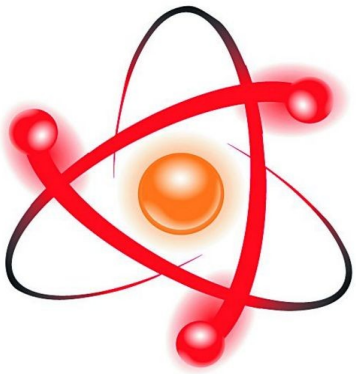
НПО УЧЕБНОЙ ТЕХНИКИ «ТУЛАНАУЧПРИБОР»

МЕТОДИЧЕСКОЕ РУКОВОДСТВО ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ



РТМТЛ-4

ЯЗЫКИ ПРОГРАММИРОВАНИЯ И СОЗДАНИЕ ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ.

Тула, 2014 г.

ЛАБОРАТОРНАЯ РАБОТА.
ЯЗЫКИ ПРОГРАММИРОВАНИЯ И СОЗДАНИЕ ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ. АВТОМАТИЗИРОВАННЫЙ ЛАБОРАТОРНЫЙ КОМПЛЕКС ДЛЯ РАБОТЫ С ПК.

Цель работы: на примере микропроцессора AVR Atmega8535 (либо Atmega16) изучить основные приёмы программирования микроконтроллеров на различных языках.

ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ.
Программирование микроконтроллеров.

Термин программирование микроконтроллеров обозначает процесс записи (программирования) информации в постоянное запоминающее устройство (ПЗУ) микроконтроллера. В общем случае, помимо программирования микроконтроллеров, в практике встречается программирование микросхем (МС) памяти и программирование логических матриц. Как правило, программирование микроконтроллеров и микросхем памяти производится при помощи специальных устройств – программаторов. Хороший программатор позволяет не только программировать (записывать), но и считывать информацию, а в ряде случаев, производить и другие действия с МС и информацией находящейся в ней (стирание, защита от чтения, защита от программирования и т. п.).

Используя различные признаки, все многообразие МС со встроенным ПЗУ можно систематизировать следующим образом:

1. **По функциональному назначению**

1.1. Микросхемы памяти;

1.2. Микроконтроллеры с внутренним ПЗУ;

1.3. Микросхемы программируемой логики (программируемые матрицы).

2. **По возможности программирования**

2.1. Однократно программируемые - устройства допускающие единственный цикл программирования;

2.2. Многократно программируемые (перепрограммируемые) - устройства допускающие множество циклов программирования (перепрограммирования).

3. **По допустимым способам программирования.**

3.1. Микросхемы, программируемые в специальном устройстве – программаторе. Для осуществления необходимой операции (запись, стирание, чтение, верификация и т.п.), подобные мс. вставляются в специальную колодку программатора, обеспечивающую электрический контакт со всеми выводами микросхемы. Для реализации выбранного режима, программатор формирует в соответствии со спецификацией производителя необходимые последовательности сигналов, которые через колодку подаются на определенные выводы программируемого микроконтроллера (микросхемы).

3.2. Микросхемы, поддерживающие режим внутрисхемного программирования (“ISP mode”), и программируемые непосредственно в плате пользователя.

Подобные микросхемы предполагают выполнение необходимой операции (программирование, стирание, чтение, верификация и т.п.) непосредственно в плате пользователя. Все действия по программированию (стиранию, чтению, верификации и т.п.) производятся с помощью внешнего программатора, определенным образом подключенного к плате пользователя. При этом плата пользователя должна быть разработана с учетом специфических требований данного режима.

3.3. Микросхемы, поддерживающие режим внутреннего самопрограммирования. Подобные микросхемы допускают выполнение необходимой операции (запись, стирание, чтение, верификация и т.п.) непосредственно в устройстве пользователя, без использования какого либо программатора. При этом устройство пользователя должно быть разработано с учетом специфических требований данного режима.

Программирование микроконтроллера подразумевает заполнение внутренней памяти микроконтроллера нужной информацией. В зависимости от типа программируемого микроконтроллера, внутренняя память микроконтроллера обладает своей структурой и организацией. В общем случае, внутренняя память микроконтроллера это: память данных, память программ, регистры специального назначения (fuse - биты) - содержимое которых определяет режимы работы микроконтроллера и/или его периферии. Таким образом: программирование микроконтроллера - это заполнение каждой области памяти своей специфической информацией.

Каждый программируемый микроконтроллер обладает своим индивидуальным набором допустимых режимов: программирование (запись), чтение, стирание, защита от чтения, защита от программирования и т.п.

