ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного
бюджетного образовательного учреждения высшего образования
"Московский технический университет связи и информатики"

Методические указания
к лабораторным работам

МИКРОКОНТРОЛЛЕРЫ

Среда программирования и отладки

Направление подготовки:
09.03.01 Информатика и вычислительная техника
11.03.02 Инфокоммуникационные технологии и системы связи

Ростов-на-Дону
2019

УДК 681.3.06 (076)
ББК 32.07

Чикалов А.Н. Микроконтроллеры. Среда программирования и отладки. Методические указания к лабораторным работам. Ростов-на-Дону: Северо-Кавказский филиал МТУСИ, 2019.- 55 с.

В пособии изложены методические рекомендации, содержательные материалы и контрольные задания для проведения лабораторных и практических занятий по изучению приемов и методов программирования микроконтроллеров семейства AVR, принципов трансляции и отладки программного обеспечения. В качестве инструмента использована среда разработки AVR Studio и среда моделирования электронных схем Proteus. Пособие содержит необходимые справочные материалы.

Методические указания предназначены для студентов, обучающихся по направлениям подготовки 09.03.01 Информатика и вычислительная техника и 11.03.02 Инфокоммуникационные технологии и системы связи, профилей Многоканальные телекоммуникационные системы, Сети связи и системы коммутации, Защищенные системы и сети связи, Системы радиосвязи и радиодоступа, Вычислительные машины, комплексы, системы и сети, Программное обеспечение и интеллектуальные системы.

Пособие предназначено для использования при изучении дисциплин Специализированные процессоры, Микропроцессорные системы, Вычислительная техника и информационные технологии, а также может быть использовано преподавателями и студентами при изучении родственных дисциплин и в процессе самостоятельной работы.

Учебное пособие обсуждено и одобрено на заседании кафедры ИВТ.
Протокол №1 от 26.08.2019 г.

Рецензент Зав. кафедрой ИВТ д.т.н. профессор Соколов С.В.

СОДЕРЖАНИЕ

1. ИССЛЕДОВАНИЕ СРЕДЫ ПРОГРАММИРОВАНИЯ AVR Studio СПЕЦИАЛИЗИРОВАННОГО КОНТРОЛЛЕРА

Цель:

1. Углубить и закрепить теоретические знания по принципам программирования и отладки в интегрированной среде;

2. Приобрести практические навыки самостоятельного изучения состава, возможностей, режимов работы среды программирования AVR Studio, освоения элементов интерфейса;

3. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы:

1.1. Назначение и основные компоненты среды AVR Studio;
1.2. Текстовый редактор исходных текстов программ;
1.3. Компилятор языка ассемблера.

Литература для подготовки к занятию

1. Белов А.В. Разработка устройств на микроконтроллерах AVR: шагаем от "чайника" до профи. - СПб.: Наука и техника, 2013. -528 с. (стр.289-306).
2. Евстифееев А.В. Микроконтроллеры AVR семейства Tiny. Руководство пользователя. М.: Издательский дом №Додэка-XXI", 2007. -432 с.
3. WWW.kit-e.ru Компоненты и технологии. AVR: программирование в среде AVR Studio/

Содержание отчета

1. Название работы.
2. Название каждого учебного вопроса и краткий конспект в объеме практических заданий по вопросу.

Вопросы для подготовки к занятию

1. Назначение транслятора, компилятора, интерпретатора.
2. Техника работы компилятора с языка ассемблера.
3. Отличие текстовых редакторов систем программирования от классических текстовых редакторов.
4. Какая среда называется интегрированной?

Актуальность занятия

Разработка устройств на основе специализированных контроллеров связана с разработкой аппаратной и программной составляющих. Кроме того,

это связано с особенностями разработки систем реального времени. Поэтому все этапы проектирования должны быть максимально автоматизированы. Этим требованиям в полной мере отвечает среда AVR Studio. Она является признанным продуктом и ее освоение может стать основой для освоения других аналогичных систем.

Задание 1.1. Назначение и основные компоненты среды AVR Studio

Практические задачи:

1. Выполнить запуск и настройку среды разработки по приведенному руководству;

2. Научиться открывать все окна и вкладки;

3. Кратко записать состав имеющихся меню, уяснить их назначение.

IDE AVR_Studio — это интегрированная среда разработки (IDE – Integrated Development Environment) приложений для всех типов микроконтроллеров семейства AVR (AT90S, ATmega, ATtiny). Свободно распространяемый продукт фирмы Atmel (Atmel.com). Среда имеет простой и интуитивно-понятный интерфейс (рис.1.1).

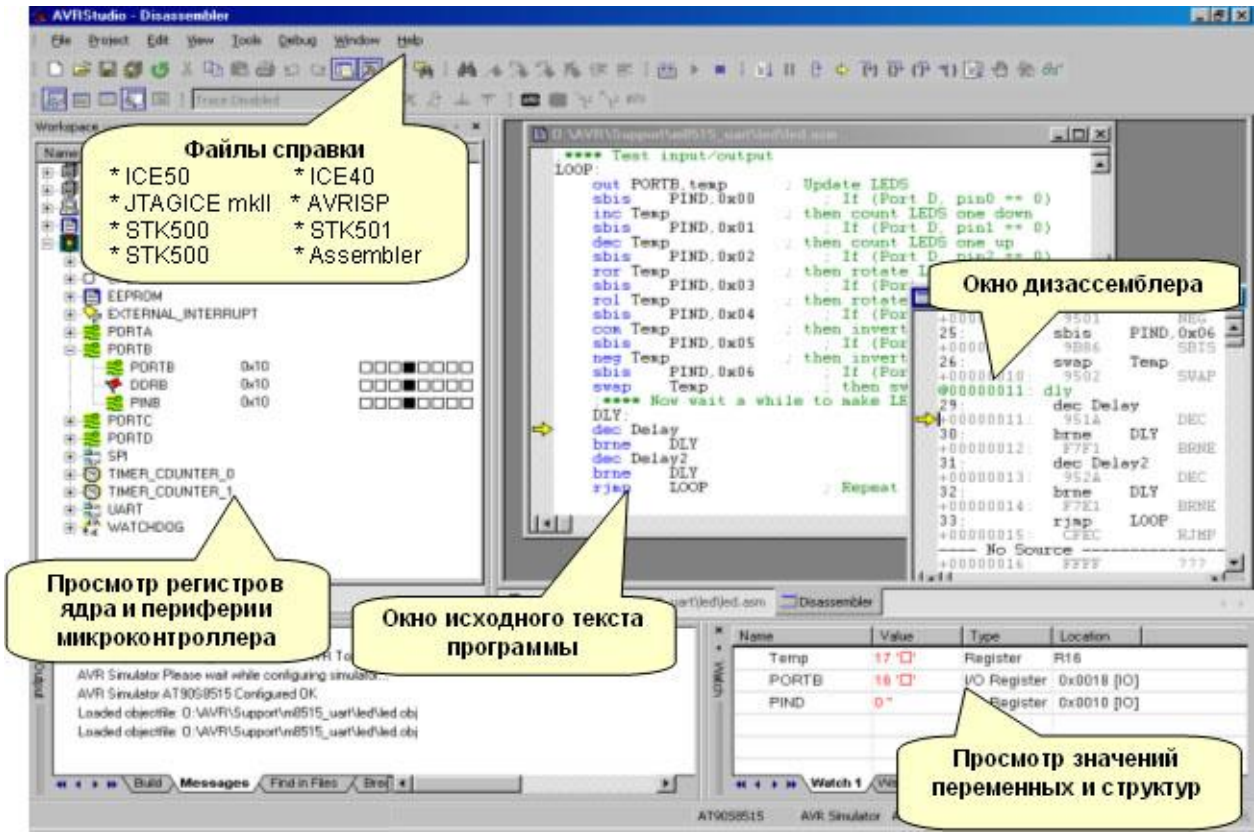


Рис.1.1. Интерфейс интегрированной среды разработки AVR Studio

IDE AVR Studio содержит:

- средства создания и управления проектом;
- текстовый редактор исходных текстов программ;
- транслятор языка ассемблера (Atmel AVR macroassembler);
- отладчик (Debugger);
- симулятор (AVR Simulator);
- загрузчик машинного кода в память МК с поддержкой внутрисхемного программирования (In-System Programming, ISP) с использованием стандартных отладочных средств Atmel AVR;
- встроенную справочную систему (Help), содержащую описание языка ассемблер и системы команд МК серии AVR и др. компоненты.

Отладчик AVR Studio поддерживает все типы микроконтроллеров AVR и имеет два режима работы: режим программной симуляции и режим управления различными типами внутрисхемных эмуляторов (In-Circuit Emulators) производства фирмы Atmel. В окне отображается код, который выполняется в отладочном окружении в настоящий момент. Внутрисхемный эмулятор позволяет производить отладку приложения на реальной целевой плате. В режиме реального времени эмулятор работает существенно быстрее, чем программный симулятор. Интерфейс пользователя не изменяется в зависимости от выбранного режима отладки.

Отладочная среда поддерживает выполнение программ, как в виде ассемблерного текста, так и в виде исходного текста языка C/C++. В последнем случае загружаемый модуль программы должен содержать не только машинный код, но и исходный текст программы, привязанный к машинному коду.

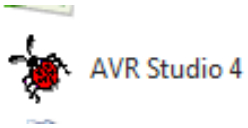
Программная среда работает с проектами. Загружен одновременно может быть только один из них. Каждый проект должен иметь на диске свой каталог. Он содержит всю информацию о разрабатываемой программе и применяемом микроконтроллере. Состоит проект из набора файлов. Основные из них следующие:

- файл проекта (расширение .aps) – содержит содержит имена всех исходных файлов, связанных с проектом, а также установки компиляции, трансляции и связывания файлов для генерации выполняемой программы. Это основной файл проекта. Он создается в процессе работы и потом все его настройки используется при "открытии" файла проекта;
- один или несколько файлов исходного текста программы (расширение .asm). Один из них является главным, остальные могут вызываться из него с помощью оператора подключения .include. По своей структуре это текстовые файлы;
- файл кодов трансляции (расширение .hex). Он появляется после трансляции и содержит двоичные коды машинных команд в текстовом формате. Этот файл используется для прошивки памяти микроконтроллера;

- файл символьных имен (расширение .map). Это текстовый файл, в котором всем именам приведены в соответствие их значения;
- листинг трансляции (расширение .lst) и т.д.

В этом пособии рассматривается AVR Studio 4.19. Это не последняя версия пакета, но она зарекомендовала себя как стабильная, малозатратная по машинным ресурсам, простая и достаточно функциональная программа.

В качестве объекта программирования будет рассматриваться микроконтроллер семейства Tiny - ATtiny 2313. Он является средним по сложности, близок по функциональным возможностям к мощным контроллерам семейства Mega, очень хорошо и широко описан в доступной литературе.



Для запуска программы используется файл AVRStudio.exe. После стандартной установки он находится в папке C:\Program Files\Atmel\AVR Tools\AVR Studio4. В результате запуска появится основное диалоговое окно программы, показанное на рис.1.2.

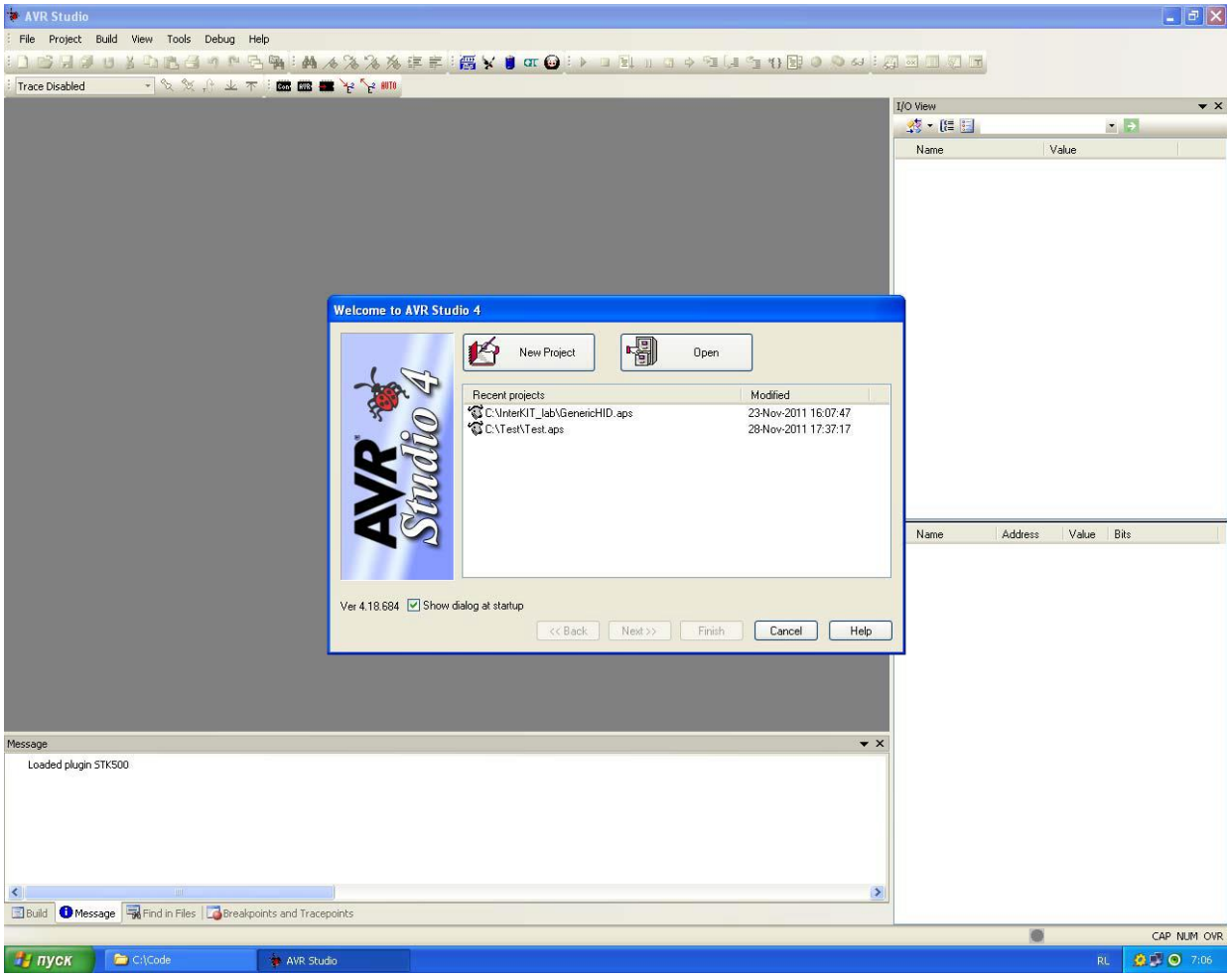


Рис.1.2. Окно программы AVR Studio в начале работы над проектом

В появившемся окне предлагается либо создать новый проект, либо открыть уже существующий.

При установленном флаге ***Show this dialog on open*** (Показывать это окно при открытии), расположенным в левом углу, окно будет появляться при каждом запуске программы. Если в течении одного сеанса необходимо работать с разными проектами, щелкнув ***Cancel*** можно закрыть это окно, а новый проект можно создавать, пользуясь меню AVR Studio

Для создания нового проекта щелкните кнопку ***New Project***. Появится окно, показанное на рис.1.3.

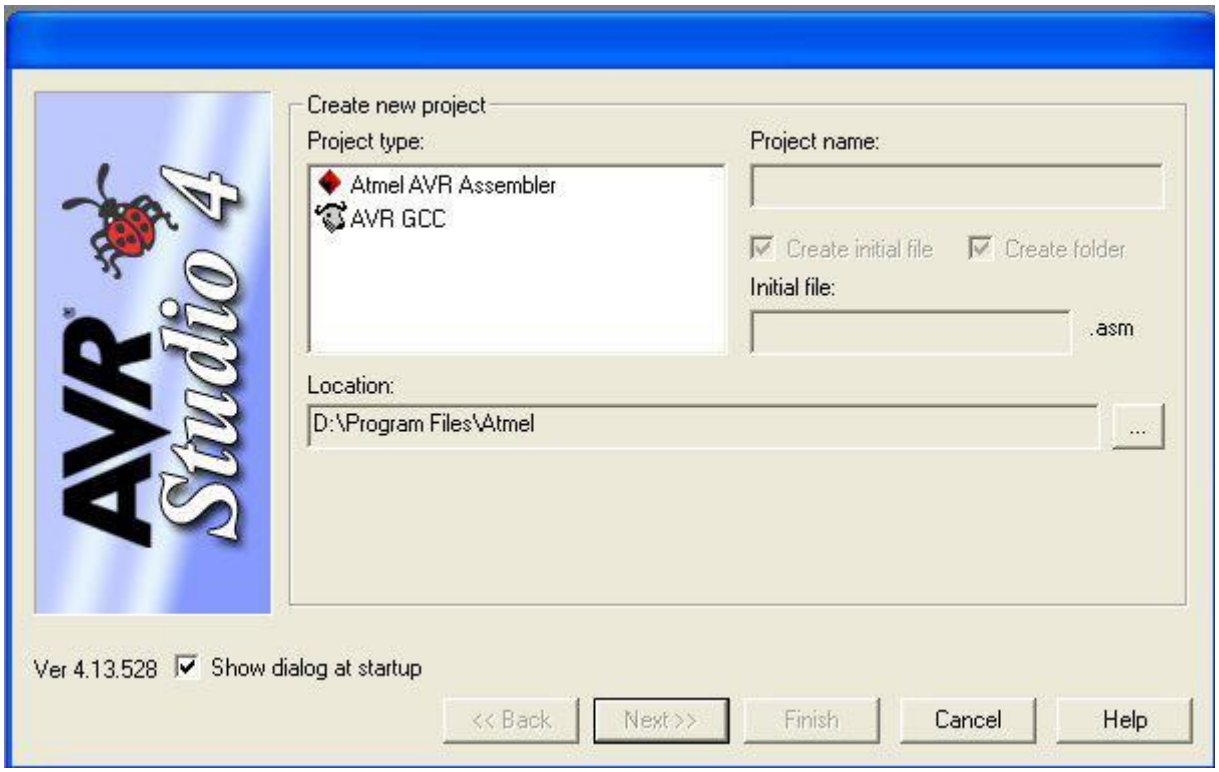


Рис.1.3. Окно создания нового проекта

В левой части окна (***Project type***) выберите тип проекта ***AtmelAVR Assembler.*** В этом случае AVR Studio использует для компиляции проекта программу Ассемблер. Никаких дополнительных действий пользователя больше не потребуется.

В случае выбора ***AVR GCC*** предполагается, что текст программы будет написан на языке С. В таком случае следует дополнительно конфигурировать AVR Studio.

