеричном формате. Для изменения формата отображения щелкните два раза по строке PORTB. В раскрывшемся окне редактирования Edit Value PORTB поставьте флажок в окошке HEX. Затем щелкните по кнопке [OK] для подтверждения выбора и закрытия окна редактирования.

Выполните вновь программу out.c в пошаговом режиме Step Over. Убедитесь, что формат отображения PORTB стал шестнадцатеричным.

Для завершения работы с отладчиком нужно выполнить команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.4 Вторая программа вывода в порт

Во второй программе происходит бесконечный цикл вывода в порт B микроконтроллера, т. е. состояния выходов порта циклически переключаются.

Текст исходной программы:

```
/*************************************************
 out2.c – вторая программа вывода данных в порт B
*************************************************/
void main( )
{
     TRISB = 0;                  // настроить линии порта B на вывод
      while(1)                   // бесконечный цикл повторения
     {
         PORTB  = 0x00;          // вывод нулей в порт B
         PORTB = 0xFF;           // вывод единиц в порт B
      }
}
```

Задание. Выполните разработку программы в следующей последовательности:

1 С помощью команды **New Project…** запустите New Project Wizard. Создайте новый проект с именем out2 в папке Lab1, микроконтроллер PIC16F84A, тактовая частота 8 МГц.

2 Наберите текст исходной программы out2.c и сохраните его в папке.

3 Выполните построение проекта.

4 При отсутствии ошибок построения проекта переключите mikroC PRO в режим отладки программы с помощью симулятора.

Для тестирования программы занесите в окно наблюдения Watch Values имя порта PORTB из исходной программы. Установите HEX-формат отображения PORTB.

Выполните программу в пошаговом режиме Step Over. После нескольких шагов убедитесь, что состояния выходов порта B циклически переключаются.

Можно узнать время, затраченное на выполнение каждого оператора в строке программы по секундомеру, который находится в окне хронометража Watch Clock. В строке Delta показывается количество командных циклов и время в микросекундах (us), затраченных на выполнение одной строки исходной программы.

Продолжайте выполнять программу в пошаговом режиме Step Over и наблюдайте за показаниями в окне Watch Clock. При каждом шаге показания секундомера Stopwatch будут увеличиваться. По значениям в строке Delta следует, что оператор вывода в порт (строка программы) выполняется за время – 1 мкс.

Выполните сброс МК с помощью команды **Run > Start Debugger** или щелчком по соответствующей кнопке управления в окне Watch Values.

Теперь запустите программу в автоматическом режиме с помощью кнопки управления Run/Pause Debugger или нажатием на клавишу F6. При работе программы в автоматическом режиме значения переменной PORTB не обновляются, а в окне хронометража идет отсчет времени от момента запуска.

Остановите выполнение программы повторным щелчком по кнопке управления Run/Pause Debugger или нажатием на клавишу F6. Зеленая стрелка слева от текста программы покажет номер строки, на которой произошел останов выполнения. Значение переменной PORTB (красного цвета) обновится. В окне Watch Clock будет отображаться общее время выполнения программы в автоматическом режиме от момента запуска.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.5 Программа вывода в порт с временной задержкой

Из результатов исследования программы out2.c следует, что вывод в порт (одна строка программы) выполняется за 1 мкс. Очевидно,

что, если бы к порту были присоединены светодиоды, то их переключение было бы незаметно. Следовательно, необходимо ввести временную задержку между переключениями состояний выводов порта.

Рассмотрим программу, в которой между переключениями выходов порта B имеется временная задержка:

```
/*****************************************************
 out3.c – программа вывода данных в порт B с временной задержкой
*****************************************************/
void main( )
{
    TRISB = 0;
    while(1)
    {
        PORTB = 0x00;
        Delay_ms(500);      // временная задержка на 500 мс
        PORTB = 0xFF;
        Delay_ms(500);      // временная задержка на 500 мс
    }
}
```

В программе out3.c для реализации временной задержки используется встроенная функция компилятора:

Delay_ms (time_in_ms),

где параметр time_in_ms – требуемое время задержки в миллисекундах. Это целое число в пределах от 1 до 4294697295.

Задание. Для исследования программы создайте новый проект с именем out3 и поместите его в папку Lab1, микроконтроллер PIC16F84A, тактовая частота 8 МГц. Наберите текст программы в файле out3.c и сохраните его в папке. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения Watch Values имя порта PORTB из исходной программы.

Выполните программу в пошаговом режиме Step Over. По показаниям в строке Delta окна хронометража убедитесь, что время задержки равно заданному в программе и реализовано компилятором с высокой точностью.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.6 Исследование программы счета с выводом в порт

Рассмотрим программу, которая в бесконечном цикле увеличивает число и выводит его в порт B микроконтроллера:

```
/*****************************************************
count.c – программа счета с выводом результата в порт B
*****************************************************/
char  counter;        // объявление однобайтной переменной-счетчика
void  main( )
{
     TRISB = 0;
     counter = 0;
      while(1)          // бесконечный цикл  счета и вывода в порт
     {
        PORTB = counter;
        counter = counter + 1;
      }
}
```

Задание. Для исследования программы создайте новый проект с именем count и поместите его в папку Lab1, микроконтроллер PIC16F84A, тактовая частота 8 МГц. Наберите текст программы в файле count.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения имена переменных counter и порта PORTB из исходной программы.

Выполните программу в пошаговом режиме Step Over, наблюдая за изменением значений переменных counter и порта PORTB. Убедитесь в правильности работы программы, сделав примерно 10 шагов.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.7 Исследование программы суммирования чисел

Рассмотрим программу, которая вычисляет суммы чисел:

```
/*****************************************************
 sum.c – программа вычисления суммы целых чисел от 1 до 10 и вывода результата в порт B
*****************************************************/
char  sum,   i;                  // объявление однобайтных переменных
void  main( )
{
    TRISB = 0;
    sum = 0;
    for (i = 1; i <= 10; i++)          // цикл суммирования чисел
    {
        sum = sum + i;
     }
     PORTB = sum;        // вывод результата суммирования в порт B
}
```

3.7.1 **Задание.** Для исследования программы создайте новый проект с именем sum и поместите его в папку Lab1, микроконтроллер PIC16F84A, тактовая частота 8 МГц. Наберите текст программы в файле sum.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения имена переменных i, sum, PORTB из исходной программы.

3.7.2 Выполните программу sum.c до конца в пошаговом режиме Step Over. Наблюдайте за изменением значений переменных i, sum, PORTB. Убедитесь, что цикл for выполняется 10 раз.

3.7.3 Теперь исследуйте работу программы в автоматическом режиме с использованием точек останова.

Симулятор среды IDE имеет возможность останавливать выполнение программы в любом месте. Это делается с помощью точек останова (breakpoints). Установить точку останова можно двумя способами.

1 Надо подвести курсор мыши к нужной строке программы и щелкнуть левой кнопкой. Появится мигающая линия, означающая, что данная строка выбрана. Затем подведите курсор к кнопке управления отладкой Toggle Breakpoint в окне Watch Values и щелкните

левой кнопкой (можно вместо этого нажать клавишу F5). Выбранная строка программы окрасится красным цветом, что означает – на установку точки останова.

Для отмены точки останова нужно опять подвести курсор к нужной строке программы и щелкнуть левой кнопкой. Затем надо подвести курсор к кнопке управления Toggle Breakpoint и щелкнуть левой кнопкой (можно вместо этого нажать клавишу F5).

2 Для установки точки останова нужно щелкнуть левой кнопкой мыши по маркеру (синему кружку) в левой части строки программы. Выбранная строка окрасится красным цветом, а маркер поменяет свой цвет на красный.

Для отмены точки останова надо щелкнуть левой кнопкой мыши по маркеру красного цвета выделенной строки текста программы.

Задание. Установите, а затем удалите точки останова в произвольных местах программы с использованием обоих способов.