Некоторые программируемые микроконтроллеры не имеют отдельного режима «стирание». Для них стирание прежней информации в памяти происходит в теневом режиме, при каждом новом цикле программирования микроконтроллера;

Некоторые программируемые микроконтроллеры поддерживают различные режимы ограничения доступа. Выбор режима ограничения доступа производится при программировании микроконтроллера. В зависимости от выбранного режима, либо все ПЗУ микроконтроллера, либо его определенные части могут быть:

1. - защищены от возможности записи/дозаписи;
2. - защищены от возможности считывания содержимого извне. При попытке считать информацию, защищенный микроконтроллер будет выдавать либо «мусор», либо «все 0», либо «все 1».

Говоря о программируемых устройствах, можно считать общепринятой

следующую систему мнемонических обозначений:

1. PROM (*Programmable Read-Only Memory)* - программируемая пользователем энергонезависимая память (ПЗУ).
2. EPROM (*Erasable Programmable Read-Only Memory)* - перепрограммируемое ПЗУ. Стирание содержимого производится при помощи ультрафиолетовых лучей, после облучения подобное ПЗУ готово к новому циклу записи информации (программированию). Устаревший тип памяти.
3. EEPROM (*Electrically Erasable Programmable Read-Only Memory)* - электрически стираемое перепрограммируемое ПЗУ. Память такого типа может стираться и заполняться данными многократно, от несколько десятков тысяч раз до миллиона.
4. FLASH (*Flash Memory*) - одна из технологических разновидностей энергонезависимой перезаписываемой памяти.
5. NVRAM (Non-volatile memory) - «неразрушающаяся» память, представляющая собой ОЗУ со встроенным источником электропитания. По своей функциональности для пользователя – аналогична традиционному ПЗУ.
6. PLD *(Programmable Logic Device)* - Программируемая логическая интегральная схема. (ПЛИС).
7. MCU (Microcontroller Unit) – микроконтроллер.

Микроконтроллеры могут быть запрограммированы двумя способами – по параллельному интерфейсу и по последовательному. «Параллельное» **программирование** более сложное в плане реализации программатора и самого **программирования**, но зато имеет немного большие возможности (например прошивка контроллера с отключенным ресетом). **Последовательное программирование** (SPI programming) очень легко реализуется, не требует повышенного напряжения, работает даже если **микроконтроллер** уже впаян в рабочую схему – это и называется **внутрисхемным программированием** (ISP – In System Programmer).

Языки программирования микроконтроллеров.

Все современные микроконтроллеры относятся к классу микропроцессорных устройств. Главным принципом действия таких элементов является исполнение последовательного потока команд, называемого программой. Микроконтроллер получает команды в виде отдельных машинных кодов. Между тем, для создания и отладки программ, машинные коды подходят плохо, так как трудно воспринимаются человеком. Этот факт привел к появлению различных языков программирования.

Языки программирования микроконтроллеров по своей структуре мало отличаются от классических языков для компьютеров. Единственным отличием становится ориентированность на работу со встроенными периферийными устройствами. Архитектура микроконтроллеров требует, например, наличия битово-ориентированных команд. Последние позволяют выполнять работу с отдельными линиями портов ввода/вывода или флагами регистров. Подобные команды отсутствуют в большинстве крупных архитектур. Даже ядро ARM, активно применяемое в микроконтроллерах, не содержит битовых команд, вследствие чего разработчикам пришлось создавать специальные методы битового доступа.

Ассемблер

Ассемблер является языком самого низкого уровня. При этом он позволяет наиболее полно раскрыть все возможности микроконтроллеров и получить максимальное быстродействие и компактный код. В некоторых случаях альтернативы ассемблеру нет, но тем не менее он имеет множество недостатков. Несмотря на получаемую компактность машинного кода, программа, написанная на языке Ассемблер, громоздка и труднопонимаема. Для ее создания требуется отличное знание архитектуры и системы команд микроконтроллеров.

Ассемблер отлично подходит для программирования микроконтроллеров, имеющих ограниченные ресурсы, например 8-ми битных моделей с малым объемом памяти. Для больших программ и тем более 32-разрядных контроллеров, лучше использовать другие языки, отличающиеся более высоким уровнем. Это позволит создавать более сложные и при этом понятные программы.

C/C++

Язык программирования C/C++, относится к языкам более высокого уровня, по сравнению с Ассемблером. Программа на этом языке лучше понятна человеку. Достоинством C/C++ является огромное число программных средств и библиотек, позволяющих просто создавать необходимый код. Фактически, C/C++ сегодня стал основным языком разработки управляющих программ. Компиляторы данного языка реализованы практически для всех моделей

микроконтроллеров. Стандартный язык дает возможность переноса программ с одной платформы на другую. Теоретически, используя разные компиляторы, можно преобразовать любую программу в команды микроконтроллера нужного типа. На практике дополнительно требуется учитывать архитектуру микроконтроллера каждого типа.