Введите имя проекта, например, "Project_1" (обязательно латинскими буквами без пробелов) в строку ввода ***Project name***. Если установлена опция ***Create initial file***, то это имя автоматически копируется в строку ***Initial file*** и

по умолчанию будет присвоено файлу, который будет содержать программу на ассемблере (project_1.asm).

Желательно каждый проект создавать в отдельной папке для удобства дальнейшего использования всех файлов проекта. При установленной опции ***Create folder*** (Создать папку) можно здесь же организовать папку, в которой будет находиться проект. Для этого щелкните кнопку ***Location*** и выберите место положения проекта. Располагать его лучше как можно ближе к корневому каталогу. В результате получится окно, показанное на рис.1.4.

Для открытия уже существующего проекта после запуска AVR Studio в окне, показанном на рис.1.7 выберите опцию ***Open File.***

Для создания нового проекта откройте меню ***Project/ New Project***. На экране появится окно ***Create new project*** (см. рис.1.3).

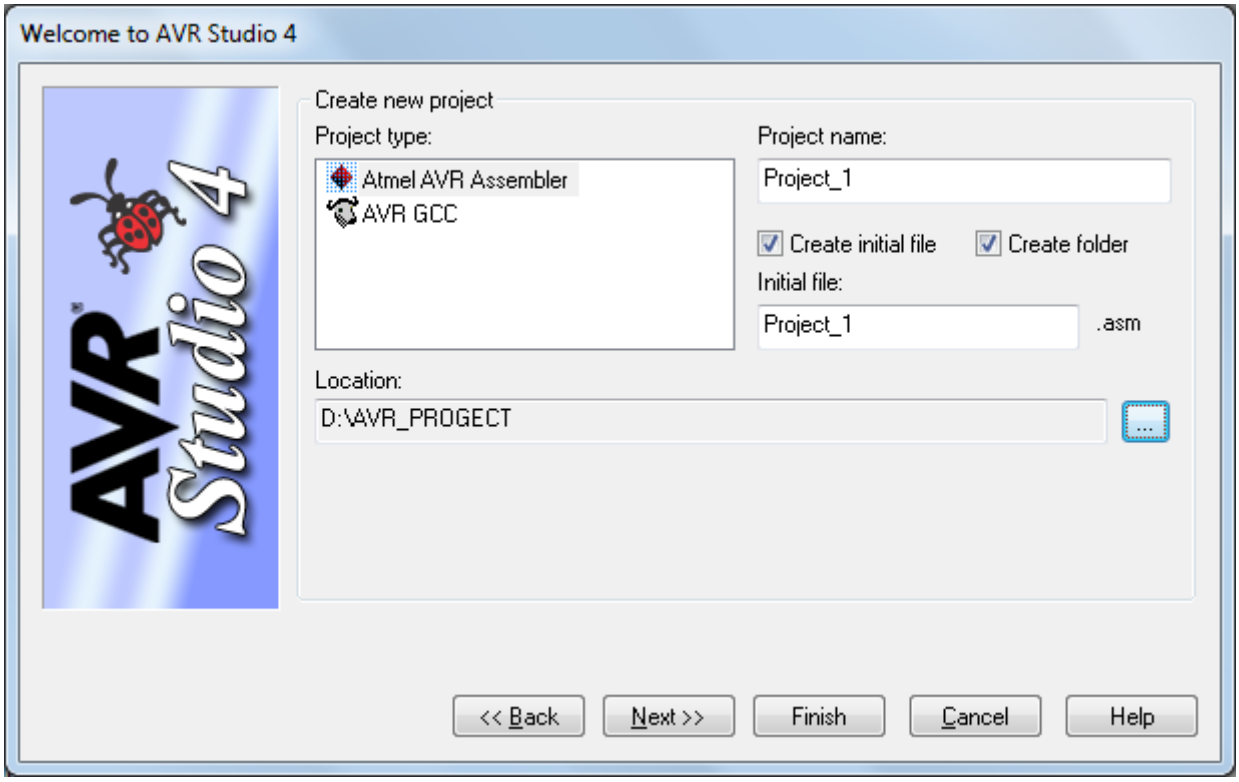


Рис.1.4. Вкладка AVR Studio с введенными именами проекта, места его размещения и именем исходного файла с кодом программы

После выбора всех опций щелкните кнопку ***Next.*** Появится окно, показанное на рис.1.5.

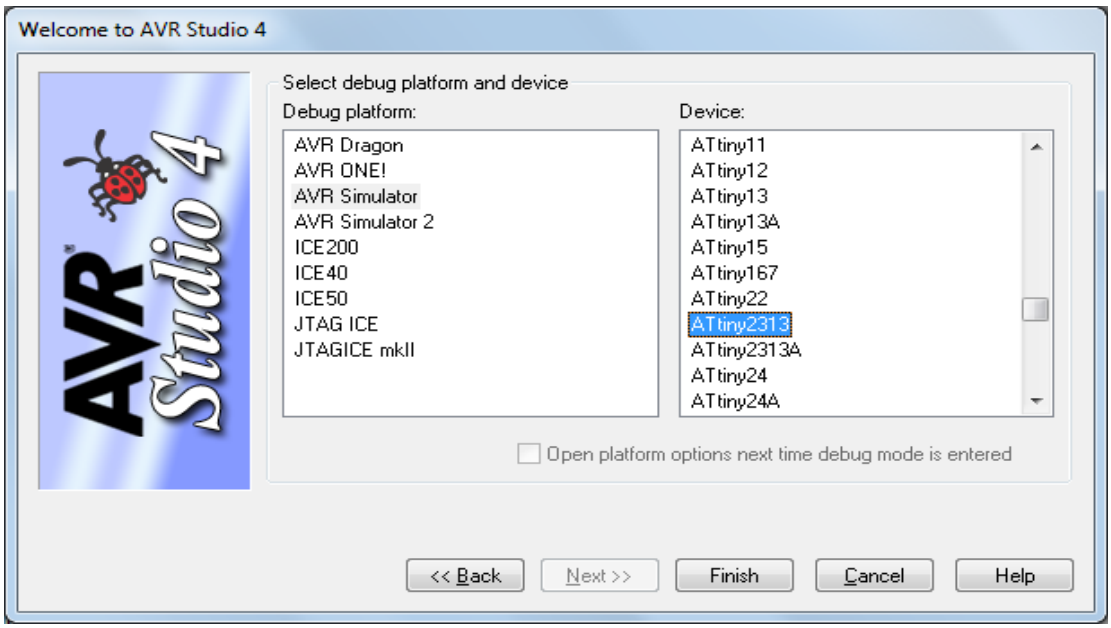


Рис.1.5. Вкладка определения типа процессора и платформы для отладки

В левой части окна выберете ***AVR Simulator***. Это программная виртуальная среда. В правой части окна выберете тип микроконтроллера (в данном случае ***ATtiny2313***) и щелкните ***Finish.*** В результате организуется основное рабочее окно среды разработчика AVR Studio (рис.1.6).

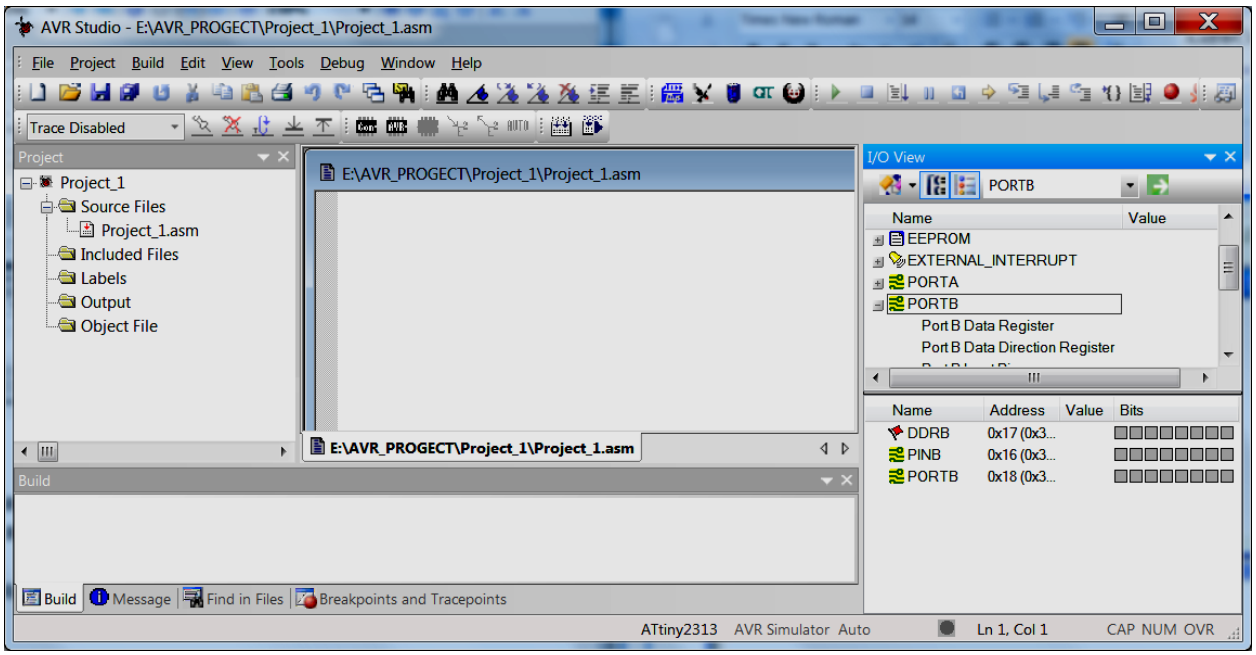


Рис.1.6. Основное окно среды разработки

Окно ***Project*** (Проект) содержит полную информацию о текущем проекте. Ветки представленного дерева описывают все исходные и результирующие файлы, метки, процедуры и присоединяемые файлы.

Заголовок ***текстового окна*** в главной области панели AVR Studio подписан именем основного файла исходного текста программы. Здесь открывается одно или несколько окон, содержащих текст программы на ассемблере. Каждое новое окно в этой области создает корешок в нижней части окна, по которому окно можно вывести на передний план. Здесь же можно открыть тексты других проектов на других языках программирования. Все окна запоминаются и автоматически открываются при открытии проекта.

Текстовые окна имеют подсветку. Операторы Ассемблера отображаются голубым цветом, комментарии - зеленым, параметры команд, метки, переменные и константы - черным. Такое решение позволяет контролировать процесс ввода и обнаруживать ошибки на этапе их совершения.

Окно просмотра ввода-вывода ***I/O View*** показывает все ресурсы микроконтроллера: РОН, порты, таймеры, АЦП, компараторы. Все сложные элементы можно нажатием крестика раскрыть в нижней части этого окна и увидеть состояние всех разрядов конкретного регистра раскрытого элемента, его адрес, название. В процессе отладки в этом окне отображается содержимое ресурсов в различных форматах, в том числе графическом путем затемнения тех разрядов, которые приобретают значение единицы. Используются элементы управления:

- переключатель режима показа - Taggle Presentation Mode;
- модульное разделенное представление - Module Split View;
- плоский вид регистров - Flat Register View;
- древовидное представление - Tree View;
- показать панель регистров - Show Registers Pane;
- показать панель модулей - Show Modules Pane.

Имеется возможность изменения конкретных битов двойным щелчком мышки.

Окно трансляции ***Build*** (строить, создавать) служит для вывода сообщений в процессе работы над проектом. По умолчанию оно имеет четыре вкладки:

- вкладка ***Build*** отображает процесс трансляции, показывая этапы трансляции, сообщения о синтаксических ошибках и различные предупреждения (***Warnings***), статистические данные при успешном завершении трансляции, затраты памяти микроконтроллера;
- вкладка системных сообщений ***Message*** показывает данные о загрузке модулей программы;
- вкладка поиска в файлах ***Find in Files*** позволяет выполнять поиск заданной последовательности символов сразу во всех файлах проекта. После поиска во вкладке отображаются все найденные вхождения с указанием имени файла и строки, где найдена искомая последовательность;
- вкладка точек останова и трассировки ***Breakpoints and Tracepoint*** позволяет дублировать точки в тексте программы и управлять ими при осуществлении отладки.

Любую из вкладок можно сделать скрытой (***Hide***), свободно перемещаемой (***Floating***) или закрепленной (***Docking***), щелкнув правой кнопкой по заголовку и выбрав пункт меню.

Окна могут быть легко изменены в размерах путем перетаскивания границ с помощью мыши. Их можно закрыть, используя крестик либо вызвав правой кнопкой пункт ***Hide*** на заголовке окна. Открывание закрытых вкладок осуществляется через меню ***View/Toolbars***.

Как и всякое Windows-приложение среда разработки имеет строку меню и панели инструментов.

Меню Файл ***File*** (рис.1.7) является стандартным и пояснений не требует.

Меню Проектов ***Project*** (рис.1.8) обеспечивает работу с проектами на различных этапах. Детализации требует только подменю Настройки Ассемблера ***Assembler Options*** (рис.1.9).

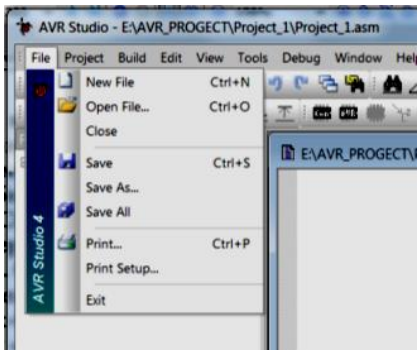


Рис.1.7. Меню File

Рис.1.8. Меню Project

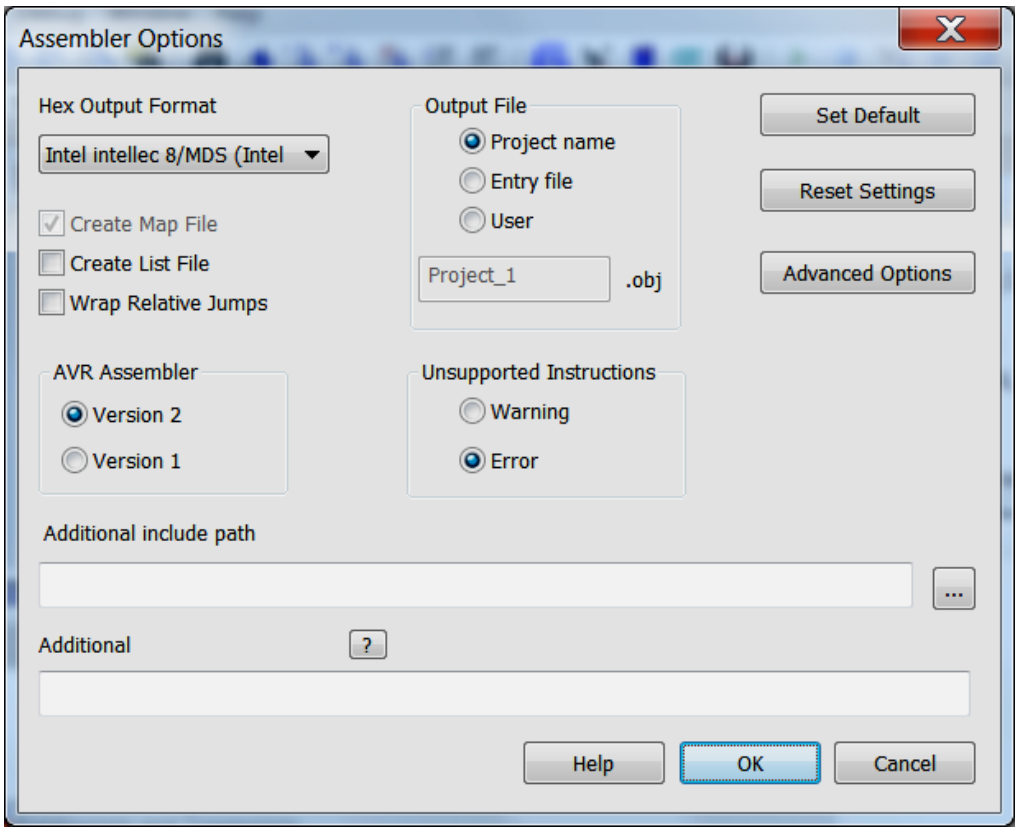


Рис.1.9. Настройки Аассемблера

Выпадающее окно ***Выходной Hex-формат*** (***Hex Output Format)*** позволяет выбрать тип файла прошивки. Наиболее популярен файл фирмы Intel. Его целесообразно и выбрать. Альтернативой им являются файлы от Motorola или Generic.

Настройка ***Создать листинг программы (Create List File)*** позволяет сформировать текстовый файл листинга с расширением ***.lst***.

Поле ***Выходного файла (Output File)*** дает возможность выбрать имя объектного файла с расширением .obj. Показанная на рис.1.9 точка позволяет сохранить за ним имя проекта. Эту настройку лучше сохранить по умолчанию.

Поле ***Неподдерживаемых команд (Unsupported Instructions)*** настраивает реакцию Ассемблера на те команды, которые не поддерживаются микроконтроллером. Необходимо выбрать "***Ошибка" (Error)***. В случае выбора "***Предупреждение"*** (***Warning)*** критических ошибок не будет, но и выполнение команды не произойдет.

Поле выбора типа ***Ассемблера*** (***AVR Assembler)*** позволяет выбрать тип транслятора. Версия 2 более совершенная.

Меню Трансляции ***Build*** назначает особенности использования транслятора (рис.1.10). Есть возможность провести трансляцию - ***Build (F7)*** или выполнить трансляцию с последующим запуском программы на выполнение в отладчике - ***Build and Run (Ctrl+F7).***

Меню Редактирования (***Edit***) в основном стандартно (рис.1.11). Особенными вкладками являются только некоторые:

- Расстановка закладок (***Toggle Bookmark***) - размещает закладки в тексте для обеспечения быстрого перехода. Этим пользуются при больших объемах текста, расставляя специальные овальные метки слева от текста. Командами Вперед (***Next***) и Назад (***Previous***) по ним перемещаются. Удаление закладки (***Remove Bookmark)*** - выполняет обратную операцию;

- Отображение знаков перемещения (***Show Whitespace***) - позволяет увидеть в окне символы табуляции и пробелов;

- Шрифт и цвет (***Font and Color***) - настраивает размер шрифта и подсветку синтаксиса. Этой возможностью нельзя злоупотреблять, т.к. пользователь привыкает к конкретной палитре.