Предположим, что в программе sum.c нас интересует результат суммирования, выполненный оператором цикла for. Кроме того, нужно узнать время, затраченное на выполнение программы от момента запуска до конца цикла for. С этой целью установите точку останова на строке текста программы:

PORTB = sum;

Выполните сброс микроконтроллера командой **Run > Start Debugger** (клавиша F9). Затем запустите программу на выполнение в автоматическом режиме при помощи кнопки управления Run/Pause Debugger или клавиши F6.

Выполнение программы остановится на строке PORTB = sum, которая примет синий цвет. Результатом суммирования в цикле for будет значение sum = 55. Время выполнения программы до точки останова будет 50 мкс.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.8 Задания для самостоятельной работы

Задание 1 Модернизируйте программу count.c таким образом, чтобы вывод в порт после каждого увеличения переменной counter выполнялся с временной задержкой длительностью 1 с.

Для разработки программы создайте новый проект с именем count2 и поместите его в папку Lab1. Выберите МК PIC16F84A, частота 8 МГц. Запишите текст программы в файл count2.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения имена переменных counter и порта PORTB из исходной программы.

Выполните программу в пошаговом режиме Step Over. Наблюдайте за изменением содержимого переменных counter и PORTB, сделав несколько шагов. Убедитесь, что вывод в порт B происходит через 1 с.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

Задание 2 Модернизируйте программу count.c таким образом, чтобы цикл вывода в порт B выполнялся только 5 раз. После этого выполнение программы прекращается. Для организации цикла вывода используйте оператор for (как в программе sum.c).

Для разработки программы создайте новый проект с именем count3 в папке Lab1. Выберите МК PIC16F84A, тактовая частота 8 МГц. Запишите текст программы в файл count3.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения имена переменных counter и порта PORTB из исходной программы.

Выполните программу в пошаговом режиме Step Over. Наблюдайте за изменением содержимого переменных counter и PORTB. При каких значениях counter и PORTB прекращается цикл вывода?

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

4 Содержание отчета

Укажите наименование и цель работы. Представьте тексты программ к заданиям для самостоятельной работы (комментарии в программах обязательны!).

5 Контрольные вопросы

1 Что такое IDE?

2 Какие функции выполняет IDE для PIC?

3 Поясните процесс разработки программного обеспечения в среде для PIC-микроконтроллеров.

4 Что такое проект в IDE?

5 На каком этапе осуществляется выбор микроконтроллера?

6 Какие окна используются в симуляторе для наблюдения за ходом выполнения программы?

7 Как можно осуществить пошаговое выполнение программы?

8 Что такое точка останова?

Лабораторная работа № 2

ИССЛЕДОВАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ ЯЗЫКА ПРОГРАММИРОВАНИЯ СИ

1 Цель работы

Изучить операции языка программирования Си: арифметические; присваивания; логические; отношения и поразрядные. Практически исследовать выполнение этих операций в программах для микроконтроллеров семейства PIC16.

2 Основные теоретические сведения

Программа на языке Си состоит из выражений. Выражение – это последовательность операндов, операций и символов-разделителей. Основные операции в Си: арифметические, присваивания, отношения, логические, поразрядные. Для обозначения этих операций используются специальные символы (значки).

3 Порядок выполнения работы

3.1 Создание папки для работы

Предположим, что при выполнении лабораторной работы № 2 мы будем использовать папку с именем Lab2, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку и создайте в ней (с помощью клавиши F7) новую папку с именем Lab2.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы при выполнении данной лабораторной работы (работы № 2). Полный путь к этой папке будет такой:

F:\MPinCS\Ivanov\Lab2

3.2 Исследование арифметических операций

3.2.1 Язык Си для PIC-микроконтроллеров включает стандартный набор арифметических операций, обозначаемых значками: сложение “ + ”, вычитание “ – “, умножение “ * ”, деление “ / “, которые не требуют особого пояснения. Специфичными являются операции определения остатка от деления, а также инкремента и декремента.

Операцию определения остатка от деления, обозначаемую значком “ % “, поясняет следующий пример:

```
int  a = 5,  b = 2,  d;
d = a % b;            // d = 1 – остаток от деления  5 / 2
d = b % a;            // d = 2
```

Следует отметить, что операция определения остатка от деления применима только к целым числам.

Операции инкремента (обозначается как “ ++ “) и декремента (обозначается как “ -- “) могут применяться только к переменным. Существуют две формы их записи: префиксная, когда операнд располагается справа от знака операции (например, ++i, --j), и постфиксная, когда операнд располагается слева от знака операции (i++, j--).

В префиксной форме (для инкремента) сначала выполняется увеличение операнда на 1, и увеличенное значение используется в выражении. В постфиксной форме (для инкремента) сначала используется в выражении значение операнда и только после этого его значение увеличивается на 1. Например:

```
int  a = 0,  b = 1,  d;
d = a++;        // d = 0,  a = 1
d = ++a;        // d = 2,  a = 2
d = ++b;        // d = 2,  b = 2
```

Рассмотрим программу arifm.c, в которой будут использоваться некоторые арифметические операции:

```
/*****************************************************
arifm.c – первая программа для исследования арифметических операций

*****************************************************/
int  i,  k = 6,  n,  m;                    // объявления переменных
void  main( )
{
   i = 10 * (k++);                 // i =     , k =
   k--;                            // k =
   i = 10 * (++k);                 // i =    , k =
   k--;                            // k =
   n = i / k;                      // n =
   m = i % k;                      // m =
}
```

Задание. Для исследования программы создайте новый проект с именем arifm в папке Lab2. Выберите МК PIC16F84A, тактовая частота 8 МГц. Наберите текст программы в файле arifm.c и сохраните его. Выполните построение проекта и затем перейдите в режим отладки.

Для тестирования программы и занесите в окно наблюдения Watch Values имена переменных i, j, k, n, m из исходной программы.

Примечание. 1. Если после запуска отладчика на экране будет отсутствовать окно наблюдения Watch Values, то его можно открыть с помощью команды меню View > Debug Window > Watch Window или нажатием комбинации клавиш Shift + F5. 2. Формат отображения данных в окне наблюдения будет по умолчанию десятичный.

Исследуйте работу программы arifm.c в пошаговом режиме Step Over. После каждого шага наблюдайте за содержимым переменных в окне Watch Values и записывайте эти значения в текст комментария программы. Эти значения необходимо будет привести в отчете по данной работе. Объясните полученные результаты выполнения операторов программы.

Для завершения работы с отладчиком нужно выполнить команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования. Сохраните файл arifm.c с комментариями, полученными в результате исследования программы, с помощью команды меню **File > Save**.

В заключение закройте проект с помощью команды **Project > Close Project.**

3.2.2 Операция присваивания в языке Си обозначается как “=”. Обычно она используется в виде

переменная = выражение;

При выполнении операции присваивания переменная получает значение выражения. Выражение может быть одиночной константой или сложной комбинацией переменных, операторов и констант. Например:

```
x = 10;
x = x + y;
x = x + y + 50;
```

При программировании часто используются операторы вида

```
x = x + 4;
x = x – y;
```

и т. п.

В этих операторах переменная, которой присваивается результат выражения, является также первым операндом. Для выполнения подобных операций язык mikroC предлагает составные операторы присваивания, которые объединяют простые арифметические операции с присваиванием. В таблице 2.1 приведены арифметические составные операторы присваивания.

Таблица 2.1 – Арифметические составные операторы присваивания

Оператор присваивания	Длинная форма	Пример
x += y;	x = x + y;	x += 12;
x -= y;	x = x – y;	x -= 34 + y;
x *= y;	x = x * y;	x *= 10;
x /= y;	x = x / y;	x /= 5;
x %= y;	x = x % y;	x %=2;

Примечание. В составном операторе присваивания между знаком арифметической операции (+, -, *, /, %) и знаком присваивания “=” пробел не допускается!