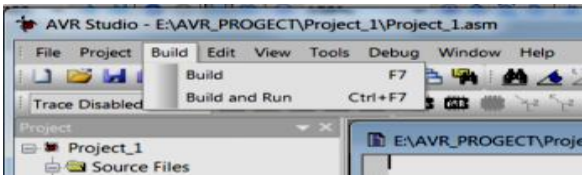


Рис.1.10. Меню Build

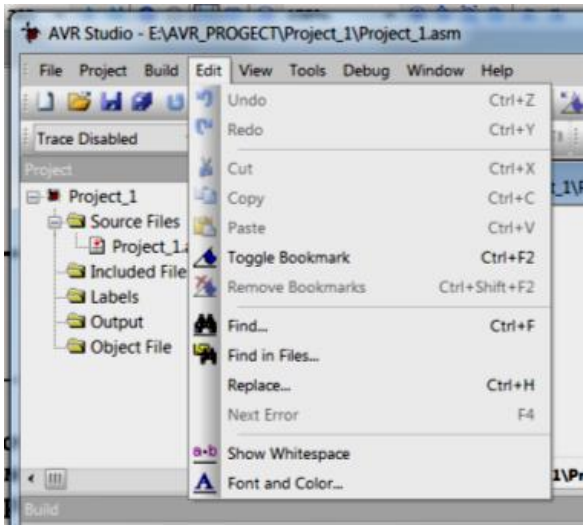


Рис.1.11. Меню Edit

Меню Вид (***View***) позволяет отобразить на экране различные окна и панели инструментов (рис.1.12). Чтобы их удалить, необходимо или нажать на крестик, или перетащить окно за пределы главного окна программы, или убрать галочку в этом меню.

Особый интерес представляет окно Проектов (***Project***). Оно структурирует все файлы проекта и позволяет упростить работу (рис.1.13). В окне отображаются папки:

- Исходных файлов (***Source File***). Здесь содержатся все ассемблерные тексты проекта;

- Присоединенных файлов (***Included Files***). Это текстовые ассемблерные файлы описания микроконтроллера, которые при трансляции присоединяются к исходным файлам;

- Меток (***Labels***). В этой папке собраны все метки проекта. Двойным щелчком по метке можно быстро перейти к ней в исходном файле;

- Выходных файлов (***Output***). Папка сохраняет файлы, которые получаются после трансляции. Это файлы прошивки (hex) и символьных имен (map);

- Объектных файлов (***Object File***). Эти файлы получаются после трансляции.

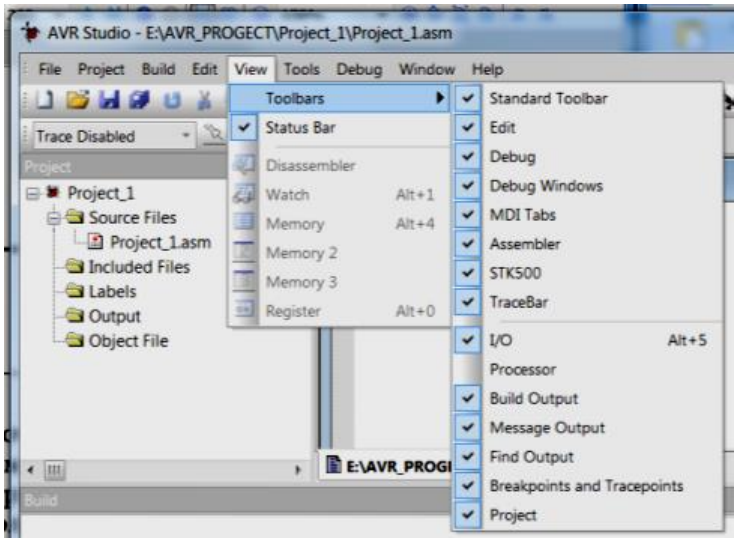


Рис.1.12. Меню View

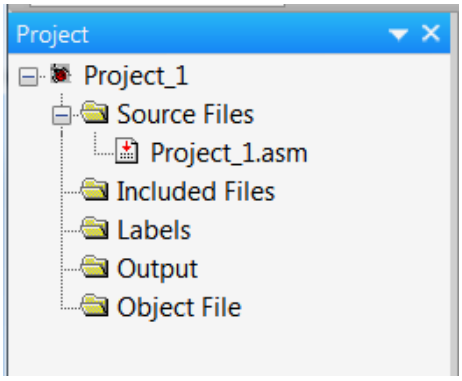


Рис.1.13. Окно Project

Меню Инструменты (***Tools***) позволяет настроить среду разработки и ее панели под конкретного пользователя (рис.1.14). Верхняя группа строк в меню позволяет обновлять отладочные платы. Нижняя - это утилиты и обновления AVR, которые широко не распространены. Часто используется строка Настройки (***Customize***). С ее помощью в закладке Команды (***Commands***) можно редактировать кнопки на панелях инструментов (рис.1.15) или управлять отображением окон и панелей в закладке Инструменты (***Toolbars***). Приемы работы с подобными окнами общеизвестны.

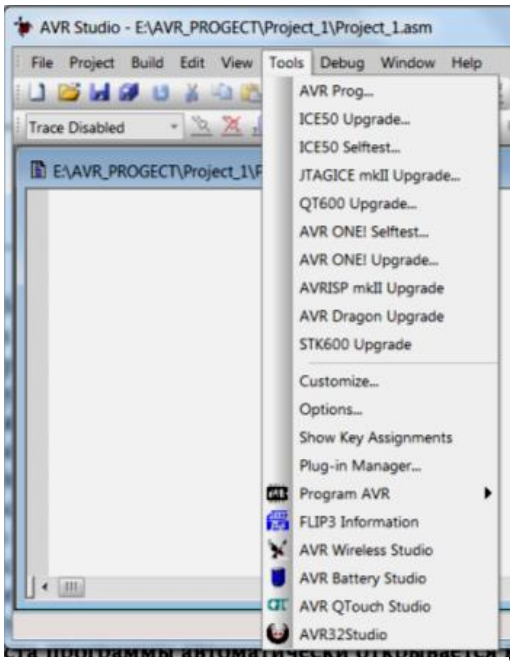


Рис.1.14. Меню Tools

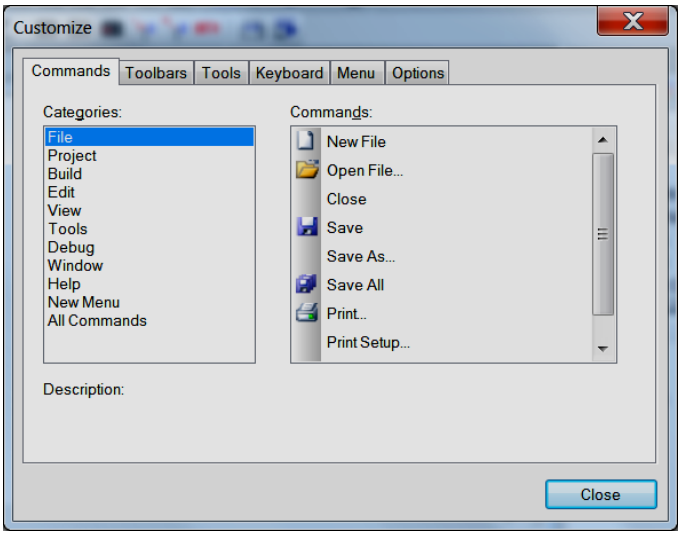


Рис.1.15. Окно настроек панели инструментов

Строка Свойства (***Options***) содержит основные настройки среды разработки (рис.1.16). В частности, окно ***Filetabs*** позволяет определить состав имени файла при отображении в окне. Выбирают, обычно, Только имя (***Filename only***) потому, что полный путь занимает слишком много места на экране.

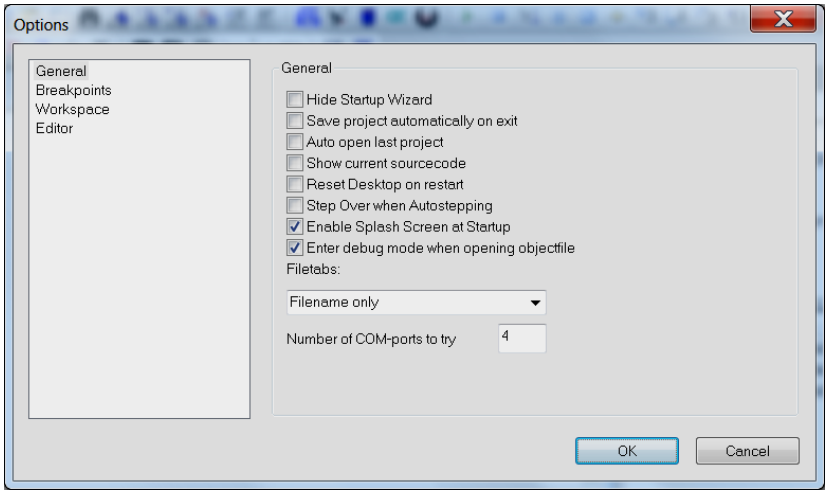


Рис.1.16. Параметры среды разработки

Меню Окон среды (***Window***) управляет настройкой окон (рис.1.17). Внизу списка меню показываются имена открытых файлов

Рис.1.17. Меню Window

Строки меню позволяют:

- расположить окна каскадом - строка ***Cascade***;
- разместить окна по горизонтали - строка ***Tile Horizontally***;
- по вертикали - строка ***Tile Vertically***;
- разделить поле окна на части командой "Расщепить" (***Split***). При этом в каждой части экрана текст перемещается самостоятельно. Границы можно свободно перемещать, а при необходимости удалить границы - просто перетащить за пределы экрана или дважды щелкнуть на самой границе. Функция полезна при работе с большими текстами программ для одновременной работы в разных местах.

В меню Помощи (***Help***) можно найти материал по различным средствам и этапам реализации проекта (рис.1.18):

- платы отладки и программирования AVR (***AVR Tools User Guide***);
- руководство пользователя среды разработки AVR Studio (***AVR Studio User Guide***);
- программа проверки наличия новых обновлений (***Check for Program Upgrade***);
- файл справки AVR Studio 4.19 (***Release Noted and Known Issues***);
- справка по языку Ассемблера (***Assembler Help***);
- характеристики среды программирования (***About AVR Studio***).

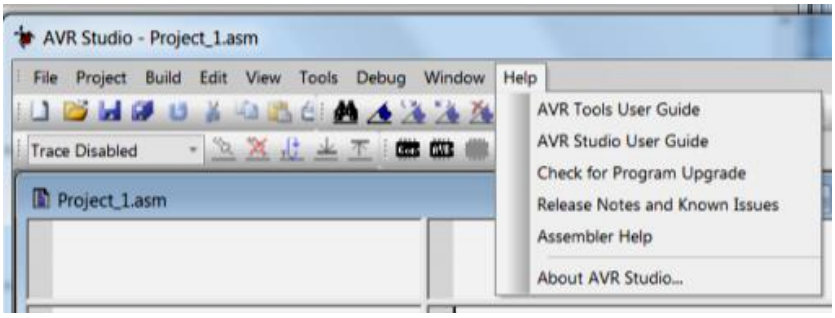


Рис.1.18. Меню Help

В отчете представить:

1. Последовательности действий для отображения окон:
   - регистров общего назначения;
   - устройств ввода-вывода;
   - параметров процессора;
   - отображения результатов трансляции;
   - системных сообщений;
   - установок точек останова;
2. Последовательности действий для выполнения:
   - запуска компиляции;
   - запуска и остановки выполнения программы;
   - установки типа процессора и платформы для отладки;
   - установки языка программирования.

Задание 1.2. Текстовый редактор исходных текстов программ

Практические задачи выполняются после усвоения материала и сформулированы в конце задания 1.2.

Текстовый редактор используется для формирования исходного текста программы. Программа для AVR на ассемблере имеет определенную структуру.

Первым элементом программы должно быть включение файла описаний для микроконтроллера. Осуществляется это директивой ассемблера

.include "tn2313def.inc"

Точка перед директивой обязательна. Имя файла должно соответствовать типу процессора. В этом файле находятся символические имена с присвоенными им числовыми значениями.

Сама программа пишется по общим правилам оформления ассемблерного текста, выравнивая записи по колонкам:

Метка:	Мнемокод	Параметры команды	; Комментарии
	Ldi	***R16, 255***	***; запись константы в R16***

Разметка текста осуществляется табуляцией для быстрого выравнивая полей записи по вертикали. Текстовый редактор нечувствителен к регистру, поэтому можно использовать как большие, так и маленькие буквы в любой комбинации.

В качестве примера выше записана команда записи константы 255 в регистр с номером R16.

Запись константы может быть сделана в разных форматах:

- в десятичном виде как в примере;

- в шестнадцатеричном формате - ***0xFF*** или ***0x0FF***. Незначащий нуль здесь не учитывается;

- в шестнадцатеричном формате - ***$FF*** или ***$0FF*** (называют "паскалевским" способом по аналогии с языком Паскаль. Здесь также незначащий нуль не учитывается. Но если в шестнадцатеричных форматах в числе символ только один, то нуль впереди целесообразно дописывать для повышения наглядности и уточнения размера регистра;

- в двоичном формате - ***0b11111111***. Здесь также из соображений наглядности целесообразно писать все восемь знаков, включая незначащие.

Все записи совершенно эквивалентны и различаются только формой. Выбор зависит от контекста и предпочтений программиста. Запись одних цифр с нулем впереди (например, 067) недопустима, потому что транслятор это воспринимает как восьмиричный формат и в памяти окажется число, соответствующее десятичной цифре 55.

Комментарии в строке отделяются от записи команды знаком "***;***" или "//". Многострочный комментарий начинается знаком "/*", а заканчивается - "*/".

Для работы в текстовом редакторе применяются типовые горячие клавиши:

End - переход в конец строки;

Ctrl+End - переход на начало текста;

Home - переход в начало строки;

Ctrl+Home - переход в конец текста;

Ctrl+Shift+→ - выделение текста пословно от места установки курсора. Обратная стрелка при той же комбинации - снимает выделение;

Shift+→ - выделение текста посимвольно. Стрелка в обратную сторону - снимает выделение;

Insert - включает/выключает режим замены (вставки). Текущий режим отображается в строке состояния аббревиатурой ***OVR***. Если установлен режим замены, то символы должны быть более темными. При отсутствии строки состояния на экране ее вызывают выполнением ***View/Status Bar.***

Регистр DDRB (адрес $17, регистр направления данных порта B - Data Direction Register Port B) задает направление данных 8-разрядного порта B. Установка единиц в соответствующие разряды обеспечивает для выводов режим выдачи данных, а запись нуля - в режим ввода.

Задание режима выдачи порта B требует записи всех единиц во все разряды. Это выполняется командой выдачи в порт. Напрямую константу в порт записать нельзя. Поэтому запись осуществляется через регистр R16:

```
OUT     DDRB,R16
```

Регистр PORTB (адрес $18, регистр данных порта B - Data Register Port B) хранит данные порта. При настройке порта на выход данные в PORTB будут выдаваться на выход. Для выдачи всех единиц необходимо выполнить

```
OUT     PORTB,R16
```

Эта совокупность команд выдает единицы на выход порта B. Если собрать схему, представленную на рис.1.19, то светодиод HL1 должен гореть, а HL2 - нет.

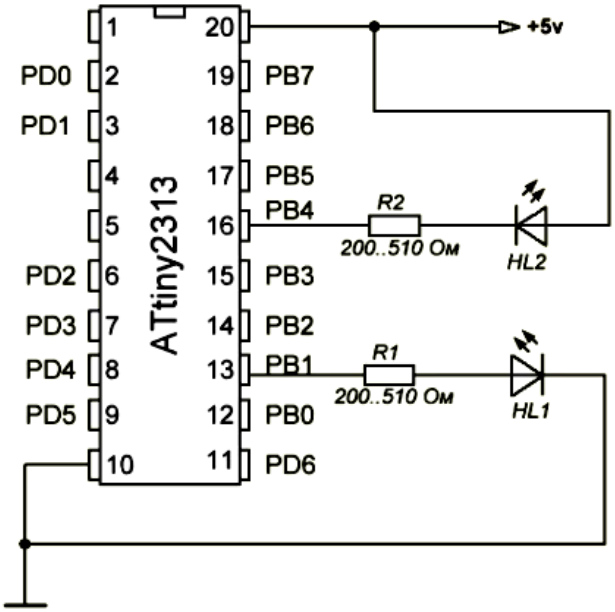


Рис.1.19. Схема для программирования микроконтроллера

Однако для корректности, чтобы процессор не перебирал все последующие свободные ячейки памяти, программу необходимо зациклить. Для этого используют команду относительного безусловного перехода. Метку можно в рассматриваемой программе установить в самое начало:

RJMP METKA

В таком виде текст программы (рис.1.20) является законченным и его можно сохранить. Сохранение исходного текста программы осуществляется в папке Source Files папки Project_1. Его расширение соответствует языку программирования. В данном случае это .asm. Дальнейшие процессы должны быть связаны с компиляцией программы.

Project_1.asm *

```
.include "tn2313def.inc"     //директива присоединения


Metka:  Ldi     R16,255      ;запись константы в R16
/* Эквивалентные записи команды:
        ldi     R16,$FF
        LdI     R16,0xFF
        lDI     R16,0b11111111
*/
        out     DDRB.R16     ;запись в порт DDRB числа из R16
                             ;это перевод разрядов порта B на выход
        out     PORTB,R16    ;единицы выданы на все разряды порта B
        rjmp    Metka
```

Рис.1.20. Текст программы управления портом B

Практические задачи:

1. Набрать в редакторе текст программы, представленный на рис.1.20;
2. Записать регистры, управляющие портом B, их символические имена и числовые адреса и их назначение;
3. Зарисовать схему подключения светодиодов.

В отчете представить:

1.Схему подключения светодиодов.
2. Текс программы.

Задание 1.3. Компилятор языка ассемблера

Практические задачи выполняются после усвоения материала и сформулированы в конце задания 1.3.

Для дальнейшей работы с симулятором необходимо произвести компиляцию программы, выбрав из меню ***Build*** *(строить)* пункт ***Build*** (путем нажатия горячей клавиши ***F7*** или кнопки ). При этом создается файл прошивки в машинных кодах с расширением .hex и некоторые другие файлы, используемые для работы с симулятором.

В окне под названием ***Build***, которое находится в нижней части окна среды разработчика нужно открыть закладку ***Build***. В случае успешной компиляции в ней должно быть сообщение: ***Assembly complete, errors 0, warnings 0*** и информация о размере программы. В данном примере имеют место две синтаксические ошибки, хотя точнее надо было бы говорить о двух сообщениях об ошибках (рис.1.21).

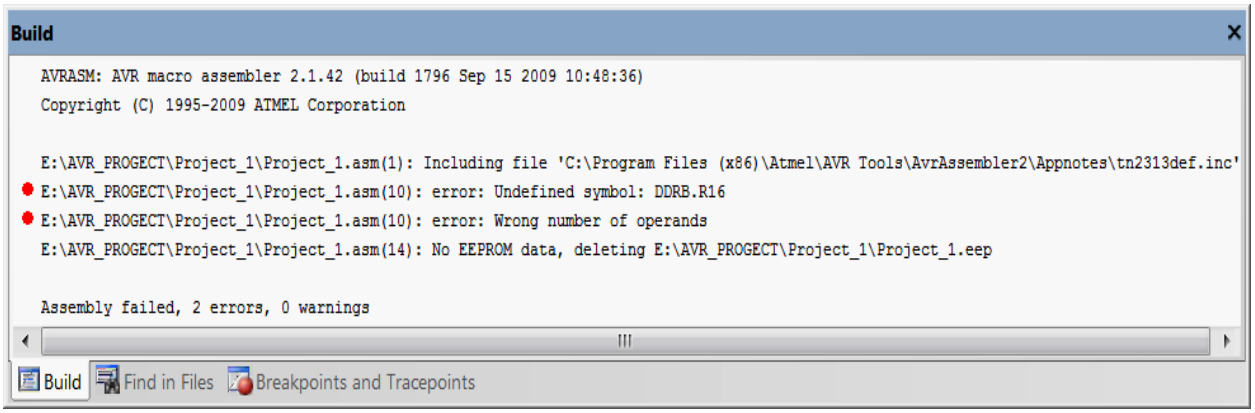


Рис.1.21. Окно с сообщением о неудачном ассемблировании

Первые две строчки в окне информируют о версии ассемблера, с помощью которого выполнена компиляция.

Третья строка показывает путь, по которому найден файл описаний tn2313def.inc. Одновременно это и отображение имени того файла, который используется в проекте. Фактически один и тот же исходный текст может компилироваться с другим файлом описания для другого микроконтроллера.

Четвертая строка с красным кружком символа ошибки сообщает, что в строке №10 (это показано в скобках) имеет место синтаксическая ошибка, которая идентифицирована как " неизвестный символ: DDRB.R16". Это произошло потому, что при записи команды в этой строке была установлена вместо запятой точка, которой в этом месте быть не может.

Если щелкнуть в этом окне на сообщении об ошибке, то строка, в которой обнаружена ошибка, будет указана маркером.

Пятая строка с ошибкой в строке №10 сообщает о "неправильном номере операндов". Такой вывод сформулирован потому, что в команде OUT должно быть два имени, разделенных запятой. Такого разделителя компилятор не смог обнаружить.

Легко видеть, что оба сообщения получены из-за одной точки между двумя именами. Т.е. ошибка одна, а сообщений два. Такое размножение сообщений по причине наличия одной синтаксической ошибки является характерной особенностью работы компиляторов. Причина этого заключается в том, что компилятор на разных этапах синтаксического и семантического анализа по мере обработки всего текста одно и то же видит с разных точек зрения (анализирует несколько раз на разных этапах компиляции). Поэтому выдает разные смысловые сообщения об обнаруженной ошибке. И это объективное развитие процесса компиляции. Поэтому при наличии большого количества сообщений следует оценить наиболее понятные из них, устранить их причины и выполнить повторную компиляцию, что при правильных решениях приведет к сокращению количества сообщений. И так повторять до полного устранения всех сообщений.

Последняя строка говорит, что данные для памяти EEPROM (сегмент .eseg) отсутствуют. Это файл с расширением .eep. Поскольку указаний о таких данных в тексте нет, то и файл не создавался.

После замены в исходном тексте точки на запятую и выполнения компиляции будут получено окно трансляции, показанное на рис.1.22.

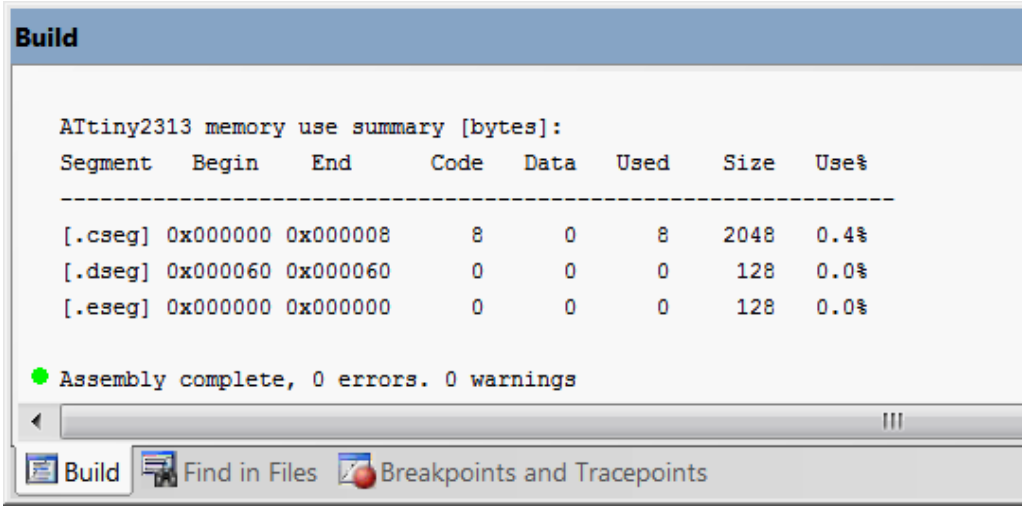


Рис.1.22. Окно с сообщением об успешном ассемблировании

Верхняя часть записей в этом окне совпадает с сообщениями на рис.1.21, а ниже в окне размещена таблица распределения памяти (см. рис.1.22). Строки отображают количество фактической и реально использованной памяти программ (директива .cseg - code segment - сегмент кодов), памяти данных (директива .dseg - data segment - сегмент данных) и памяти постоянной для хранения данных (директива .eseg - erasable segment - перепрограммируемый сегмент или EEPROM). Два последних сегмента в этом примере не использовались.

В таблице видно, что программа заняла 8 байт памяти программ, т.к. написано четыре команды по два байта каждая (колонка Code), никаких констант не образовано (колонка Data), всего использовано памяти 8 байт (колонка Used), общий размер имеющейся памяти 2048 байт (колонка Size), процент использованной памяти - 4% (колонка Use%).

После компиляции в папке с проектом создается несколько файлов. Один из них файл прошивки с расширением .hex. Он является текстовым файлом кодов машинных кодов. При открытии в простейшем текстовом редакторе, например, в Блокноте, можно увидеть коды в шестнадцатеричном формате (рис.1.23)

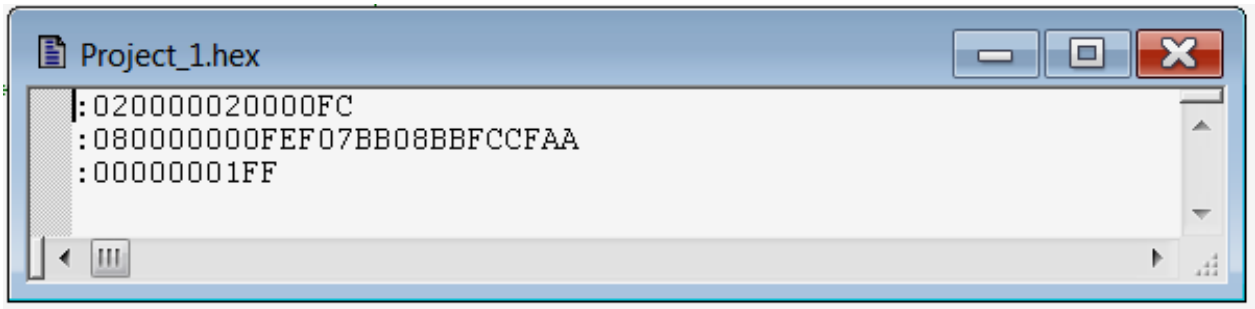


Рис.1.23. Содержимое hex-файла прошивки

Первая и последняя строки являются служебными. Они обеспечивают настройку программатора. Вторая и каждая последующая строка имеют одинаковую побайтную структуру:

08 - количество байт кодов команд в строке;

0000 - адрес начала размещения данных строки в памяти;

00 - служебный байт. Для строки с данными он соответствует коду 00, первой строки - 02, последней - 01;

0FEF 07BB 08BB FCCF - это непосредственно те заявленные в начале строки 8 байт кодов команд, которые являются машинными кодами программы;

AA - это последний байт строки, являющийся контрольной суммой.

Коды программы будут размещены при программировании в памяти программ в последовательно расположенных ячейках, начиная с адреса $0000. Займут они четыре ячейки по два байта каждая. Кроме того, следует учесть, что двухбайтные данные в файле размещаются в обратном порядке: сначала младший байт, затем старший. Эти правила выполняются для всех микропроцессорных устройств. Поэтому в память программ байты будут записаны уже в перевернутом виде. Размещение байт программы в памяти показано на рис.1.24.

Программная память (FLASH)		
	F E D C B A 9 8	7 6 5 4 3 2 1 0
$0000	EF	0F
$0001	BB	07
$0002	BB	08
$0003	CF	FC
$0004	FF	FF

Рис.1.24. Размещение кодов программы в памяти программ

Начиная с адреса $0004, располагаются "чистые" ячейки. В памяти в них будут записаны коды $FF, которые процессором не идентифицируются и он выполнять их не будет, хотя время на их выборку будет потрачено.

В ячейке с адресом $0000 находится код, который соответствует первой команде программы: ***LDI R16,255***. В ячейке $0001 - код второй команды и т.д. Зная структуру команд, их можно редактировать непосредственно в этом файле.

Кроме файла прошивки, в проект вносится файл описаний tn2313def.inc по тому пути, который показан в окне трансляции (см.рис.1.21). Его отсутствие приведет к множественным ошибкам, связанным с неизвестными именами переменных и констант. Имя файла должно строго соответствовать типу микроконтроллера. Директива ***.include*** включает в исходный текст программы содержимое всего этого файла описаний.

По своей структуре это текстовый файл. В нем содержатся директивы ассемблера ***.equ*** (от equivalent - эквивалентный). Ассемблер по ним присваивает при трансляции конкретное значение заданному символическому имени. Фрагмент этого файла показан на рис.1.25. Это позволяет пользоваться содержательными символическими именами, повышая читабельность программы. Кроме того, при необходимости использования программы для другого контроллера потребуется лишь заменить файл описаний, что обеспечивает переносимость программ. На рис.1.25 видно, что используемым в программе именам регистров ввода-вывода DDRB и PORTB сопоставляются числовые адреса 0x17 и 0x18 соответственно.

Кроме значений символических имен в файле есть некоторые константы, имеющие также зарезервированные имена. Например, на рис.1.26 представлены параметры блоков памяти микроконтроллера. В частности последняя ячейка памяти программ имеет адрес 0x03FF, объем ОЗУ - 128 байт т.д..

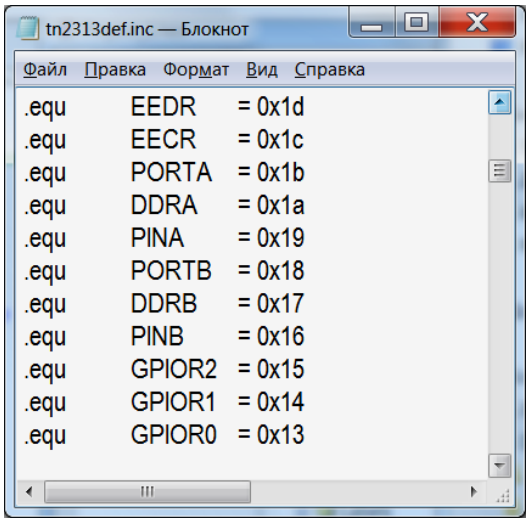


Рис.1.25. Фрагмент файла описаний с именами стандартных регистров

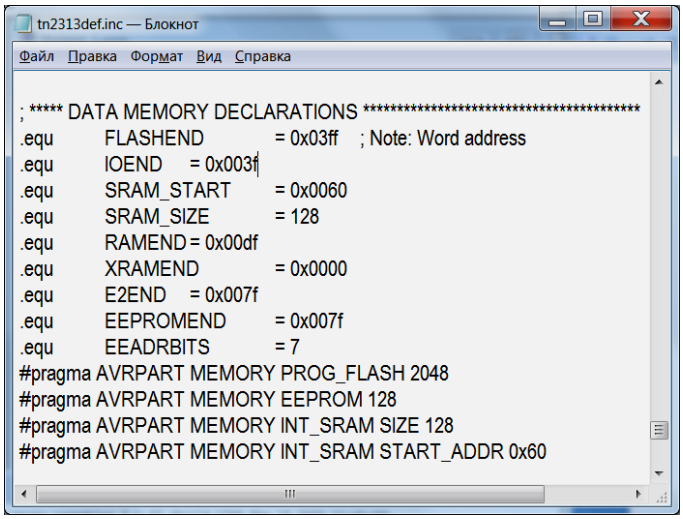


Рис.1.26. Фрагмент файла описаний с именами типовых констант

Еще одним полезным файлом может быть листинг программы. Это файл с расширением ***.lst.*** Чтобы он формировался необходимо выполнить ***Project/Assembler Options/Create List File (***или ***Project/Configurationr Options/Generate List File).*** После очередной компиляции появится Project_1.lst. В нем расположен полный текст программы с включенным файлом описаний, а также кодами всех команд с указанием их адресов в памяти, сообщениями об ошибках, занятой памяти (рис.1.27).

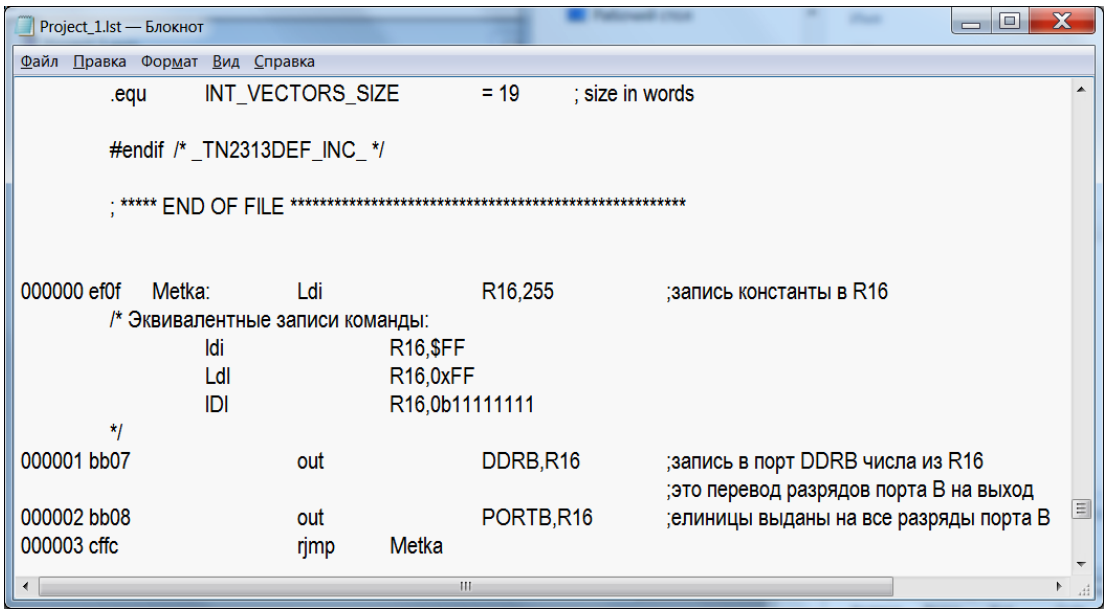


Рис.1.27. Фрагмент листинга программы

Практические задачи:

1. Откомпилировать исходную программу (рис.1.20). При обнаружении ошибок выяснить их причину и устранить.
2. Убедиться, что светодиод HL2 не горит, а HL1 горит.
3. Для схемы на рис.1.19 запрограммировать порт B таким образом, чтобы светодиод HL2 загорелся. В предыдущей программе PORTB настроен на выход и равен единице, поэтому HL2 не горит. Программу занести в отчет. Откомпилировать программу, добиться устранения ошибок.

В отчете представить:

1. Исправленную программу включения обоих светодиодов.
2. Путь к файлу описаний.
3. Путь к файлу листинга.

Вопросы и задания для самоконтроля

1. Каким образом запускается и настраивается AVR Studio?
2. Что такое IDE?
3. Какие параметры можно настроить?
4. Какие возможности обеспечиваются интегрированной средой AVR Studio?
5. Чем отличаются режимы отладки: программной симуляции и режим управления различными типами внутрисхемных эмуляторов?
6. Что является свидетельством корректной компиляции?

7. Что содержит файл исходного текста программы? Какое у него расширение?
8. Что содержит файл описаний? Какое у него расширение?
9. Что содержит файл прошивки? Какое у него расширение?
10. Что содержит листинг программы? Какое у него расширение?
11. Как задать необходимость формирования листинга программы?
12. Каков формат записи ассемблерных программ?
13. Как установить требуемое окно рабочей среды?
14. Каким образом осуществляется управление размещением программы в памяти?
15. Что делает директива ассемблера .include?
16. Как выполняется директива ассемблера .equ?
17. Сформулируйте назначение всех меню среды разработки.
18. Что содержит файл с расширением .obj?
19. Установить окно I/O View и пронумеровать разряды регистра B.
20. Как создать аналогичную программу для другого типа МКК?
21. Как найти файл описаний?
22. Что такое "двоичные коды машинных команд в текстовом формате", которые хранятся в файле с расширением .hex?
23. Записать в двоичной, шестнадцатеричной, восьмеричной и десятичной форме число 27 на ассемблере.
24. Почему в используемой программе адреса регистров 0x17 (DDRB) и 0x18 (PORTB) не попадают в область адресов регистров общего назначения R23 и R24 с аналогичными цифровыми значениями?

2. ИССЛЕДОВАНИЕ РАБОТЫ ИНТЕГРИРОВАННОЙ СРЕДЫ AVR Studio В РЕЖИЕ ОТЛАДКИ

Цель:

1. Углубить и закрепить теоретические знания по принципам программирования и отладки в интегрированной среде;

2. Приобрести практические навыки самостоятельного изучения состава, возможностей, режимов работы среды отладки AVR Studio, освоения элементов интерфейса;

3. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы:

2.1. Запуск и настройка отладчика AVR Studio;

2.2. Работа отладчика для языка ассемблера.

Литература для подготовки к занятию

1. Белов А.В. Разработка устройств на микроконтроллерах AVR: шагаем от "чайника" до профи. - СПб.: Наука и техника, 2013. -528 с. (стр.289-306).
2. Евстифеев А.В. Микроконтроллеры AVR семейства Tiny. Руководство пользователя. М.: Издательский дом Додэка-XXI", 2007. -432 с.
3. WWW.kit-e.ru Компоненты и технологии. AVR: программирование в среде AVR Studio.