В качестве примера рассмотрим программу arifm2.c, в которой используются арифметические составные операторы присваивания для целых чисел и чисел с плавающей точкой:

```
/*********************************************************
arifm2.c – вторая программа для исследования арифметических операций
*********************************************************/
int   i = 55,  j = 66;
float   x = 2.5,  y = 4.6;
void  main( )
{
    i += j;              // i =
    j -= 6;              // j =
    i *= 4;              // i =
    j /= 3;              // j =
    i %= 2;              // i =
    x += y;              // x =
    y -= 4.0;            // y =
    x *= 4.0;            // x =
    y /= 3.0;            // y =
}
```

Задание. Для исследования программы создайте новый проект с именем arifm2 в папке Lab2, МК PIC16F84A, частота 8 МГц. Наберите текст программы в файле arifm2.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения Watch Values имена переменных i, j, x, y из исходной программы.

Исследуйте работу программы arifm2.c в пошаговом режиме Step Over. После каждого шага наблюдайте за содержимым переменных в окне Watch Values и записывайте эти значения в текст комментария программы. Эти значения необходимо будет привести в отчете по данной работе. Объясните полученные результаты выполнения операторов программы.

Для завершения работы с отладчиком нужно выполнить команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования. Сохраните файл arifm2.c с комментариями, полу-

ченными в результате исследования программы, используя команду **File > Save**.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.3 Исследование операций отношения и логических

3.3.1 Операции отношения применяются для вычисления соотношений между операндами. Логические операции, используя правила логики, также возвращают соотношения между операндами.

В формальной логике ключевым понятием являются ЛОЖЬ и ИСТИНА. В языке Си лжи соответствует 0, а истине – любое значение, отличное от 0. Выражения, использующие операции отношения или логические, возвращают 0 для лжи и 1 для истины. В таблице 2.2 приведены обозначения логических и операций отношения, применяемых в языке mikroC.

Таблица 2.2 – Логические операции и операции отношения

Знак операции	Выполняемое действие	Пример
&&	Логическое И	if (k > 1 && k < 10)
||	Логическое ИЛИ	if (c == 0 || c == 9)
!	Логическое НЕ	if (!(c > 1 && c < 9))
<	Меньше	if (j < 0)
<=	Меньше или равно	if (j <= 0)
>	Больше	if (j > 10)
>=	Больше или равно	if (x >= 8.2)
==	Равно	if (c == b)
!=	Не равно	if (c != b)

В качестве примера рассмотрим программу logic.c, в которой используются операции отношения и логические для целых чисел:

```
/*****************************************************
logic.c – первая  программа для исследования операций отношения и логических
*****************************************************/
char  var1 = 10,  var2 = 20;
char  res1, res2, res3, res4, res5, res6, res7, res8;
void  main( )
{
```

```
    res1 = var1 > var2;          // res1 =
    res2 = var1 < var2;          // res2 =
    res3 = var1 == var2;         // res3 =
    res4 = var1 != var2;         // res4 =
    res5 = var1 && var2;         // res5 =
    res6 = var1 || var2;         // res6 =
    res7 = ! var1;               // res7 =
    res8 = ! var2;               // res8 =
}
```

Задание. Для исследования программы создайте новый проект с именем logic в папке Lab2, МК PIC16F84A, частота 8 МГц. Наберите текст программы в файле logic.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения Watch Values имена переменных var1, var2, res1, res2, res3, res4, res5, res6, res7, res8 из исходной программы.

Исследуйте работу программы logic.c в пошаговом режиме Step Over. После каждого шага наблюдайте за содержимым переменных в окне Watch Values и записывайте эти значения в текст комментария программы. Эти значения необходимо будет привести в отчете по данной работе. Объясните полученные результаты выполнения операторов программы.

Для завершения работы с отладчиком нужно выполнить команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования. Сохраните файл logic.c с комментариями, полученными в результате исследования программы.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.4 Исследование поразрядных операций

3.4.1 В языке Си широко используются поразрядные (побитовые) операции, с помощью которых можно выполнять тестирование, установку, сброс, инвертирование и сдвиг отдельных битов операндов. Поразрядные операции могут выполняться только с целыми типами данных, т. е. char, int, long. В таблице 2.3 приведены поразрядные операции языка mikroC.

Таблица 2.3 – Поразрядные операции в языке Си

Знак операции	Выполняемое действие	Пример
&	Поразрядное И	i & 0x25
|	Поразрядное ИЛИ	j | 64
^	Поразрядное исключающее ИЛИ	k ^ 0x0F
~	Поразрядное НЕ (инверсия)	~ m
<<	Поразрядный сдвиг влево	i << 2
>>	Поразрядный сдвиг вправо	j >> 3

Рассмотрим программу bit_op.c, в которой используются различные поразрядные операции:

```
/*****************************************************
 bit_op.c – первая программа для исследования поразрядных операций
*****************************************************/
char  i = 0x0F,  j = 0x1A,  k1,  k2,  k3,  k4,  k5,  k6,  k7;
void  main( )
{
    k1 = i & j;              // k1 =
    k2 = i | j;              // k2 =
    k3 = i ^ j;              // k3 =
    k4 = ~ i;                // k4 =
    k5 = ~ j;                // k5 =
    k6 = i << 2;             // k6 =
    k7 = j >> 3;             // k7 =
}
```

Задание. Для исследования программы создайте новый проект с именем bit_op в папке Lab2, МК PIC16F84A, частота 8 МГц. Наберите текст программы в файле bit_op.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения Watch Values имена переменных k1, k2, k3, k4, k5, k6, k7 из исходной программы. Установите двоичный формат отображения значений переменных.

Исследуйте работу программы bit_op.c в пошаговом режиме Step Over. После каждого шага наблюдайте за содержимым переменных в окне Watch Values и записывайте эти значения в текст комментария программы. Эти значения необходимо будет привести в отчете

по данной работе. Объясните полученные результаты выполнения операторов программы.

Для завершения работы с отладчиком нужно выполнить команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования. Сохраните файл bit_op.c с комментариями, полученными в результате исследования программы.

В заключение закройте проект с помощью команды **Project > Close Project.**

3.4.2 Для сокращения текста программ в Си широко используются поразрядные составные операторы присваивания, которые являются комбинацией поразрядных операций и присваивания. В таблице 2.4 приведены составные поразрядные операции присваивания.

Таблица 2.4 – Составные поразрядные операторы присваивания

Оператор	Длинная форма	Пример
x &= y;	x = x & y;	i &= 0x0F;
x |= y;	x = x | y;	j |= 64;
x ^= y;	x = x ^ y;	k ^= 0x80;
x <<= n;	x = x << n;	j <<= 2;
x >>= n;	x = x >> n;	k >>= 3;

Примечание. В составных операторах присваивания между знаком поразрядной операции (&, |, ^, >>, <<) и знаком присваивания "=" пробел не допускается!

Рассмотрим программу bit_op2.c, в которой используются различные составные поразрядные операторы присваивания:

```
/*****************************************************
 bit_op2.c – вторая программа для исследования поразрядных
 составных операций присваивания
*****************************************************/
char   i, j, k;
void  main( )
{
    i = 0xFF;                 // i =
    i &= 0x0F;                // i =
    i |= 0x31;                // i =
    i ^= 0x94;                // i =

    j = 0xFF;                 // j =
```

```
    /* установить код  0110 в старшей тетраде переменной  j  */
  j &= 0x0F;                  // j =
  j |= 0x60;                  // j =
     /* инвертировать 7-й разряд в переменной  j  */
  j ^= 0x80;                  // j =
      /*  инвертировать 3-й разряд в переменной  j  */
  j ^=0x04;                   // j =

  k = 0xF5;                   // k =
      /*  сдвиги беззнаковой переменной  k  */
  k >>= 1;                    // k =
  k >>= 2;                    // k =
  k <<= 1;                    // k =
  k <<= 2;                    // k =

}
```

Задание. Для исследования программы создайте новый проект с именем bit_op2 папке Lab2, МК PIC16F84A, частота 8 МГц. Наберите текст программы в файле bit_op2.c и сохраните его. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования программы занесите в окно наблюдения Watch Values имена переменных i, j, k из исходной программы. Установите двоичный формат отображения значений переменных.