Содержание отчета

1. Название работы.
2. Название каждого учебного вопроса и краткий конспект в объеме практических заданий по вопросу.

Вопросы для подготовки к занятию

1. Назначение отладчика.
2. Какие параметры и объекты необходимо контролировать при отладке?
3. Какова логика проведения отладки при отсутствии отладчика?
4. Какие особенности отладки имеют место в системах реального времени?

Актуальность занятия

Разработка устройств на основе специализированных контроллеров связана с разработкой аппаратной и программной составляющих. Кроме того, это связано с особенностями разработки систем реального времени. Поэтому все этапы проектирования и отладки должны быть максимально

автоматизированы и адекватны объекту. Этим требованиям в полной мере отвечает среда AVR Studio. Она является признанным продуктом, и ее изучение может стать основой для освоения других аналогичных систем.

Задание 2.1. Запуск и настройка отладчика AVR Studio

Практические задачи:

1.Выполнить запуск отладчика, в отчете отразить выполняемые операции;

2. Осуществить настройку отладчика, в отчете перечислить параметры настройки и технологию доступа к ним.

IDE AVR_Studio — это интегрированная среда разработки приложений для микроконтроллеров семейства AVR (IDE – Integrated Development Environment), .свободно распространяемый продукт фирмы Atmel (Atmel.com).

Отладчик AVR Studio поддерживает все типы микроконтроллеров AVR и имеет два режима работы: режим программной симуляции и режим управления различными типами внутрисхемных эмуляторов (In-Circuit Emulators) производства фирмы Atmel. Важно отметить, что интерфейс пользователя не изменяется в зависимости от выбранного режима отладки.

Отладочная среда поддерживает выполнение программ, как в виде ассемблерного текста, так и в виде исходного текста языка C/C++. В последнем случае загружаемый модуль программы должен содержать не только машинный код, но и исходный текст программы, привязанный к машинному коду.

При разработке и отладке программ удобно использовать встроенные средства поддержания проекта. Понятие "проект" в интегрированных средах разработки включает: перечень файлов, содержащих исходные модули, пути размещения этих и создаваемых в процессе трансляции и компоновки файлов, параметры настройки процессов создания и отладки программы и др. Удобство заключается в том, что однократно осуществленная настройка проекта, сохраняется в поименованном файле (name.aps), а потом используется при открытии файла проекта.

В процессе симуляции можно наблюдать состояния регистров общего назначения R0…R31, портов ввода/вывода (I/O), процессора, памяти программы (ПЗУ) и памяти данных (ОЗУ).

Выбор устройства и настройка отладчика

Необходимо проконтролировать, чтобы симуляция кода проводилась именно для выбранного типа микроконтроллера и отладочной платформы. Это осуществляется из меню ***Debug*** (Отладка) пункт ***Select Platform and Device*** (см. рис.1.5). Для рассматриваемых примеров это ***ATtiny2313*** и ***AVR***

Sumulator. Эти настройки выполняются, обычно, при создании проекта и повторно эти операции проводят только ради контроля или при смене контроллера.

Запуск отладчика

Запуск можно осуществить несколькими способами:

- выполнить ***Debug/Start Debugging***;
- одновременно нажать ***Ctrl+Shift+Alt+F5***;
- выбрать на панели инструментов кнопку  ;
- выполнить ***Build/Build and Run***;
- нажать ***Ctrl+F7***;
- выбрать на панели инструментов кнопку  .

Последние два способа предполагают выполнение компиляции с последующим запуском на выполнение.

О переходе в режим симуляции свидетельствует появление желтой стрелки указателя отладчика слева от текста с кодом команд (он устанавливается и убирается при нажатии ***Ctrl+ F2,*** а для поиска маркированных команд - ***F2***), указывающей на инструкцию, которая будет выполняться на следующем шаге в соответствии с содержимым счетчика команд PC (рис.2.3).

Выбор средств отображения состояния системы

После запуска изменяется интерфейс и устанавливается курсор на начало программы в главном окне. Дополнительно окна можно открыть и закрыть в меню ***View*** (см. рис.1.12), либо активировав соответствующую кнопку панели инструментов. В режиме отладки дополнительно из окна просмотра можно открыть (рис.2.1):

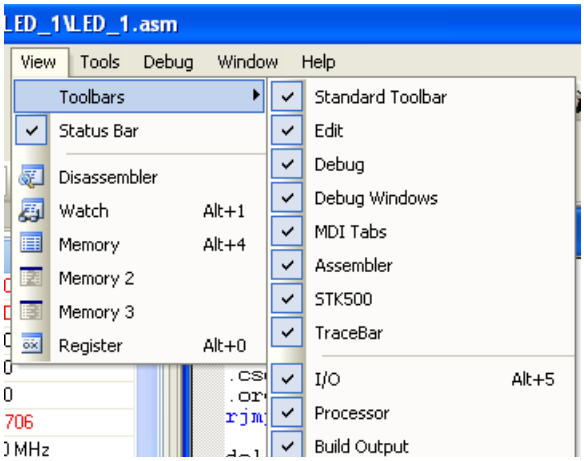


Рис.2.1. Всплывающие списки панелей инструментов и вызываемых окон для отображения состояний микроконтроллера в режиме симуляции-отладки программы

- окно Дизассемблера (***Disassembler***). Показывает результаты обратного преобразования машинных кодов в исходный текст. Это "отображение машинных кодов процессором": как он их "понимает" и как выполняет. Точное взаимное соответствие между кодами и исходным текстом программы в этом процессе получено может быть не всегда;

- окно Просмотра значения переменных (***Watch***). В это окно (рис.2.2) можно заносить интересующие вас регистры и переменные (в тексте программы на имени нажать правую кнопку и выбрать ***Add Watch***), наблюдать их состояние по ходу выполнения про-

граммы, удалить из окна (выбрать имя и правой кнопкой выбрать ***Remove Selected Item***);

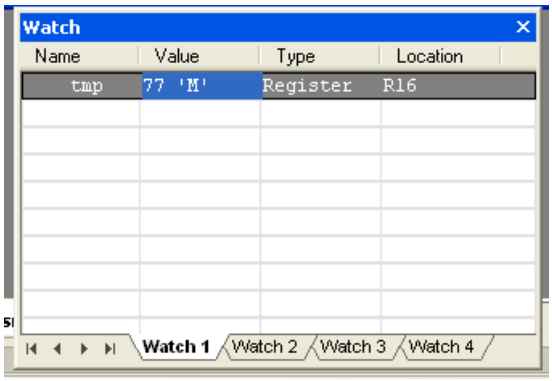


Рис.2.2. Окно просмотра значений регистров и переменных, определенных в программе

- окно Памяти (***Memory***). В этом окне можно просмотреть все виды памяти микроконтроллера, выбрать формат отображения данных, добавить таблицу отображения данных в ASCII-кодах;

- окно Регистров общего назначения (***Register***). В нем показаны все 32 регистра общего назначения.

Окно Процессора (***Processor***) отображает самые важные регистры процессора и отдельные параметры процесса симуляции (рис.2.3):

- состояние программного счетчика (***Program Counter***). Это адрес очередной выполняемой команды. Его можно изменить прямо в окне и процесс выполнения программы будет соответствовать введенному адресу;

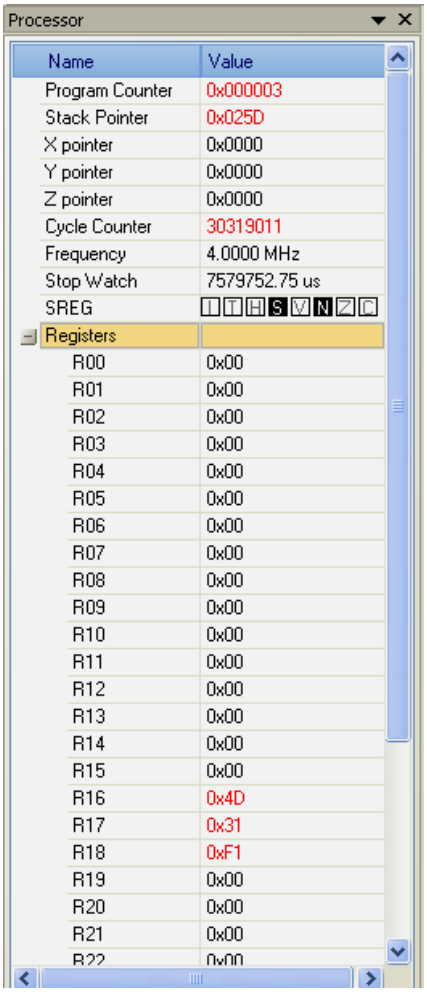


Рис.2.3. Окно ***Processor***

- состояние указателя стека SPL (***Stack Pointer***). В этом окне его изменить нельзя, но можно изменить в окне ***Memory*** по его адресу $3D. Для этого в окне ***Memory*** выбирается закладка ***I/O*** и вписывается нужное значение;

- состояние двухбайтных регистров указателей (***point***) X, Y, Z. Первые два используются для адресации памяти данных, поэтому отображаются как 8-разрядные. Регистр Z адресует память программ, поэтому отображается 16-разрядным;

- счетчик циклов (***Cycle Counter***). Показывает выполненное процессором общее количество машинных циклов. Каждая команда для выполнения требует разное количество машинных циклов, но чаще всего один;

- частота работы микроконтроллера (***Frequency***). Ее можно настроить, выполнив ***Debug/AVR Simulator Options***. В открывшимся окне (рис.2.4) выбирается опция ***Devise Selection*** и устанавливается необходимая тактовая частота.

- счетчик времени (***Stop Watch***). Он показывает время, прошедшее с начала работы процессора. Следует понимать, что это время не реальное, а имитационное, рассчитанное по тактовой часто-

те процессора. Счетчик времени и счетчик циклов можно обнулить, нажав правую кнопку и выбрав команду ***Reset***;

- состояние регистра статуса (флагов) ***SREG***. Выбрав отдельный бит, его состояние можно изменить на противоположное вручную;

- регистры общего назначения ***Registers*** (***Alt+0*** или ).

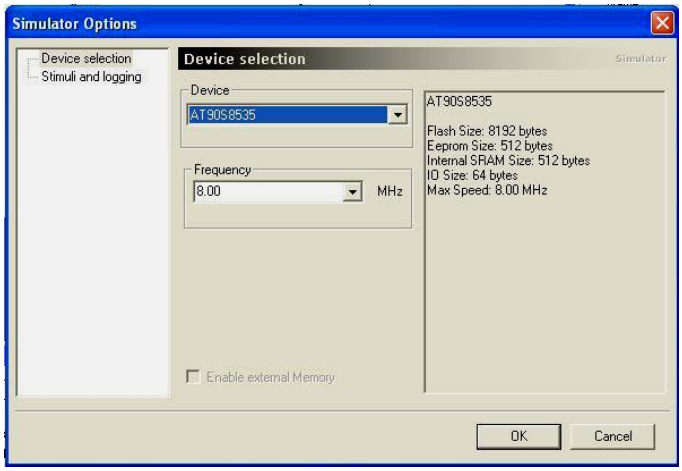


Рис.2.4. Выбор частоты работы МКК

Настройка окна ввода-вывода

Открыть окно Ввода-вывода (***I/O View***) можно, выполнив ***View/Toolbar/(I/O)*** или нажав ***Alt+5***. В окне отобразятся элементы архитектуры выбранного микроконтроллера (рис.2.5). Окно позволяет наблюдать работу любого периферийного устройства.

Для настройки окна используют кнопки в основном заголовке или меню, вызываемое правой кнопкой на заголовке панели регистров и самих регистров.

Выпадающее меню основного заголовка может изменить вид отображения окна:

- в виде дерева (***Tree View***);
- в виде полного списка всех портов (***Flat Register View***);
- в виде раздельных окон (***Module Split View***). При выборе устройства во втором окне появляются регистры, ему принадлежащие (см. рис.2.5). При этом плюсиком раскрываются названия этих регистров и элементы управления, в которые можно установить необходимые значения. Состав второго окна можно изменить в выпадающем списке главного меню.

Темный квадратик разряда регистра соответствует единице ("1"), светлый - нулю ("0"). В процессе отладки значения разрядов регистров можно корректировать вручную. Они меняются щелчком мыши. Редактирование доступно только при запущенном режиме отладки.

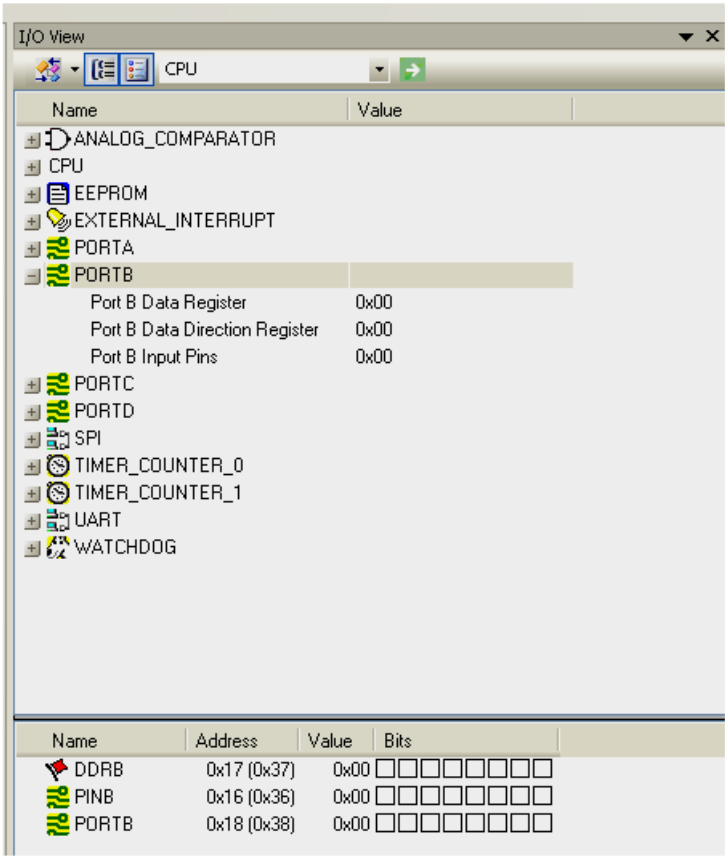


Рис.2.5. Окно устройств ввода/вывода

С помощью установок можно, нажав правую кнопку:

- пронумеровать разряды регистров - ***Show Bitnumber***;
- открыть нужные колонки данных при отображении - ***Columns***;
- сохранить в текстовом файле все значения регистров выбранного модуля - ***Export***;
- установить систему счисления для показа состояния регистров - ***Hexadecimal Display*** или только для одного регистра, если меню вызывать на нем;
- выделить/спрятать контакты - ***Pin/Unpin Selected***. Выделяет регистр синим цветом и позволяет не убирать его из окна при любых переключениях окна. Регистр будет доступен для просмотра и редактирования всегда.

В отчете представить:

1. Технологию запуска отладчика;
2. Параметры настройки отладчика и технологию доступа к ним.

Задание 2.2. Работа отладчика для языка ассемблера

Практические задачи выполняются после усвоения материала и сформулированы в конце задания 2.2. В процессе изучения материала необходимо выполнять описанные операции практически.

Полный перечень режимов работы симулятора доступен из функции главного меню ***Debug*** (рис.2.6). Некоторые функции продублированы пиктограммами на панели инструментов, которые можно вывести на экран из меню ***View/Toolbars*** (см. рис.2.1).

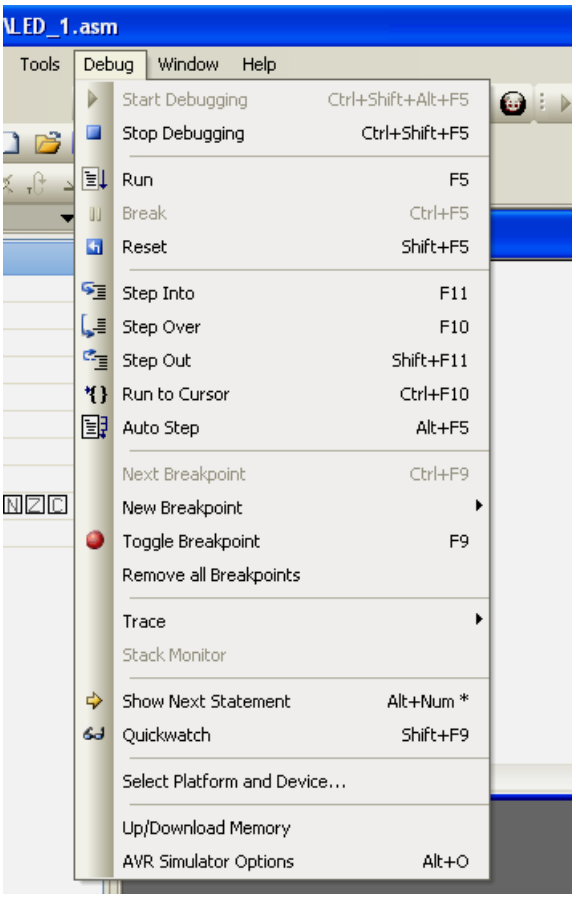


Рис.2.6. Инструменты и функции AVR Studio в режиме отладки

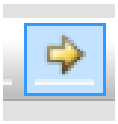
Процесс отладки заключается в отслеживании хода выполнения программы и управления данными в регистрах с помощью окна с текстом кода, окна консоли вывода текущих сообщений (***Message***) и специальных окон, показывающих состояния регистров ввода/вывода (***I/O View***), позволяющих вести контроль значений переменных (***Watch***), отслеживать состояния процессора (***Processor***).

Функции отладчика, управляющие ходом симуляции кода программы, представлены в таблице

.

Наименование	Назначение	Комбинация клавиш	Пикто-грамма	Комментарии

Reset	Сброс	***Shift+F5***		Сброс МКК, все начинается с нуля
Step Into	Переход к следующей строке кода	***F11***		Пошаговое выполнение программы без пропусков
Step Over	Переход через подпрограмму	***F10***		Выполнение подпрограммы без пошагового режима (быстро). Рационально при отлаженной подпрограмме
Step Out	Выход из подпрограммы	***Shift+F11***		Позволяет принудительно без пошагового режима выйти из цикла или подпрограммы
Run to cursor	Выполнение до курсора	***Ctrl+F10***		Выполнить операции кода до точки, указанной текстовым курсором
Autostep	Автоматическое пошаговое выполнение программы	***Alt+F5***		Выполнение программы по строкам кода в автоматическом режиме до нажатия паузы
Break	Принудительная остановка (пауза)	***Ctrl+F5***		Остановка выполнения программы (пауза)
Toggle BreakPoint	Назначение точек остановки автоматического выполнения	***F9***		Позволяет поставить на место, указанное курсором точку программной остановки автоматического выполнения

Remove All BreakPoint	Удаление всех точек останова			Удаляет все точки останова автоматического выполнения
Start Debugging	Запуск отладчика	***Ctrl+Sift+ Alt+F5***		Переход к режиму отладки-симуляции
Stop Debugging	Выход из отладчика	***Ctrl+ Shift+F5***		Остановка симуляции и переход в режим редактора. Это сброс МКК, все регистры обнуляются
Run	Запуск программы на непрерывное выполнение	***F5***		Выполнение программы с максимальной скоростью, но она значительно меньше реальной, т.к. это режим симуляции
Show Next Statement	Показать следующую команду	***Alt+Num***		При больших текстах программ покажет ту часть текста, где расположен указатель счетчика команд (желтый курсор)

Точки останова предусматриваются двух типов: программные и точки останова по данным. В состоянии остановки можно удобно контролировать, какие изменения произошли с регистрами и переменными в данном месте программы. Используя точки остановки, можно также ускорить ход выполнения программы при отладке.

Программные точки останова указываются текстовым курсором с последующим выполнением ***Toggle BreakPoint*** (переключатель точек останова) после нажатия правой кнопки мыши. Несколько более широкий вариант предполагает выполнение ***Debug/New Breakpoint/Program Breakpoint.*** В текстовом файле слева от строки появляется бордовый шар, до которого выполняется программа. Продолжение выполнения программы инициируется повторным запуском.

Удаление точки останова выполняется ***Remove All BreakPoint*** (если удаляются все точки) или ***Toggle BreakPoint*** (если удаляется выбранная кур-

сором точка). Кроме того, выполнив ***View/Toolbars/Breakpoints and Tracepoints***, в окне (рис.2.7) можно отключить/включить точку останова (убрав/добавив галочку, что приводит к обесцвечиванию/засвечиванию бордового шара в исходном тексте) или удалить точку совсем (выбрав в окне точку и выполнив ***Remove Selected***).

Временно отключить/включить выбранную точку можно также в контекстном меню, выполнив ***Disable/EnableBreakpoin.***

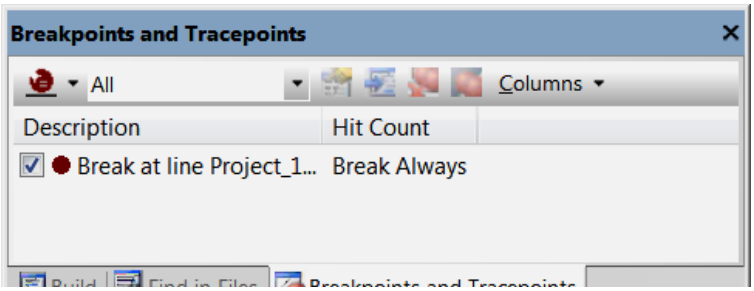


Рис.2.7. Окно точек останова и трассировки

Точки останова по данным срабатывают при выполнении некоторого условия. Для установки точки останова по данным следует выполнить ***Debug/New Breakpoint/Data Breakpoint.*** При этом в появившемся окне (рис.2.8):

- в списке выбора условий останова (***Break when)*** выбирается математическая формулировка условия останова;
- в окне ***Location*** - адрес регистра, в котором это условие будет контролироваться;
- в окне ***Value*** - численное значение величины выбранного условия;
- в поле ***Access type*** - режим работы регистра для останова (чтение, запись или произвольная операция);
- в поле ***Break execution after*** - количество выполнений условия до останова программы.

Управление и просмотр точек останова по данным осуществляется в окне ***Breakpoints and Tracepoints*** (см. рис.2.7).

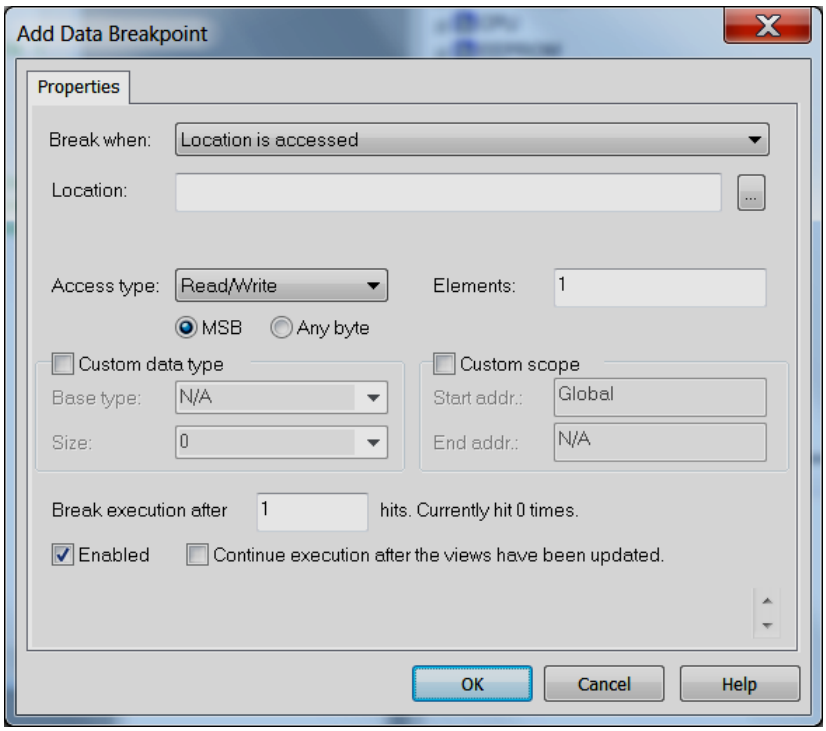


Рис.2.8. Окно установки точек останова по данным

Запустив программу на исполнения, проследите, как изменяется положение указателя хода выполняемой операции, нажимая несколько раз клавиши ***F11, F10, Shift+F11, Ctrl+F10, Alt+F5*** и ***Ctrl+F5*** или выбрав соответствующую опцию из пиктограмм отладчика. Обратите внимание, что при изменении состояний регистров микроконтроллера и его портов, их цвета изменяются с черного на красный.

Наблюдайте, как увеличивается значение данных порта B и как изменяются значения данных в счетчике команд, счетчике цикла, в регистрах процессора и в слове состояния процессора.

Изменение программного кода

Чтобы отредактировать соответствующим образом код программы, остановите работу отладчика. Поместите курсор в нужное место кода и измените инструкцию. Если теперь нажать {F5}(Run), то появится диалоговое окно, которое указывает, что один из исходных файлов проекта был изменен, и требуется новая компиляция и перекомпоновка проекта. Перекомпилируйте проект и вновь запустите отладчик. Обратите внимание, что предыдущие настройки окон и положение точек остановки программы сохраняется.

Сохранение проекта

Перед окончанием работы AVR Studio предложит сохранить ваш проект. AVR Studio запомнит все открытые вами окна и их настройки и воспроизведет их при следующем открытии проекта. Сохранение проекта осуществляется выбором пункта ***Project/Save***" (или ***Save As***…).

Загрузка программы в ПЗУ

После того как прикладная программа отлажена на симуляторе, ее можно загрузить в память микроконтроллера. Для этой цели служит программа и устройство: программатор. Для загрузки предварительно должен быть сформирован файл прошивки (*.hex-файл).

Переход среды AVR Studio в режим программирования производится активизацией строки “AVR Prog…” пункта меню "Tools". При правильном соединении компьютера и платы контроллера на экране появляется диалоговое окно программы прошивки.

Для выхода из симулятора следует выбрать ***Debug/Stop Debugging*** или нажать кнопку  .

Практическое задание:

1. Дополнить программу: назначить выводы портов D и A на выход и выдать на порт D байт из чередующихся нулей и единиц, а на порт A – "1" на разряд №1 и "0" на разряд №0. Программу дописать в отчет;
2. Оттранслировать программу и исправить все ошибки;
3. В режиме эмуляции проверить все изменения регистров;
4. Установить точку останова и проверить работу микроконтроллера;
5. Просмотреть в окне Memory изменение ячеек памяти, определить адреса изменяющихся ячеек и их соответствие именам регистрам ввода-вывода;
6. Определить количество машинных тактов в одном периоде выполнения программы, затраченное на это время, длительность одного машинного такта. Выполнить расчеты, подтверждающие правильность полученных значений эмулятором;
7. Изменить тактовую частоту процессора. Оценить разность во времени выполнения одного машинного такта при нынешней и прежней тактовой частоте;
8. Проверить поведение эмулятора при установленной точке останова.

В отчете представить:

1. Дополненную программу по п.1 практического задания.
2. Расчеты длительности машинного такта, числа циклов и длительности одного периода выполнения программы.

Вопросы для самоконтроля

1. Какие режимы отладчика существуют?
2. Как настроить среду программирования на языке Ассемблера или С?
3. Каковы признаки синтаксических ошибок при компиляции?
4. Как запустить отладчик для работы с языком С?
5. Какие объекты можно наблюдать в режиме отладки?

6. Какова последовательность отладки?
7. Какие ошибки можно выявить с помощью отладчика после успешной компиляции?
8. Какие опции настраиваются для компоновки?
9. Как изменить программный код отлаживаемой программы?
10. Каков состав программных средств AVR Studio?
11. Можно ли изменять содержимое регистров в процессе отладки?
12. Зачем используют точки останова?
13. Чем отличаются режимы Autostep, Step Over, Step Into?
14. Провести дизассемблирование исследуемого кода. Определить смысл команды по адресу 0000.
15. Каким образом рассчитать длительность машинного такта, числа циклов и длительность одного периода выполнения программы?

3. СИСТЕМА СХЕМОТЕХНИЧЕСКОГО МОДЕЛИРОВАНИЯ Proteus

Цель:

1. Углубить и закрепить теоретические знания по принципам работы электронных устройств;

2. Приобрести практические навыки самостоятельного изучения состава, возможностей, режимов работы среды моделирования Proteus, освоения элементов интерфейса;

3. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы:

3.1. Запуск и настройка системы моделирования Proteus;
3.2. Сборка электронной схемы;
3.3. Моделирование схемы и отладка.

Литература для подготовки к занятию

1. Proteus по-русски: Тематический обзор печати и интернет-ресурсов Радиоежегодник 2013 выпуск 24;
2. www.rlocman.ru/radioyearbook - электронная версия журнала Радиоежегодник 2013 выпуск 24;
3. http://cxem.net/comp/comp117.php - цикл статей по работе в Proteus;
4. http://kazus.ru/forums/forumdisplay.php?f=25 форум по обсуждению вопросов по работе с программой Proteus.

Содержание отчета

1. Название работы.
2. Название каждого учебного вопроса и краткий конспект в объеме практических заданий по вопросу.

Вопросы для подготовки к занятию

1. Назначение систем моделирования электронных схем.
2. Какие параметры цифровых программируемых схем невозможно увидеть в системах отладки?
3. Какие параметры можно контролировать в системах моделирования электронных схем?
4. Какие допущения имеют системы моделирования электронных схем?

Актуальность занятия

Разработка устройств на основе специализированных контроллеров связана с разработкой аппаратной и программной составляющих. Эти

компоненты должны работать совместно и в реальном времени. Поэтому процессу отладки отводят очень значительное время и придают принципиальное значение. Овладение инструментами программирования и отладки является обязательным условием эффективного проектирования устройств управления на микроконтроллерах.

Задание 3.1. Запуск и настройка системы моделирования Proteus

Практические задачи:

1.Выполнить запуск системы;

2. Осуществить настройку системы.

Proteus - это система моделирования электронных схем. Основывается работа программы на моделях электронных составляющих, принятых в PSpice, отсутствующие модели можно создать самостоятельно. Кроме того реализованы модели микропроцессоров, микроконтроллеров 8051, AVR, PIC, HC11, MSP430, ARM7/LPC2000 и другие. Работает с большинством компилятором (в частности, СИ) и ассемблерами.

Proteus состоит из двух главных модулей:

ISIS - (Intelligent Schematic Input System - интеллектуальная система схематического ввода) –графический редактор принципиальных схем, служит для ввода разработанных проектов с последующим моделированием и передачей для проектирования печатных плат в ARES;

ARES - графический редактор печатных плат со встроенным менеджером библиотек и автотрассировщиком ELECTRA, автоматической расстановкой компонентов на печатной плате.

PROTEUS имеет уникальные компоненты:

USBCONN - этот инструмент дает возможность включиться к действительному USB порту компьютера;

COMPIM - данный компонент дает возможность виртуальному устройству включится к реальному COM-порту вашего ПК.

Микроконтроллер это программируемый универсальный цифровой модуль. Для разработки и отладки программы для него используют интегрированные системы программирования и отладки. Они позволяют использовать множество удобных и эффективных процедур, оценивающих логику разрабатываемых программ. Однако такие системы не предназначены для демонстрации работы устройства в реальном времени. Для микроконтроллеров это принципиально важно. Такую возможность предоставляют системы моделирования. Профессиональная версия Proteus обеспечивает также трехмерное изображение получающегося устройства и трассировку печатных плат.

Запуск симулятора

Запуск можно осуществить несколькими способами:

- запустить приложение с Рабочего стола, выбрав ярлык;
- выполнив последовательно Все_программы/Proteus 8 Demonstration/Proteus 8 Demonstration.exe.

Настройка параметров системы моделирования

Далее последовательно выбрать:

- ***NewProject;***
- в стартовом окне мастера нового проекта ***New Project Wizard: Start*** зафиксировать имя и путь к файлу нового проекта. Выбрать Next;
- в окне эскиза ***New Project Wizard: Schematic Design*** - сохранить - "По умолчанию" (***DEFAULT***), нажать Next;
- в окне макета печатной платы ***New Project Wizard: PCB layout*** (PCB - Printed Circuit Board) установить: Не создавать макет печатной платы (***Do not creat a PCB layout***). Выбрать Next;
- в окне встроенного программного обеспечения ***New Project Wizard: Firmware*** выбрать "Отсутствие проекта встроенного программного обеспечения" (***No Firmware Project***). При работе с оригинальным ПО в этом окне возможно выбрать "Создать проект встроенного программного обеспечения" (Create Firmware Project). В этом случае необходимо выбрать семейство микроконтроллера (Family), тип контроллера (Controller) и наименование компилятора (Compiller). Это позволяет осуществлять разработку ПО непосредственно в Proteus. При использовании демоверсии можно просмотреть допустимые варианты этих составляющих, но использовать их недопустимо. Поэтому загружать в модель необходимо уже готовый откомпилированный код. Нажать Next;
- в итоговом окне мастера нового проекта ***New Project Wizard: Summary*** просмотреть итоговые свойства проекта (создание только схемы), уточнить детализацию (Details) и нажать "Готово" (Finish). На экране будет представлено окно с итоговыми результатами настройки.

В отчете представить:

1. Состав параметров, выбираемых при запуске Proteus.

Задание 3.2. Сборка электронной схемы

Практические задачи:

1.Собрать схему, представленную на рис.1.19;

2. Осуществить настройку параметров элементов схемы.

Основные элементы интерфейса Proteus

На рис.3.1 представлено основное окно моделирующей программы

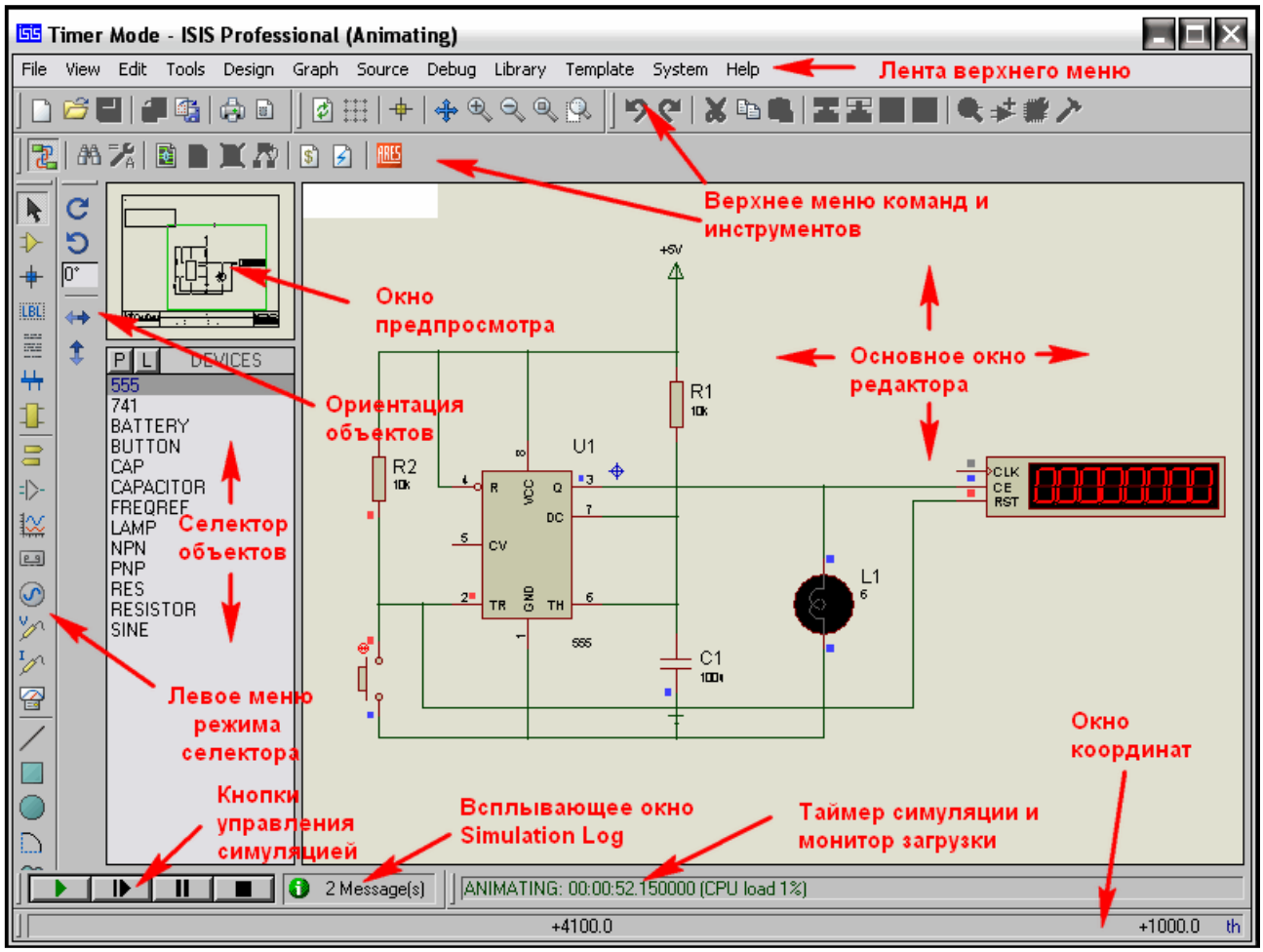


Рис.3.1. Основное окно моделирующей программы

Состав меню верхней ленты в основном типовой. Так для меню ***File*** базовыми возможностями являются (рис.3.2):

NewDesign – создать новый проект;

OpenDesign (Ctrl+O) – открыть существующий проект;

SaveDesign (Ctrl+5) – сохранить изменения в текущем проекте;

SaveDesignAs... – сохранить проект в других версиях программы Proteus;

SaveDesignAsTemplate... – сохранить проект как образец;

WindowsExplorer... – откроется папка, где хранится проект;

ImportBitmap... вставка внешнего рисунка в формате BMP;

ImportSection... – вставка внешнего фрагмента (позволяет перемещать части схемы из одного проекта в другой);

ExportSection... – преобразование во внешний фрагмент;

ExportGraphics – преобразование схемы в другие форматы (рисунок BMP, чертеж DXF и др.);

MailTo... – отправить по электронной почте;

Print...– печать, настройка печать.