Исследуйте работу программы bit_op2.c в пошаговом режиме Step Over. После каждого шага наблюдайте за содержимым переменных в окне Watch Values и записывайте эти значения в текст комментария программы. Эти значения необходимо будет привести в отчете по данной работе. Объясните полученные результаты выполнения операторов программы.

Для завершения работы с отладчиком нужно выполнить команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования. Сохраните файл bit_op2.c с комментариями, полученными в результате исследования программы.

В заключение закройте проект с помощью команды **Project > Close Project**.

4 Содержание отчета

Укажите наименование и цель работы. Представьте таблицы 2.1– 2.4 и тексты всех исследуемых программ с результатами выполнения.

5 Контрольные вопросы

1 Какие арифметические операции имеются в языке Си?

2 Как выполняются операции прединкремента и постинкремента? Приведите примеры.

3 Как выполняется операция определения остатка от деления? Приведите пример.

4 Как обозначаются и выполняются составные арифметические операции присваивания?

5 Как кодируются логические понятия ЛОЖЬ и ИСТИНА в Си?

6 Какие логические операции имеются в языке Си?

7 Какие операции отношения имеются в языке Си?

8 Какие имеются в языке Си поразрядные операции?

9 Как обозначаются и выполняются составные поразрядные операции присваивания?

Лабораторная работа № 3

ИССЛЕДОВАНИЕ ОПЕРАТОРОВ ВЫБОРА ДЛЯ УПРАВЛЕНИЯ ПРОГРАММОЙ В ЯЗЫКЕ СИ

1 Цель работы

Изучить операторы выбора для управления программой в языке Си. Практически исследовать выполнение этих операторов в программах для микроконтроллеров семейства PIC16.

2 Основные теоретические сведения

Операторы управления программой – это средства, с помощью которых можно изменять порядок выполнения программы. Они позволяют выполнять ветвление, циклическое повторение одного или нескольких операторов, передачу управления в нужное место кода программы.

Язык Си предоставляет три категории операторов управления программой:

1) операторы выбора – это if и switch;

2) операторы цикла – это while, do … while и for;

3) операторы перехода – это break, continue и goto.

В данной лабораторной работе Вы будете изучать и исследовать операторы выбора.

3 Порядок выполнения работы

3.1 Создание папки для работы

При выполнении лабораторной работы № 3 мы будем использовать папку с именем Lab3, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку и создайте в ней (с помощью клавиши F7) новую папку с именем Lab3.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 3. Полный путь к этой папке будет такой:

f:\MPinCS\Ivanov\Lab3

В данной лабораторной работе будут широко использоваться операции ввода и вывода данных через порты микроконтроллера. Поэтому сначала рассмотрим, как в языке Си программируются операции работы с портами.

3.2 Исследование операций ввода/вывода в Си

Микроконтроллеры семейства PIC16 имеют несколько портов ввода/вывода, позволяющих получать информацию от внешних устройств (ввод данных) или отправлять данные внешним устройствам (вывод данных). Количество портов и их разрядность (по-другому, число линий порта) определяется конкретным типом микроконтроллера. Так, простой микроконтроллер PIC16F84A имеет два порта, которые обозначаются буквами A и B. Причем порт A имеет 5 линий (5-разрядный), а порт B имеет 8 линий (8-разрядный). Более сложный микроконтроллер PIC16F877 имеет 5 портов, которые обозначаются буквами A, B, C, D, E. Порт A имеет 6 линий, порты B, C и D – по 8 линий каждый, а порт E имеет только 3 линии.

Для управления вводом/выводом через каждый порт служит соответствующая пара регистров:

– регистр данных – PORTx;

– регистр управления направлением передачи – TRISx.

Количество таких пар регистров соответствует количеству портов. С их помощью через любой порт может быть одновременно выдано или принято от одного до восьми бит данных.

Порты в микроконтроллерах PIC – двунаправленные. Это означает, что любой вывод порта может использоваться и как вход, и как выход. Направление передачи задает соответствующий управляющий

регистр TRISx. Запись в некоторый разряд регистра TRISx логической единицы делает соответствующий вывод порта входом, а логического нуля – выходом. Например, для того чтобы нулевой и первый разряды порта B были входами, а все остальные – выходами, в регистр TRISB необходимо записать значение 00000011.

Настроив порты на вывод, мы можем их использовать для вывода данных в порт. Например, мы можем вывести какие-либо данные в порт C микроконтроллера PIC16F877 следующим образом:

```
char  port_data;          // объявление однобайтной переменной
TRISC = 0;                // настроить все линии порта C на вывод
port_data = 0x0F;      // присвоить переменной port_data значение 0x0F
PORTC = port_data;     // вывести значение  переменной port_data
                          // в порт C
PORTC = 0xF0;          // вывести значение 0xF0  в порт C
```

Если настроить порты на ввод, то их можно использовать для чтения данных с их входов (для ввода данных из портов). Так, для чтения данных из порта B (ввода данных) микроконтроллера PIC16F877 нужно выполнить следующий код программы:

```
char port_data;           // объявление однобайтной переменной для
                          // хранения введенных данных
TRISB = 0xFF;             // настроить все линии порта B  на ввод
port_data = PORTB;     // чтение данных с линий порта B (ввод из
                          //  порта B)
```

Рассмотрим в качестве примера программу, в которой выполняется ввод данных из порта B, а затем их вывод в порт C микроконтроллера PIC16F877:

```
/*****************************************************
in_out.c – программа для исследования ввода и вывода данных
Микроконтроллер PIC16F877
*****************************************************/
char  in_port;          // переменная для хранения данных, введенных
                        // из порта B
void  main( )
{
```

```
    TRISB =0xFF;              // настроить все линии порта B на ввод
    TRISC = 0;                // настроить все линии порта C на вывод
    while(1)                  // бесконечный цикл ввода и вывода
    {
       in_port = PORTB;       // ввод из порта B
       PORTC = in_port;       // вывод в порт C
     }
}
```

Задание. Выполните разработку программы в следующей последовательности:

1 Запустите программу IDE.

2 С помощью команды New Project… запустите New Project Wizard. Создайте новый проект с именем in_out в папке f:\....\Lab3. Выберите микроконтроллер PIC16F877, тактовая частота 8 МГц.

3 Наберите текст исходной программы in_out.c и сохраните его в папке (с помощью команды меню **File > Save**).

4 Выполните построение проекта (с помощью команды меню **Build > Build**).

5 При отсутствии ошибок построения проекта переключите mikroC PRO в режим отладки (с помощью команды меню **Run > Start Debugger**).

Проведите исследование работы программы in_out.c в следующей последовательности.

Для тестирования программы занесите в окно наблюдения Watch Values последовательно имена портов PORTB и PORTC, а затем имя переменной in_port.

Примечания. 1. Если после запуска отладчика на экране будет отсутствовать окно наблюдения Watch Values, то его можно открыть с помощью команды меню View > Debug Window > Watch Window или нажатием комбинации клавиш Shift + F5. 2. Формат отображения данных в окне наблюдения будет по умолчанию десятичный.

Выполните программу in_out.c в пошаговом режиме Step Over путем нажатия на клавишу F8, или щелкая мышью по соответствующему значку управления отладкой.

Когда синяя полоса окажется на строке с номером 7 (in_port = PORTB;), поставьте курсор мыши на колонку Value в строке PORTB и дважды щелкните левой кнопкой. Мигающая линия курсора появится справа от числа. С помощью клавиши Back Space удалите это число и наберите новое число, например, 10. Затем нажмите клавишу Enter.