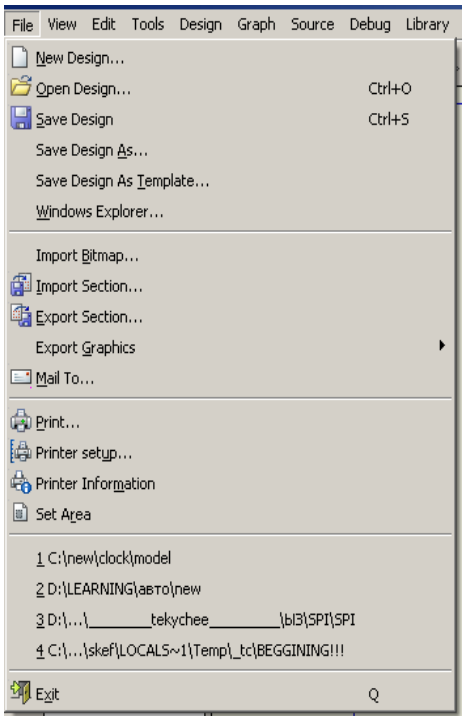


Рис.3.2. Меню ***File***

Меню "Вид" (***View***) предназначено для задания параметров представления внешнего вида создаваемой схемы (рис.3.3):

Redraw - перерисовать;

Grid - установка сетки;

Origin - относительная/абсолютная система координат;

- размеры сетки ***Snap*** (в точках или дюймах);

Pan - понорамирование;

Zoom In - увеличить масштаб;

Zoom Out - уменьшить масштаб;

Zoom All - показать все;

Zoom to Area - показать выделенную область;

Toolbars - панели инструментов.

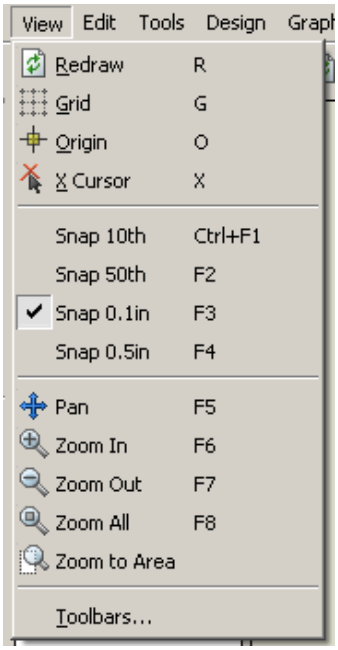


Рис.3.3. Меню ***View***

Меню Редактирования (***Edit***) позволяет:

- отменить последний (***Undo Changes***) или восстановить предыдущий (***Redo Changes***) шаги редактирования;

- вырезать (***Cut to Clipboard***), скопировать (***Copy To Clipboard***) выделенное изображение в буфер, вставить из буфера (***Paste From Clipboard***);

- выделить все объекты схемы (***Select All Objects***);

- найти и отредактировать свойства объекта по его позиционному обозначению (***Find/Edit Component***).

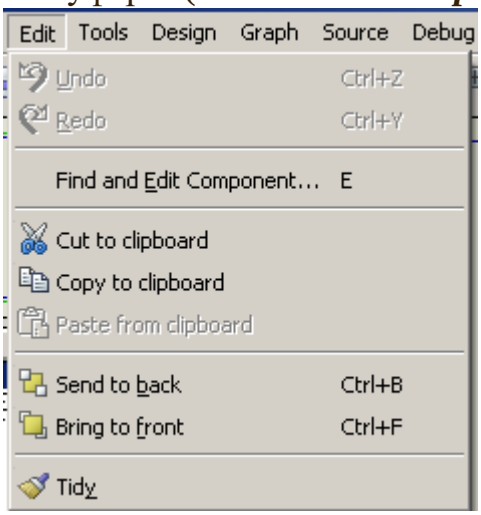


Рис.3.4. Меню ***Edit***

Выбор компонентов электронной схемы

Выбор компонентов осуществляется из библиотеки элементов. Сделать это можно двумя способами:

- осуществив переход ***Library/Pick_part_from_Library;***
- нажав кнопку ***P*** в окне установленных компонентов ***DEVICES.*** Если окно в настоящий момент отсутствует, то его можно вызвать, нажав кнопку Компонентов (***Component Mode)*** на левой панели инструментов  .

В открывшемся окне Выбора устройств (***Pick Devices)*** поиск осуществляется по категориям (***Category***) и подкатегориям (***Sub-category***), типу корпуса (***Manufacturer***). На рис.3.5 представлено окно для выбора микроконтроллера AtTiny2313. Следует учесть, что контакты цепей питания микроконтроллеров в этой системе не отображаются: питание на них подано по умолчанию.

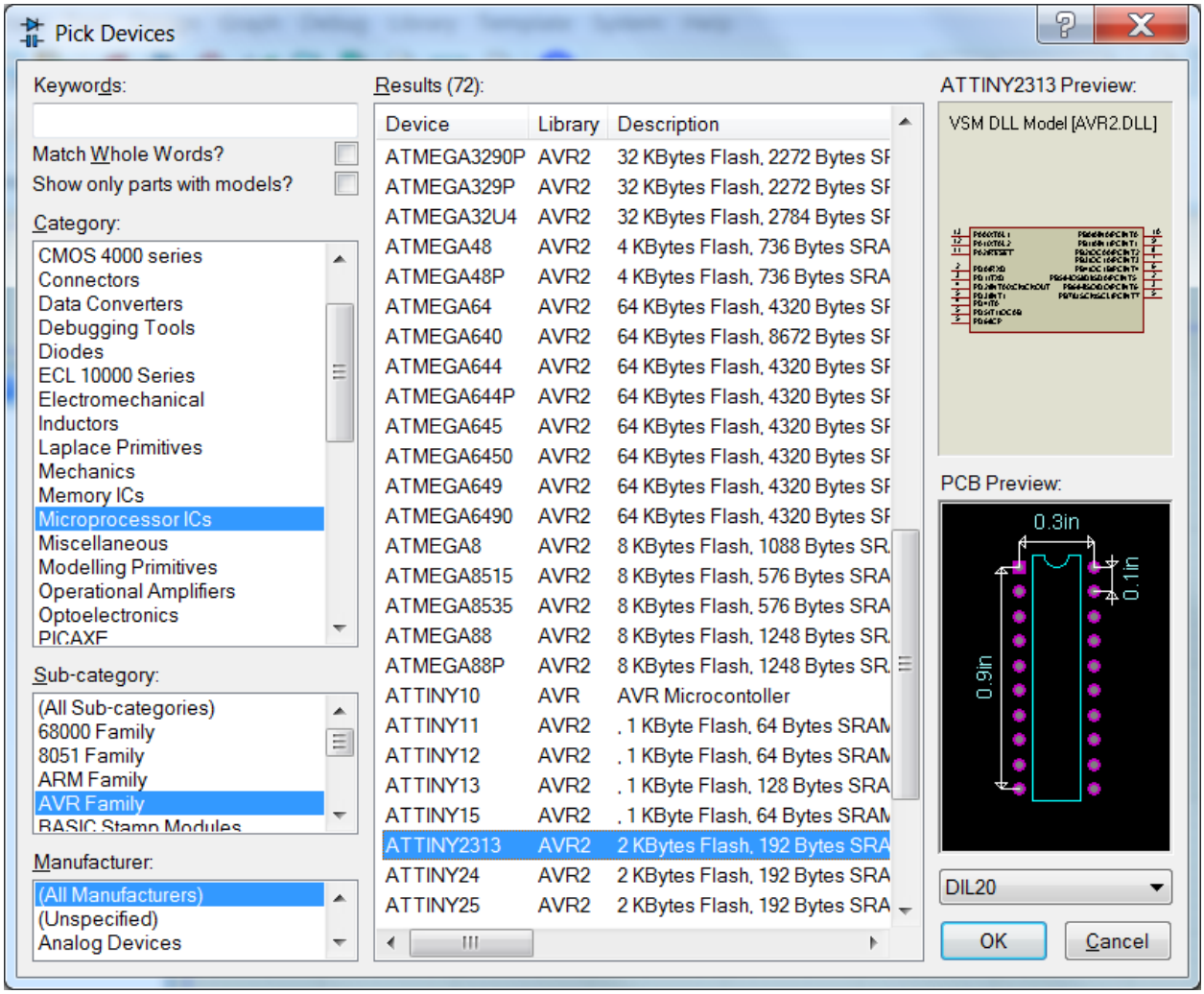


Рис.3.5. Окно ***Pick Devices***

Облегчить поиск можно, заполнив окно Ключевое слово (***Keywords***). При этом по мере ввода ключевых символов система будет последовательно актуализировать все отображаемые окна.

В окне ***Pick Devices*** в полях предварительного просмотра отображаются условное графическое обозначение компонента (***ATTINY2313 Preview***) и его конструктивное исполнения (***PCB Preview)*** для последующей разработки печатной платы и трассировки цепей.

Если на выбранном компоненте дважды щелкнуть левой кнопкой, то элемент переместится в окно ***DEVICES.*** В него рационально после поиска разместить все необходимые компоненты, что значительно ускорит набор электронной схемы. Аналогичные результаты получаются при нажатии кнопки ***Ок***. Удаление элементов из списка ***DEVICES*** выполняется выделением элемента в списке, вызовом правой кнопкой контекстного меню и выбором опции Удалить (***Delete***).

Для будущей схемы будут полезны некоторые конкретные виды элементов:

1. Элементы коммутации в группе Переключателей и реле (***Switches & Relays***). Здесь можно выбрать кнопки (***BUTTON***), групповые переключатели на разное количество соединений (***Switch***), а также реле (***Relays***);

2. Элементы индикации в группе Оптоэлектроники (***Optoelectronics***). В подгруппах Светодиодов (***LEDs***), Графических светодиодных индикаторов (***Graphical LCDs***), Знакосинтезирующих индикаторов (***Segment Displays***) можно подобрать необходимый прибор;

Светодиоды различаются по цвету (например, зеленый ***LEDs-GREEN***) и режиму моделирования. В режиме цифровом (***Digital***) светодиод не потребляет ток, а в режиме аналоговом (***Analog***) потребление тока осуществляется по общим правилам;

3. Резисторы выбираются в группе ***Resistors***. Они различаются по рассеиваемой мощности, конструкции, величине сопротивления и т.д. Для использования переменных резисторов лучше выбрать тип ***POT-HG***;

4. Конденсаторы находятся в группе ***Capacitors***;

5. Транзисторы в группе ***Transistors***;

6. Терминальные элементы (концевые, ввода и вывода, пограничные) выбираются кнопкой ***Terminals Mode***  в окне Редактора схем ***Schematic Capture.*** В частности там можно выбрать элемент Общий повод ***GROUND***, Питание ***POWER;***

7. Измерительные приборы выбираются в группе ***INSTRUMENTS***  в окне Редактора схем ***Schematic Capture.*** Здесь можно найти, в частности, четырехканальный осциллограф (***OSCILLOSCOPE***), вольтметры (***VOLTMETER***) и амперметры (**AMMETER**) в режиме измерения постоянного (***DC***) и переменного (***AC***) токов;

8. Генераторы сигналов выбираются кнопкой ***Generator Mode***  в окне Редактора схем ***Schematic Capture.*** Доступны генераторы постоянного напряжения ***DC***, синусоидального ***SINE***, импульсного ***PULSE*** и т.д.

Размещение и редактирование элементов

Размещение элементов в окне редактора осуществляется путем выделения необходимого элемента в окне ***DEVICES,*** щелчком в окне (элемент отображается розовым цветом) и еще одним щелчком фиксации на выбранном месте размещения в окне (элемент отображается черным цветом).

Выделение элементов осуществляется щелчком ЛКМ в окне редактора. Выделение группы элементов осуществляется нажатием и обводкой группы элементом ЛКМ. Все элементы можно выделить через меню ***Edit/Select All Objects.*** Операция сопровождается отображением стрелки белого курсора.

Масштабирование изображения окна осуществляется вращением колесика прокрутки мыши в нужном направлении. Для удобного наблюдения набранных элементов в рабочем окне используется фокусирующий контур зеленого цвета в окне Редактора схемы (***Schematic Capture***). Им можно выбрать ту часть поля, которая будет отображаться в настоящий момент в центре рабочего окна. Фиксация окна выполняется нажатием ЛКМ.

Перемещение элемента на экране осуществляется нажатием и перемещением элемента ЛКМ. Если выделить контуром группу элементов, то перемещение будет осуществляться всей группы. Это же можно выполнить нажатием кнопки панели инструментов Переместить блок (***Block Move***)  , а также выбрав команду из контекстного меню, вызвав его ПКМ.

Вращение выделенного элемента или группы элементов выполняется нажатием кнопки панели инструментов Вращать блок (***BlockRotate)***  , или выбрав соответствующую команду из контекстного меню, вызываемого ПКМ.

Копирование элементов на экране выполняется кнопкой панели инструментов Копировать блок (***Block Copy***)  или, выбрав соответствующую команду из контекстного меню, вызываемого ПКМ.

Удаление элементов или группы элементов с экрана выполняется кнопкой панели инструментов Удалить блок (***Block Delay***)  или, выбрав соответствующую команду из контекстного меню, вызываемого ПКМ. Кроме того, это можно сделать клавишей ***Delate***.

Связи создаются на схеме при наличии курсора в виде карандаша путем щелчка ЛКМ на розовом квадратном маркере, который появляется в местах, доступных для соединения. Линия протягивается от начала до следующей точки соединения. Линия может фиксироваться в местах искривления нажатием ЛКМ в процессе прокладки. Удаление связи предполагает выделе-

ние ее ПКМ и выбор в контекстном меню команды Удалить проводник (***Delate Wire***). Альтернативой может быть двойное нажатие ПКМ.

Надписи на схеме можно выполнить с помощью кнопки Текстовый режим ***Text Script Mode***  . Это значительно повышает читабельность схемы.

Настройка свойств компонентов схемы

Напряжение питания схемы выбирается в окне Конфигурации шин, перейдя по ***Design/Configure Power Rails***. В открывшемся окне конфигурации (рис.3.6) устанавливается тип источника ***VCC/VDD*** и величина питающего напряжения. Эти параметры допустимо изменять при выключенном режиме моделирования.

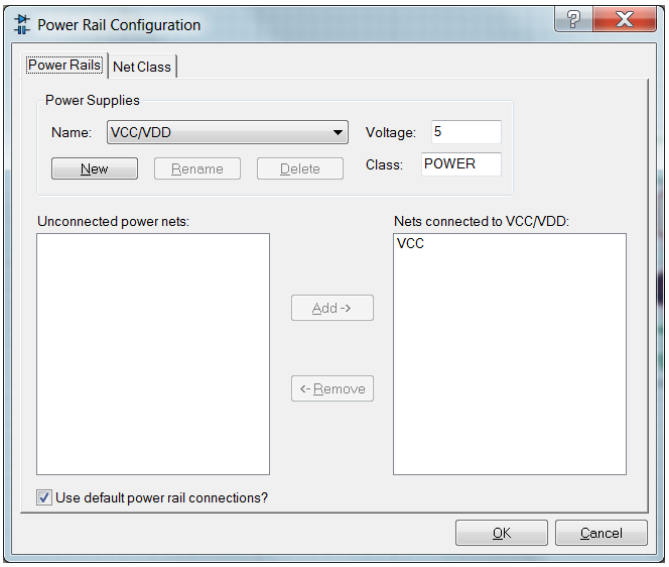


Рис.3.6. Окно конфигурации шин питания

Параметры элементов схемы редактируются в окне редактирования в объеме свойств, характерных для конкретного элемента. Окно свойств выводится при нажатии ПКМ и выборе команды ***Edit Properties*** или двойным щелчком ЛКМ. В это же окно можно переместиться, задав позиционное обозначение элемента через ***Edit/(Find/Edit Component).***

Программирование микроконтроллера в Proteus возможно как созданием исходного текста программы с последующей компиляцией (в демоверсии эта опция недоступна), так и загрузкой уже откомпилированного файла. Для загрузки файла необходимо в окне редактирования свойств микроконтроллера ***Edit Component*** прописать путь к выбираемому файлу (рис.3.7). Делается это в поле Программного файла ***Program File***. Выбрать можно объектный файл (расширение ***.obj***) или hex-файл (расширение **.hex**). Отличие заключается в том, что в режиме отладки при загрузке файла с расширением ***.obj*** в окне исходного кода можно наблюдать исходный текст программы (***Debug/AVR/Source Code***).

В поле ***Program File*** нужно указать форматы:

.cof - если вы хотите вести отладку по исходному тексту программы на языке Си в компиляторе CodeVisionAVR - CVAVR;

.elf - если вы хотите вести отладку по исходному тексту на языке Си программы компилятора WinAVR - это отличный компилятор, БЕСПЛАТНЫЙ и профессиональный, но новичкам немного затруднительно его использовать;

UBROF - формат файла если вы хотите вести отладку по исходнику на языке Си программы, созданной в компиляторе IAR - это самый лучший компилятор для AVR, но новичкам ОЧЕНЬ затруднительно его использовать.

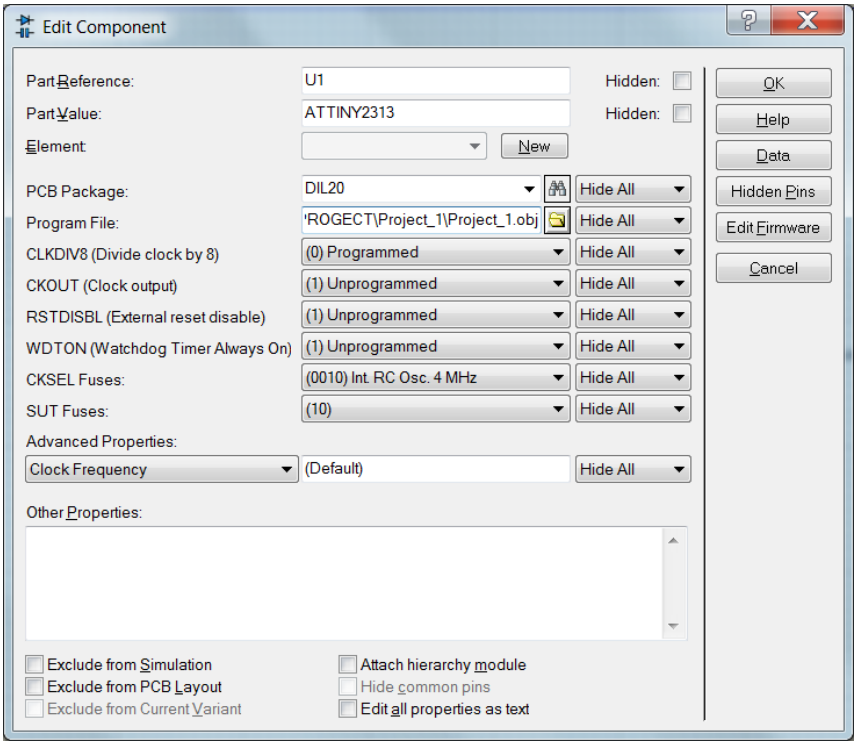


Рис.3.7. Окно свойств микроконтроллера

В окне редактирования свойств можно изменить все остальные свойства микроконтроллера, в том числе установки фьюз-битов.

После выполнения всех процедур формирования схемы внешний вид рабочего окна должен быть подобным тому, который представлен на рис.3.8.

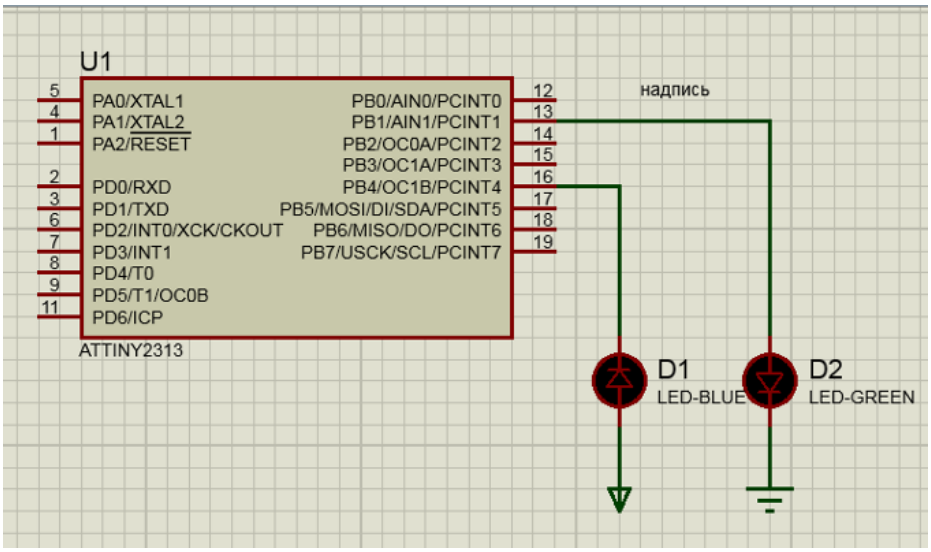


Рис.3.8. Схема соединения микроконтроллера

В отчете представить:

1. Состав параметров моделируемых элементов: микроконтроллера, вольтметра, резистора, светодиода, терминаторов;

2. Способы доступа к окну свойств компонент. Выполняемые технологические последовательности можно показать графически в виде схемы операций, как это сделано на рис.3.9.

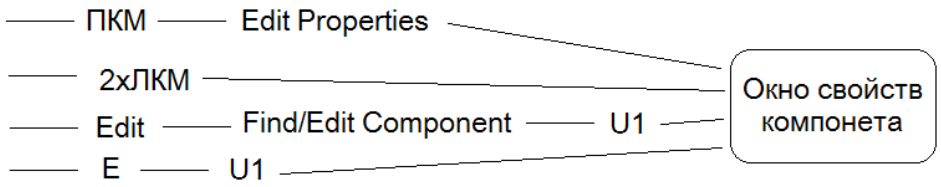


Рис.3.9. Схема операций для доступа в окно Редактирование свойств компонента

Задание 3.3. Моделирование схемы и отладка

Практические задачи:

1. Запустить схему на моделирование и оценить правильность ее поведения;
2. Осуществить пошаговое выполнение программы. Реализовать все варианты режима отладки.
3. Подключить светодиоды ко всем выходам порта. Дополнить программу так, чтобы в пошаговом режиме было видно упорядоченное движение светящихся светодиодов.

Моделирование электронных схем

Моделирование осуществляется под управлением кнопок управления моделированием (рис.3.10). Последовательно на рисунке слева направо размещены:

Рис.3.10. Кнопки управления моделированием

- запуск (продолжение) моделирования или симуляции (***Run Simulation) F12***;
- пошаговое моделирование (***Step Into Source Line) F11***. Обычно, одному шагу соответствует одна команда ассемблера;
- пауза симуляции (***Pause***);
- остановка симуляции (***Stop VSM Debugging) Esc***.

Аналогичные режимы можно инициировать, выполнив переход из меню ***Debug***.

В процессе моделирования выводы элементов подсвечиваются квадратиками разных цветов:

- красный - логическая "1" (высокий потенциал);
- синий - логический "0" (низкий потенциал);

- серый – не определено (высоко-импедансное или Z- состояние);

- желтый - конфликт на линии. Например, на заземленный проводник осуществляется попытка подать +5v.

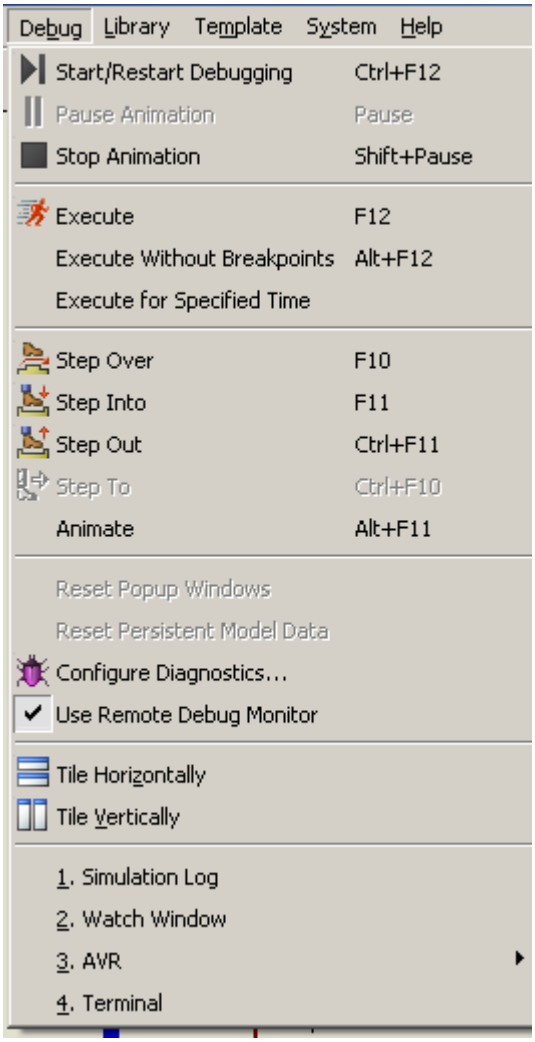


Рис.3.11. Меню Отладчика

Пошаговая отладка осуществляется из меню ***Debug*** (рис.3.11). В этом режиме каждый новый запуск (***F11***) приводит к выполнению только одной команды программы. Отклонения от этого правила возможны:

- ***Step Over Source Line (F10)*** - позволяет выполнить подпрограммы без пошагового режима. Вариант при отладке допустим при наличии проверенных и отлаженных подпрограмм;

- ***Step Out from Source Line (Ctrl+F11)*** - позволяет быстро закончить выполнение подпрограммы (выйти из нее) без пошагового режима.

Точки останова позволяют остановить процесс симуляции на выбранной команде. Для установки точки следует два раза щелкнуть ЛКМ на выбранном операторе (строке) программы. Должна появиться закрашенная точка. Удаление точки выполняется таким же двойным нажатием. Но точка при этом становится неактивной и не останавливает процесс симуляции. Для полного удаления точки необходимо нажать ее еще раз. Аналогичные и другие действия с точками можно инициировать из контекстного меню, вызываемого на выбранной точке.

Контроль состояния микроконтроллера осуществим в режиме отладки. При этом из меню ***Debug\AVR*** можно получить доступ к следующим окнам отладчика (рис.3.12):

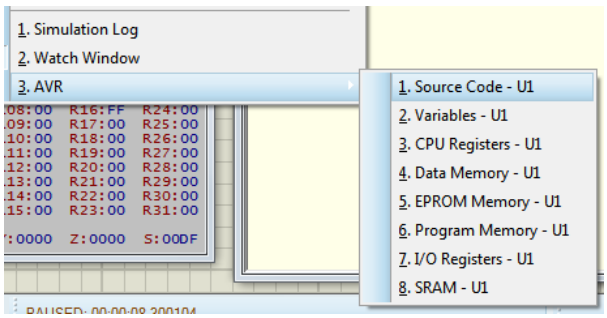


Рис.3.12. Список окон отладчика

- Окно просмотра ***(Watch Window)***. В этом окне (рис.3.13) можно размещать имена регистров МКК или адреса памяти и отслеживать их содержимое по ходу программы. Для добавления объекта необходимо в контекстном меню (выбирается ПКМ) выбрать либо

объект (***Add Item By Name***) ***Alt+N*** либо адрес памяти (***Add Item By Address***) ***Alt+A***. При выборе по имени выпадает весь список возможных имен. С помощью квадратика при имени уже внесенного в окно регистра можно развернуть регистр на группы функционально связанных битов.

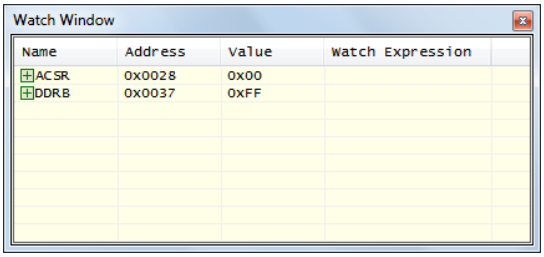


Рис.3.13. Окно просмотра

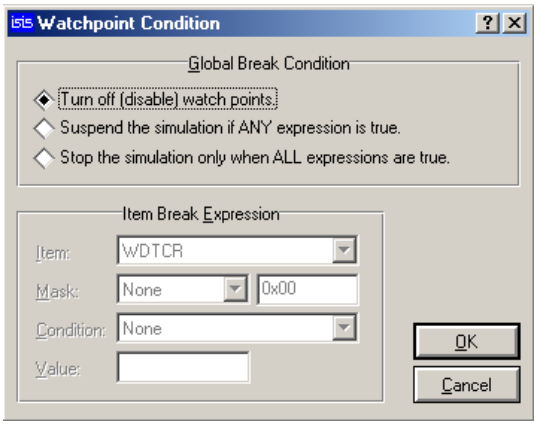


Рис.3.14. Окно параметров точки останова

В окне можно не только наблюдать переменные, но и задавать некоторые условия и действия при достижении этих условий - например, остановить симуляцию. По сути это точки останова. Для этого в контекстном меню выбирается окно Состояние контрольной точки (***WatchpointCondition)***. В окне (рис.3.14) можно отключить контроль заданного условия, остановить симуляцию при выполнении одного из условий или при выполнении всех условий;

- Исходный код микроконтроллера ***(Source Code***). Это окно (рис.3.15) доступно только при загрузке готового файла контроллера с расширением ***.obj*** или при создании файла в среде Proteus. При наличии этого окна на экране очередной шаг процессора будет изменять положение курсора в окне. Курсор будет указывать на выполняемую сейчас команду;

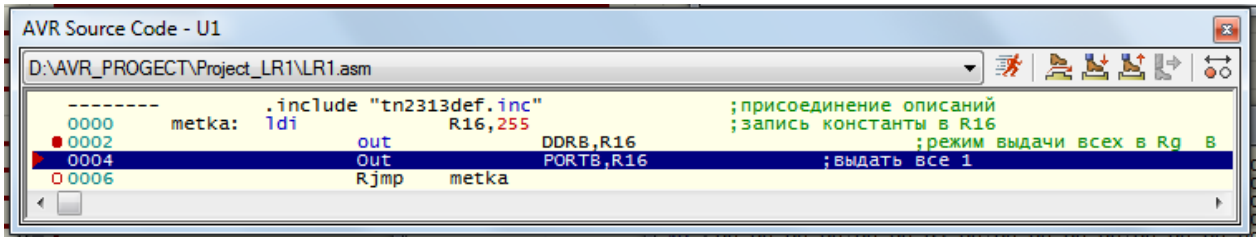


Рис.3.15. Окно исходного текста программы

- Переменные (***Variable***). Окно позволяет отследить список имен переменных, заявленных в программе, контролировать всю динамику изменения их значений (***Valie***);

- Регистры процессора (***CPU Registers***). Это окно (рис.3.16) показывает содержание РОН (***R0-R31***), счетчика команд (***PC***), шестнадцатеричный и двоичный код регистра статуса (флагов) ***SREG,*** также количество выполненных циклов (***Cycle Count***), значения индексных регистров X, Y, Z;

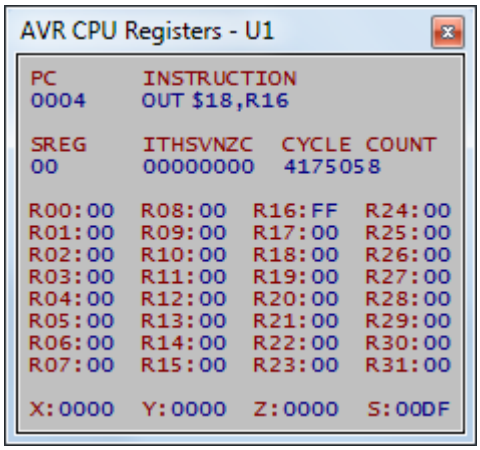


Рис.3.16. Окно регистров процессора

- Память данных (***Data Memory*** или ***AVR Sram***). В этом окне (рис.3.17) отображается содержимое ОЗУ;

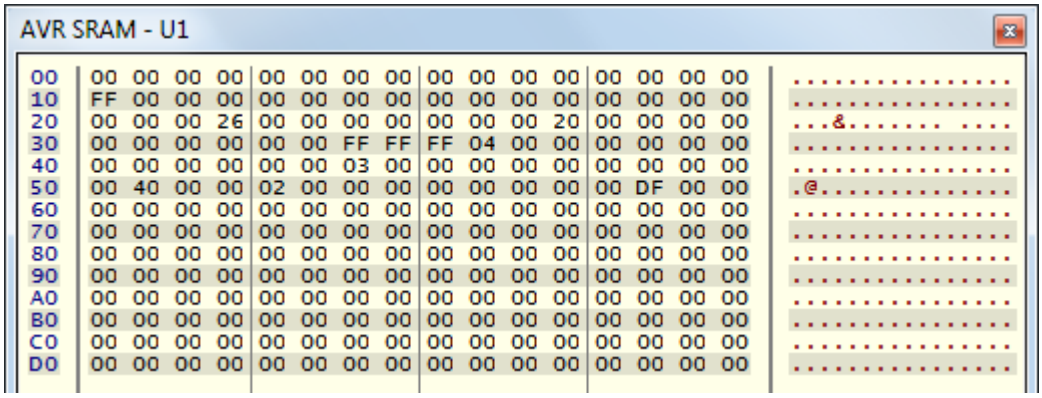


Рис.3.17. Окно Память данных

- Постоянная память данных (***EPROM Memory***);
- Память программ (***Program Memory***);
- Регистры ввода-вывода (***AVR I/O Registers***). В окне (рис.3.18) отображаются регистры ввода-вывода, управляющие внешними устройствами контроллера.

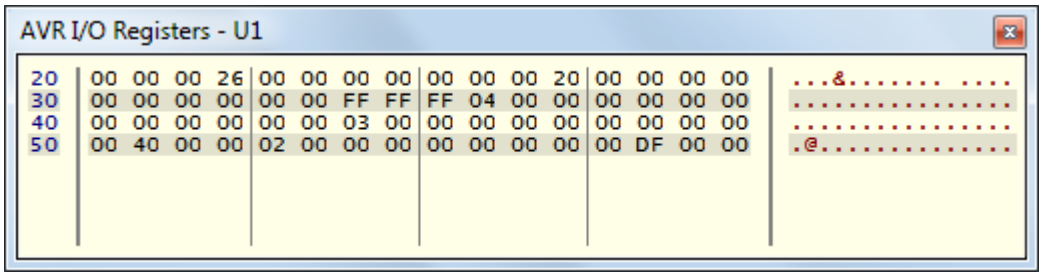


Рис.3.18. Окно регистров РВВ

В отчете представить:

1. Доработанную программу МКК;
2. Допустимые способы изменения исходного текста программы МКК.

Вопросы для самоконтроля

1. Какие режимы работы Proteus существуют?
2. Какие параметры проекта выбираются при старте системы?
3. Как осуществляется выбор компонентов для набора схемы?
4. Назовите приемы редактирования элементов схемы.
5. Как изменить свойства компонентов схемы?
6. Для чего используется цифровой и аналоговый режимы работы светодиодов? Чем они отличаются?
7. Какие объекты можно наблюдать в режиме отладчика?
8. Как изменить программный код отлаживаемой программы?
9. С какой целью используют точки останова? Как точку поставить и удалить? Как удалить временно?
10. Назовите режимы работы отладчика?
11. Как можно изменить исходный текст программы в Proteus?
12. Чем отличается среда отладки Proteus от AVR Studio?
13. Найдите значение регистра данных порта B в окне регистров ввода-вывода и Окне ОЗУ. Сравните значение данных.
14. Измерьте напряжение на выводах микроконтроллера в состоянии логического нуля и единицы.