В строке PORTB появится новое значение Value (красного цвета), равное 10. Так как линии порта B настроены на ввод, то набранное новое значение 10 будет являться кодом данных (в десятичном формате) на входах порта B.

Выполните следующий шаг программы – строку 8 (in_port = PORTB;). Убедитесь, что значение переменной in_port стало равным 10, т. е. произошел ввод данных со входов порта B.

Выполните следующий шаг программы – строку 9 (PORTC = in_port;). Убедитесь, что содержимое регистра данных порта PORTC стало равным 10, т. е. произошел вывод в порт C.

Когда после очередного шага синяя полоса вновь окажется на строке с номером 7 (in_port = PORTB;), измените данные на входах порта B. Введите, например, число 100. Продолжайте выполнять программу в пошаговом режиме, наблюдая за содержимым переменных в окне Watch Values. Объясните полученные результаты работы программы.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом произойдет возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.3 Исследование операторов выбора

Язык Си поддерживает два типа операторов выбора: if (если) и switch (переключатель). Эти операторы позволяют проверять выполнение определенных условий и выбирать возможное продолжение вычислительного процесса. Возможны следующие конструкции этих операторов:

1) оператор if единственного выбора;
2) оператор if … else двойного выбора;
3) оператор if … else if … else множественного выбора;
4) оператор switch множественного выбора.

3.3.1 *Исследование оператора* if *единственного выбора*. Оператор if единственного выбора для одного выполняемого оператора имеет вид

```
if (условие)
    оператор;
```

для блока выполняемых операторов:

```
if (условие)
```

```
{
    оператор1;
    ……..
    операторN;
}
```

Программа вычисляет условие, заключенное в круглых скобках. Если оно истинно, то выполняется оператор (или блок операторов, заключенный в фигурных скобках). Если условие ложно, то оператор (или блок) не выполняется.

В качестве примера рассмотрим программу if1.c, в которой используется оператор if единственного выбора:

```
/*********************************************************
 if1.c –  программа для исследования оператора  if  единственного выбора.  Микроконтроллер PIC16F877
*********************************************************/
char  in_port;     // переменная для хранения вводимых данных
                   // из порта B
void  main( )
{
    TRISB = 0xFF;                  // настроить порт B  на ввод
    TRISC = 0;                     // настроить порт C на вывод
    PORTC = 0x00;                  // вывести нули в порт C
    in_port = PORTB;               // ввод из порта B
    if (in_port  >=  0x0F)
        PORTC = in_port;           // вывод в порт C
}
```

В программе вводится число из порта B и записывается в переменную in_port. Если это значение больше или равно значению 0x0F = 00001111, то программа выводит его в порт C. В противном случае программа никаких действий не производит.

Задание. Выполните разработку программы в следующей последовательности:

1 С помощью команды **New Project…** запустите New Project Wizard. Создайте новый проект с именем if1 в папке f:\....\Lab3. Выберите микроконтроллер PIC16F877, тактовая частота 8 МГц.

2 Наберите текст исходной программы if1.c и сохраните его в папке (с помощью команды меню **File > Save**).

3 Выполните построение проекта (с помощью команды меню **Build > Build**).

4 При отсутствии ошибок построения проекта переключите mikroC PRO в режим отладки (с помощью команды меню **Run > Start Debugger**).

Проведите исследование работы программы if1.c в следующей последовательности.

Для тестирования программы занесите в окно наблюдения Watch Values последовательно имена портов PORTB и PORTC, а затем имя переменной in_port.

Для исследования работы программы измените формат отображения всех переменных в окне наблюдения на шестнадцатеричный.

Выполните программу if1.c в пошаговом режиме Step Over путем нажатия на клавишу F8, или щелкая мышью по соответствующему значку управления отладкой.

Когда синяя полоса окажется на строке с номером 6 (in_port = PORTB;), поставьте курсор мыши на колонку Value в строке PORTB и дважды щелкните левой кнопкой. Наберите новое число, например, 0x7F, а затем нажмите клавишу Enter. Продолжайте выполнять программу в пошаговом режиме, наблюдая за перемещением синей полосы по ветвям программы.

Когда синяя полоса окажется на фигурной скобке (конец функции main), дальнейшее выполнение программы прекратится. Произведите сброс микроконтроллера с помощью команды меню **Run > Start Debugger**, или щелкнув по соответствующему значку управления отладкой. Введите новое значение на входах порта B, например, 0x03. Выполните программу до конца в пошаговом режиме, наблюдая за перемещением синей полосы по ветвям программы.

Объясните полученные результаты выполнения операторов программы при различных значениях данных на входах порта B.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**, при этом происходит возврат в режим редактирования.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.3.2 *Исследование оператора* if … else *двойного выбора*. Оператор if … else (если … то) двойного выбора обеспечивает две аль-

тернативы продолжения выполнения программы. Выбор осуществляется, исходя из проверяемого условия.

Общий вид оператора if … else двойного выбора для одного выполняемого оператора:

```
if (условие)
    оператор1;
else
    оператор2;
```

Вместо одиночных операторов 1 и 2 могут быть блоки операторов, заключенные в фигурные скобки.

В случае истинности условия выполняется оператор1, в противном случае – оператор2.

Рассмотрим в качестве примера программу if2.c, где использован оператор if … else двойного выбора:

```
/*******************************************************
 if2.c –  программа для исследования оператора  if … else  двойного выбора. Микроконтроллер PIC16F877
*******************************************************/
char  in_port;      // переменная для хранения вводимых данных
                    // из порта B
void  main( )
{
   TRISB = 0xFF;            // настроить все линии порта B на ввод
   TRISC = 0;               // настроить все линии порта C на вывод
   TRISD = 0;               // настроить все линии порта D на вывод
   PORTC = 0x00;                    // вывод нулей в порт C
   PORTD = 0x00;                    // вывод нулей в порт D
   in_port = PORTB;                         // ввод из порта B
   if (in_port > 0  &&  in_port < 0x0F)
      PORTC = in_port;                      // вывод в порт C
   else
      PORTD = in_port;                      // вывод в порт D
}
```

В программе if2.c вводится число из порта B и записывается в переменную in_port. Если это значение находится в диапазоне от 0

до 0x0F = 00001111, то программа выводит его в порт C. В противном случае это число выводится в порт D.

Задание. Выполните разработку программы в следующей последовательности:

1 С помощью команды **New Project…** запустите New Project Wizard. Создайте новый проект с именем if2 в папке f:\....\Lab3. Выберите микроконтроллер PIC16F877, тактовая частота 8 МГц.

2 Наберите текст исходной программы if2.c и сохраните его в папке (с помощью команды меню **File > Save**).

3 Выполните построение проекта (с помощью команды меню **Build > Build**).

4 При отсутствии ошибок построения проекта переключите mikroC PRO в режим отладки (с помощью команды меню **Run > Start Debugger).**

Проведите исследование работы программы if2.c в следующей последовательности.

Для тестирования программы занесите в окно наблюдения Watch Values последовательно имена портов PORTB, PORTC и PORTD, а затем имя переменной in_port.

Для исследования работы программы измените формат отображения всех переменных в окне наблюдения на шестнадцатеричный.

Выполните программу if2.c в пошаговом режиме Step Over путем нажатия на клавишу F8, или щелкая мышью по соответствующему значку управления отладкой.

Когда синяя полоса окажется на строке с номером 8 (in_port = PORTB;), поставьте курсор мыши на колонку Value в строке PORTB и дважды щелкните левой кнопкой. Наберите новое число, например, 0x7F, а затем нажмите клавишу Enter. Продолжайте выполнять программу в пошаговом режиме, наблюдая за перемещением синей полосы по ветвям программы.

Когда синяя полоса окажется на фигурной скобке (конец функции main), дальнейшее выполнение программы прекратится. Произведите сброс микроконтроллера с помощью команды меню **Run > Start Debugger**, или щелкнув по соответствующему значку управления отладкой. Введите новое значение на входах порта B, например, 0x03. Выполните программу до конца в пошаговом режиме, наблюдая за перемещением синей полосы по ветвям программы.

Объясните полученные результаты выполнения операторов программы при различных значениях данных на входах порта B.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.2.3 *Исследование оператора* if … else if … else *множественного выбора*. Язык mikroC позволяет использовать вложенные операторы if … else для реализации множественного выбора.

Общий вид оператора if … else if … else множественного выбора:

```
if (условие1)
    оператор1;
else if (условие2)
    оператор2;
 ……….
else if (условиеN)
    операторN;
else
    оператор(N+1);
```

Последняя else-часть в операторе if … else может и отсутствовать.

Оператор if … else if … else множественного выбора выполняет серию последовательных проверок до тех пор, пока не будет установлено следующее:

1 Одно из условий в if-части или в частях else if является истиной. В этом случае выполняются соответствующие ему операторы.

2 Ни одно из вложенных условий не является истиной. Программа выполнит операторы в последней else-части, если она имеется.

Рассмотрим программу if3.c, в которой используется оператор if … else if … else множественного выбора:

```
/*****************************************************
 if3.c – программа исследования оператора  if … else if … else
множественного выбора. Микроконтроллер PIC16F877
*****************************************************/
char  in_port;    // переменная для хранения вводимых данных
                                        // из порта B
void  main( )
{
```

```
    TRISB = 0xFF;                       // настроить порт B на ввод
    TRISC = 0;                          // настроить порт C на вывод
    PORTC =0x00;                        // вывод нулей в порт C
    in_port = PORTB;                    // ввод данных из порта B
    if (in_port > 0  &&  in_port < 0x0F)
        PORTC = 0x01;                   // вывод в порт C значения 0x01
    else  if (in_port >= 0x0F  &&  in_port < 0x3F)
        PORTC = 0x02;                   // вывод в порт C значения 0x02
    else  if (in_port >= 0x3F  &&  in_port < 0x7F)
        PORTC = 0x03;                   // вывод в порт C значения 0x03
    else
        PORTC = 0x04;                   // вывод в порт C значения 0x04
}
```

Программа вводит данные с входов порта B и записывает их в переменную in_port. Затем используется оператор if … else if … else множественного выбора для определения нахождения значения in_port в определенном диапазоне:

```
0 <  in_port  <  0x0F          – диапазон 1
0x0F <=  in_port  <  0x3F   – диапазон 2
0x3F <=  in_port  <  0x7F   – диапазон 3
0x7F <=  in_port  <=  0xFF – диапазон 4
```

Полученное значение номера диапазона выводится в порт C.

Задание. Выполните разработку программы в следующей последовательности:

1 С помощью команды **New Project…** запустите New Project Wizard. Создайте новый проект с именем if3 в папке f:\....\Lab3. Выберите микроконтроллер PIC16F877, тактовая частота 8 МГц.

2 Наберите текст исходной программы if3.c и сохраните его в папке (с помощью команды меню **File > Save**).

3 Выполните построение проекта (с помощью команды меню **Build > Build**).

4 При отсутствии ошибок построения проекта переключите mikroC PRO в режим отладки (с помощью команды меню **Run > Start Debugger**).

Проведите исследование работы программы if3.c в следующей последовательности.

Для тестирования программы занесите в окно наблюдения Watch Values последовательно имена портов PORTB и PORTC, а затем имя переменной in_port.

Для исследования работы программы измените формат отображения всех переменных в окне наблюдения на шестнадцатеричный.

Выполните программу if3.c в пошаговом режиме Step Over путем нажатия на клавишу F8, или щелкая мышью по соответствующему значку управления отладкой.

Когда синяя полоса окажется на строке с номером 6 (in_port = PORTB;), поставьте курсор мыши на колонку Value в строке PORTB и дважды щелкните левой кнопкой. Наберите новое число, например, 0x07 (соответствующее диапазону 1), а затем нажмите клавишу Enter. Продолжайте выполнять программу в пошаговом режиме, наблюдая за перемещением синей полосы по ветвям программы. Убедитесь в правильности выполнения программы по значению, выведенному в порт C.

Когда синяя полоса окажется на фигурной скобке (конец функции main), дальнейшее выполнение программы прекратится. Произведите сброс микроконтроллера с помощью команды меню **Run > Start Debugger**, или щелкнув по соответствующему значку управления отладкой. Введите новое значение на входах порта B, например, 0x1F (соответствующее диапазону 2). Выполните программу до конца в пошаговом режиме, наблюдая за перемещением синей полосы по ветвям программы. Убедитесь в правильности выполнения программы по значению, выведенному в порт C.

Далее исследуйте работу программы при значениях 0x5F (диапазон 3) и 0x8F (диапазон 4) на входах порта B.

Объясните полученные результаты выполнения операторов программы при различных значениях данных на входах порта B.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.3.4 *Исследование оператора выбора* switch. Оператор switch (можно перевести как переключатель) используется для выбора одного варианта из многих. Он проверяет, совпадает ли значение выражения с одним из значений, входящих в некоторое множество целых констант, и выполняет соответствующую этому значению ветвь программы.

Общий вид оператора switch:

```
switch (выражение)
{
    case  константа1:
      оператор1;
      break;
    case  константа2:
      оператор2;
      break;
    ……
    case  константаN:
      операторN;
      break;
     default:
      оператор(N+1);
}
```

Оператор switch выполняется так. Сначала вычисляется выражение, стоящее в скобках после ключевого слова switch. Вычисленное значение сравнивается со значением констант: константа1, константа2, …, константаN. При совпадении вычисленного значения с некоторой константой выполняется соответствующий ей оператор. Затем управление передается оператору break (прервать), который производит немедленный выход из оператора switch. Если вычисленное значение не совпадает ни с одной из констант, выполняется оператор в ветви, помеченной default (по умолчанию).

Правила использования оператора switch следующие.

1 Switch требует выражения целого типа. Это значение может быть константой, переменной или выражением. Оператор switch не работает с типами данных с плавающей точкой.

2 Значение после каждой метки case должно быть константой.

3 Язык Си не поддерживает метки case, содержащие диапазон значений. Каждое значение должно указываться с отдельной меткой case.

4 Необходимо использовать оператор break после каждого набора выполняемых операторов. Оператор break вызывает продолжение выполнения программы после завершения текущего оператора switch. Если не использовать оператор break, то выполнение программы продолжится на последующих метках case.

5 Ветвь, помеченная словом default (по умолчанию), выполняется тогда, когда ни одна из констант ветвей case не подходит. Ветвь default не является обязательной и, если она отсутствует, оператор switch ничего не выполнит.

6 Набор операторов в каждой ветви case не нужно заключать в фигурные скобки.

Примечание. В связи с тем, что в ветвях, помеченных словом case, можно указывать только константу, для анализа принадлежности к диапазону значений следует использовать оператор if … else if … else множественного выбора.

Рассмотрим программу, в которой используется оператор switch:

```
/*****************************************************
 switch.c – программа исследования оператора выбора  switch
Микроконтроллер PIC16F877
*****************************************************/
char  in_port;              // переменная для хранения введенного кода
void  main( )
{
    TRISB = 0xFF;                    // настроить порт B на ввод
    TRISC = 0;                       // настроить порт C на вывод
    PORTC = 0x00;                    // вывод нулей в порт C
    in_port = PORTB;                 // ввод данных из порта B
    switch (in_port)
    {
        case   0xFE:
            PORTC = 0x01;            // вывод «вариант 1» в порт C
            break;
        case   0xFD:
            PORTC = 0x02;            // вывод «вариант 2» в порт C
            break;
        case    0xFB:
            PORTC = 0x03;            // вывод «вариант 3» в порт C
            break;
        case    0xF7:
            PORTC = 0x04;            // вывод «вариант 4» в порт C
            break;
         default:
             PORTC = 0xFF;         // вывод «неизвестный код» в порт C
```

```
    }
}
```

В программе switch.c предполагается ввод определенного кода из порта B. С помощью оператора switch проверяется один из четырех вариантов входного кода. Для каждого варианта кода выполняется вывод соответствующего «сообщения» в порт C. В качестве такого «сообщения» выбрано определенное число:

Входной код из порта B	Вариант кода	Код для вывода в порт C
0xFE = 11111110	Вариант 1	0x01 = 00000001
0xFD = 11111101	Вариант 2	0x02 = 00000010
0xFB = 11111011	Вариант 3	0x03 = 00000011
0xF7 = 11110111	Вариант 4	0x04 = 00000100

Если входной код не соответствует ни одному из указанных четырех вариантов, то в порт C выводится код 0xFF = 11111111.

Входной код представляет собой двоичное число, в котором только один 0. Такой код получается при опросе контактов переключателей (клавиш), присоединенных к выводам порта B. Наличие 0 в определенном разряде кода будет соответствовать замкнутому контакту.

Задание. Выполните разработку программы в следующей последовательности:

1 С помощью команды **New Project…** запустите New Project Wizard. Создайте новый проект с именем switch в папке f:\....\Lab3. Выберите микроконтроллер PIC16F877, тактовая частота 8 МГц.

2 Наберите текст исходной программы switch.c и сохраните его в папке (с помощью команды меню **File > Save**).

3 Выполните построение проекта (с помощью команды меню **Build > Build**).

4 При отсутствии ошибок построения проекта переключите mikroC PRO в режим отладки (с помощью команды меню **Run > Start Debugger**).

Проведите исследование работы программы switch.c в следующей последовательности.

Для тестирования программы занесите в окно наблюдения Watch Values последовательно имена портов PORTB и PORTC, а затем имя переменной in_port.

Для исследования работы программы измените формат отображения всех переменных в окне наблюдения на шестнадцатеричный.

Выполните программу switch.c в пошаговом режиме Step Over путем нажатия на клавишу F8, или щелкая мышью по соответствующему значку управления отладкой.

Проверьте работу программы для всех четырех вариантов кода на входах порта B, указанных в таблице. В заключение установите на входах порта B код 0xFF.

Объясните полученные результаты выполнения операторов программы.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**.

В заключение закройте проект с помощью команды **Project > Close Project**.

3.4 Задания для самостоятельной работы

Задание 1 Напишите программу для PIC16F877, в которой в бесконечном цикле вводится байт данных из порта C. Если он меньше 100, то он передается в порт B, иначе он передается в порт D. Разработанную программу назовите vvod.c.

Для исследования программы создайте проект с именем vvod в папке f:\...\Lab3. Выберите PIC16F877, частота 8 МГц. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования разработанной программы занесите в окно наблюдения Watch Values имена портов PORTB, PORTC и PORTD, а также имя переменной для хранения введенного числа.

Для исследования работы программы vvod.c выполните ее в пошаговом режиме Step Over. Убедитесь в правильности выполнения алгоритма программы.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**.

В заключение закройте проект с помощью команды **Project > Close Project**.

Задание 2 Разработайте программу, которая определяет большее из трех введенных чисел. В программе числа вводятся последовательно из порта B микроконтроллера PIC16F877: сначала – первое, затем – второе, а потом – третье. Для хранения каждого введенного числа надо предусмотреть отдельную переменную. Найденное большее число выводится в порт C. Разработанную программу назовите max.c.

Для исследования программы создайте проект с именем max в папке f:\...\Lab3. Выберите PIC16F877, частота 8 МГц. Выполните построение проекта, а затем перейдите в режим отладки.

Для тестирования разработанной программы занесите в окно наблюдения Watch Values имена портов PORTB и PORTC, а также имена переменных для хранения введенных чисел.

Для исследования работы программы max.c выполните ее в пошаговом режиме Step Over. Убедитесь в правильности выполнения алгоритма программы.

Для завершения работы с отладчиком выполните команду меню **Run > Stop Debugger**.

В заключение закройте проект с помощью команды **Project > Close Project**.

4 Содержание отчета

Укажите наименование и цель работы. Представьте тексты всех исследуемых программ, включая задания для самостоятельной работы (комментарии в программах обязательны!).

5 Контрольные вопросы

1 Как производится настройка линий портов PIC-микроконтроллеров на ввод?

2 Как производится настройка линий портов PIC-микроконтроллеров на вывод?

3 Какие операторы выбора имеются в языке Си?

4 В каких случаях оператор switch не может использоваться вместо оператора if … else if … else?

5 Какие функции выполняет оператор break в конструкции с оператором switch?

6 Может ли в операторе switch отсутствовать ветвь default? Как в этом случае выполняется программа?

Лабораторная работа № 4

ИССЛЕДОВАНИЕ ОПЕРАТОРОВ ЦИКЛА И ПЕРЕХОДА ДЛЯ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ В ЯЗЫКЕ СИ

1 Цель работы

Изучить операторы цикла и перехода для управления вычислительным процессом в языке программирования Си. Практически исследовать выполнение этих операторов в программах для микроконтроллеров семейства PIC16.

2 Основные теоретические сведения

Операторы управления вычислительным процессом позволяют выполнять ветвление, циклическое повторение одного или нескольких операторов, передачу управления в нужное место кода программы.

Язык Си предоставляет три категории операторов управления программой:

1) операторы выбора – это if и switch;

2) операторы цикла – это while, do … while и for;

3) операторы перехода – это break, continue и goto.

В данной лабораторной работе Вы будете изучать и исследовать операторы цикла и перехода.

3 Порядок выполнения работы

3.1 Создание папки для работы в IDE

При выполнении лабораторной работы № 4 мы будем использовать папку с именем Lab4, которую необходимо предварительно создать.

С этой целью откройте Вашу рабочую папку и создайте в ней (с помощью клавиши F7) новую папку с именем Lab4.

В дальнейшем Вы будете записывать и хранить в этой папке все файлы при выполнении лабораторной работы № 4. Полный путь к этой папке будет такой:

f:\MPinCS\Ivanov\Lab4

3.2 Оператор цикла for

Цикл for является универсальным, поскольку компоненты цикла могут быть произвольными выражениями. Общая форма записи оператора цикла for имеет вид

```
for (инициализация; условие; увеличение/уменьшение переменной цикла)
{
    тело цикла;
}
```

Оператор for имеет три компонента:

1 Инициализация – это место, где обычно находится оператор присваивания, используемый для установки начального значения переменной цикла.

2 Условие – это место, где находится выражение, определяющее условие работы цикла.

3 Увеличение/уменьшение переменной цикла – это место, где определяется характер изменения переменной цикла на каждой итерации, т. е. повторения цикла.

Тело цикла – это оператор или группа операторов, которые будут выполняться в цикле. Цикл for работает до тех пор, пока условие истинно. Когда условие становится ложным, выполнение программы продолжается с оператора, следующего за циклом for.

Например, в следующей программе осуществляется вывод чисел от 1 до 100 включительно в порт С микроконтроллера:

```
void  main( )
{
    char  x;
    for (x = 1; x <= 100; x++)
        PORTC = x;
}
```

В программе переменная x изначально установлена в 1. Поскольку x меньше 100, выполняется оператор PORTC = x, который выводит x в порт С. После этого x увеличивается на 1 и проверяется условие: по-прежнему ли x меньше либо равен 100. Данный процесс продолжается до тех пор, пока x не станет больше 100, и в этот мо-

мент цикл прервется. В данном примере x является переменной цикла, которая изменяется и проверяется на каждой итерации цикла.

Вариации цикла for. Важной особенностью цикла for является то, что все три компонента цикла являются необязательными.

Например, если оставить все три компонента пустыми, то получим бесконечный цикл:

```
for (;  ;);
```

Кроме того, тело цикла может быть пустым, т. е. не содержать операторов. Такие циклы могут использоваться для получения временных задержек при выполнении программы:

```
for (i = 0;  i < 10000; i++);
```

3.3 Оператор цикла while

Цикл while является разновидностью условного цикла, повторяющегося до тех пор, пока условие выполнено. Таким образом, цикл while может не выполняться ни разу, если условие проверки изначально ложно. Форма записи оператора цикла while имеет вид

```
while (условие)
{
    блок операторов;
}
```

Пример. Следующий цикл выводит в порт С числа от 1 до 100:

```
int  x = 1;
while (x < = 100)
{
    PORTC = x;
    x++;
}
```

3.4 Оператор цикла do…while

В цикле do… while условие повторения проверяется после каждого прохождения тела цикла. Следовательно, цикл do… while выполняется, по крайней мере, один раз.

Форма записи цикла do… while имеет вид

```
do
{
     блок операторов;
}
while (условие);
```

Пример. Следующий цикл выводит в порт С значения квадратов чисел от 2 до 10:

```
int  x = 2;
do
{
    PORTC = x * x;
    x++;
}
while (x < = 10);
```

3.5 Оператор перехода break

Оператор break имеет два назначения. Первое – это окончание работы оператора switch. Второе – это принудительное окончание цикла, минуя стандартную проверку условия. Когда оператор break встречается в теле цикла, цикл немедленно заканчивается и выполнение программы переходит на строку, следующую за циклом.

Рассмотрим пример использования оператора break для досрочного прекращения работы цикла for:

```
char  in_port,  x;
void  main( )
{
   TRISB = 0xFF;                    // настроить порт B на ввод
   TRISC = 0;                       // настроить порт C на вывод
   PORTC = 0;
   in_port = PORTB;                 // ввод числа из порта B
   for (x = 1; x < 100; x++)
   {
       if (x == in_port)
```

```
            break;                          // прекращение цикла вывода
          PORTC = x;                        // вывод в порт С
    }
}
```

В этой программе оператор цикла for выполняет вывод в порт С переменной x, изменяющейся от 0 до 100. Однако, если значение переменной x будет равно переменной in_port, введенной из порта В, причем это значение меньше 100, то цикл вывода немедленно прекращается.

3.6 Оператор перехода continue

Работа оператора continue чем-то похожа на работу оператора break. Но вместо форсированного окончания цикла оператор continue переходит к следующей итерации цикла, пропуская оставшийся код тела цикла.

Рассмотрим пример применения оператора continue для вывода в порт С чисел от 1 до 10 за исключением числа 5:

```
void  main( )
{
    char  k;                        // объявление однобайтной переменной
    TRISC = 0;                      // настроить порт С на вывод
    PORTC = 0;                      // вывод нулей в порт С
    for (k = 1; k <= 10; k++)
    {
        if (k == 5)                 // если k равно 5, то
            continue;               // пропустить вывод в порт С
        PORTC = k;                  // вывод в порт С
    }
}
```

3.7 Оператор перехода goto

Оператор goto (идти к ….) – это безусловный переход на метку. Метка – это идентификатор Си, завершающийся двоеточием. Пример записи оператора goto:

```
   goto  label;
        ……
        …….
   label: …….
```

Рассмотрим простейшую программу, где используется оператор перехода goto:

```
void main( )
{
    char  x = 1;
    loop:
        x++;
        if (x <= 10)
            goto  loop;
}
```

В программе реализуется цикл инкремента переменной x от 1 до 10 с помощью оператора goto и метки loop. Очевидно, что такой цикл можно было бы реализовать и с использованием операторов while или for.

3.8 Задание для самостоятельной работы

Разработайте все варианты программы, которая находит сумму целых чисел от 1 до 15 включительно и выводит полученное значение в порт С микроконтроллера PIC16F877.

Вариант 1 Используйте в программе для организации цикла оператор for. Программу назовите for.c. Создайте проект с именем for в папке f:\...\Lab4, выполните его построение и произведите тестирование программы.

Примечание. Для того чтобы при отладке программы можно было занести переменные в окно наблюдения, их нужно объявлять в исходной программе как глобальные, т. е. до функции main( ).

Вариант 2 Используйте для организации цикла суммирования оператор while. Программу назовите while.c. Создайте проект с именем while в папке f:\...\Lab4, выполните его построение и произведите тестирование программы.

Вариант 3 Используйте для организации цикла суммирования оператор do…while. Программу назовите dowhile.c. Создайте проект с именем dowhile в папке f:\...\Lab4, выполните его построение и произведите тестирование программы.

Вариант 4 Используйте для организации цикла суммирования оператор for (как в варианте 1). Однако при достижении значения числа, равному 10, цикл суммирования должен немедленно прекратиться. Для этой цели примените оператор перехода break. Програм-

му назовите break.c. Создайте проект с именем break в папке f:\...\Lab4, выполните его построение и произведите тестирование программы.

Вариант 5 Используйте для организации цикла суммирования оператор for (как в варианте 1). Однако для суммирования должны использоваться только четные значения чисел, т. е. 2, 4, 6 и т. п. Для пропуска суммирования нечетных значений чисел используйте оператор перехода continue. Для проверки целого числа, например, x, на четность можно использовать арифметическую операцию x % 2 – определение остатка от деления на 2. Если результат операции будет равен 0, то число x – четное, если же результат будет 1, то x – нечетное. Программу назовите continue.c. Создайте проект с именем continue в папке f:\...\Lab4, выполните его построение и произведите тестирование программы.

Вариант 6 Используйте для организации цикла суммирования чисел от 1 до 15 оператор перехода goto. Программу назовите goto.c. Создайте проект с именем goto в папке f:\...\Lab4, выполните его построение и произведите тестирование программы.

4 Содержание отчета

Укажите наименование и цель работы. Представьте тексты всех вариантов программ к заданию для самостоятельной работы (комментарии в тексте программ обязательны!).

5 Контрольные вопросы

1 Какие операторы цикла имеются в языке Си?

2 Какие операторы перехода имеются в языке Си?

3 Чем отличаются операторы цикла while и do … while?

4 Какими способами можно реализовать бесконечный цикл?

5 В каких случаях удобно использовать оператор break?

6 Чем отличается действие оператора continue от действия оператора break?

7 Как можно применять оператор goto?

Литература

1 Шпак, Ю. А. Программирование на языке С для AVR и PIC микроконтроллеров / Ю. А. Шпак. – К. : МК-Пресс ; СПб. : КОРОНА-ВЕК, 2011. – 550 с.

2 Уилмсхерст, Т. Разработка встроенных систем с помощью микроконтроллеров PIC. Принципы и практические примеры / Т. Уилмсхерст ; пер. с англ. – К. : МК-Пресс ; СПб. : КОРОНА-ВЕК, 2008. – 544 с.

3 Компиляторы для PIC-контроллеров. – 2017. – Режим доступа: http://www.microchip.ru.

4 MikroC PRO for PIC. User's manual. – 2017. – Режим доступа: http://www.mikroe.com.

Содержание

Учебное электронное издание комбинированного распространения

Учебное издание

Виноградов Эдуард Михайлович

ПРОГРАММИРОВАНИЕ PIC-КОНТРОЛЛЕРОВ НА ЯЗЫКЕ СИ

Практикум по выполнению лабораторных работ по дисциплине «Микропроцессоры в системах управления» для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения

Электронный аналог печатного издания

Редактор *Т. Н. Мисюрова*
Компьютерная верстка *Н. Б. Козловская*

Подписано в печать 19.09.18.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Цифровая печать. Усл. печ. л. 3,49. Уч.-изд. л. 3,80.
Изд. № 5.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
пр. Октября, 48, 246746, г. Гомель