Язык C/C++ имеет достаточно сложную для изучения структуру. Получаемый программный код конкретной задачи, имеет больший объем, чем код той же задачи, реализованной на Ассемблере. Тем не менее язык C/C++ следует признать единственным правильным выбором для профессионального программирования микроконтроллеров.

```
 // Пример программы на языке C
  // Мигание встроенным светодиодом Arduino
void setup() {
  pinMode(13, OUTPUT);     // Инициализация выхода 13
}
void loop() {
  digitalWrite(13, HIGH);   // Зажечь светодиод
  delay(1000);              // Задержка
  digitalWrite(13, LOW);    // Зажечь светодиод
  delay(1000);              // Задержка
}
```

Pascal

Язык Pascal еще более удобен для восприятия и изучения. Тем не менее, он не имеет такого распространения как C/C++, особенно при программировании микроконтроллеров. Некоторые отдельные фирмы поддерживают данный язык, с целью упрощения перехода на контроллеры с больших ПК. В частности вариант языка под названием MicroPASCAL входит в состав поставки отладочных средств фирмы Mikroelektronika.

```
// Пример программы на языке MicroPASCAL
// Мигание светодиодом
program LED_Blinking;
begin
  PORTC := 0;                 // Инициализация PORTC
  TRISC := 0;                 // Настройка PORTC
  while TRUE do               // Начало бесконечного цикла
  begin
    PORTC := not PORTC;      // Инвертирование PORTC
    Delay_ms(1000);          // Задержка
  end;
end.
```

BASIC

Старинный язык первоначального обучения программированию, в настоящее время в основном сохранился в виде реализации Visual BASIC от Microsoft. Используется он и для программирования микроконтроллеров. Реализаций этого языка гораздо больше, чем того же Pascal. Связано это в первую очередь с простотой языка. BASIC часто выбирают разработчики программно-аппаратных платформ, нацеленных на упрощенную разработку

электронных устройств. Можно назвать такие проекты, как PICAXE, Amicus18, microBASIC и некоторые другие. Недостатком BASIC является плохая структурированность кода. Этот язык не стоит выбирать для первоначального изучения с целью дальнейшего перехода на C/C++. Программирование микроконтроллеров на BASIC можно рекомендовать любителям, не нацеленным на создание, в основном, простых устройств.

```
' Пример программы на ProtonBASIC
' Мигание светодиодом на PORTB.0 Amicus18.
    While 1 = 1             ' Начало бесконечного цикла
        High RB0            ' Включить PortB.0
        DelayMS 500         ' Задержка полсекунды
        Low RB0             ' Выключить PortB.0
        DelayMS 500         ' Задержка полсекунды
    Wend                    ' Закрытие цикла
```

Визуальные языки. Графический Ассемблер.

В отличие от классических языков программирования, визуальные языки позволяют разрабатывать программы в виде изображений рис. 1.1. Среди таких языков можно выделить FlowCODE, Scratah, а также графический ассемблер, реализованный в учебной среде программирования **Algorithm Builder.**

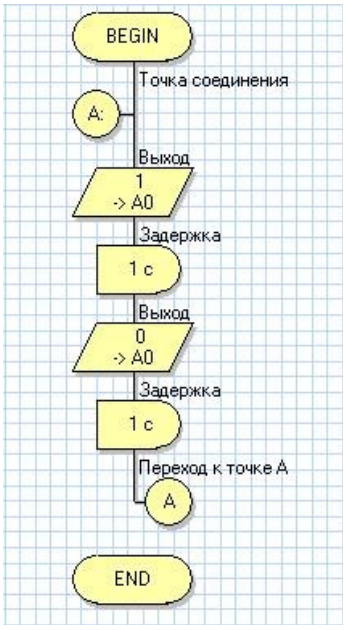


Рис. 1.1

Достоинством визуальных языков является хорошо воспринимаемая структура алгоритма. Это позволяет просто разобраться в его функционировании любому человеку, знающему основные символы языка. Перевод структурных схем в команды микроконтроллера, как правило, выполняется не сразу. Вначале алгоритм транслируется в команды ассемблера или какого-либо языка высокого уровня. Только затем, все преобразуется в машинный код. Такая схема, несмотря на свою сложность, позволяет использовать наиболее удобные компиляторы разных разработчиков.

Еще одним достоинством визуального программирования становится простота изучения, поэтому подобные языки часто используются для обучения детей. Недостатком визуального подхода является громоздкость исходных материалов. Тем не менее, подобные языки программирования нашли очень большое распространение для решения специальных задач.

Компиляторы.

Чтобы преобразовать исходный текст программы в файл прошивки микроконтроллера, применяют компиляторы.

Фирма Atmel поставляет мощный компилятор ассемблера, который входит в среду разработки AVR Studio, работающую под Windows. Наряду с компилятором, среда разработки содержит отладчик и эмулятор. AVR Studio совершенно бесплатна и доступна на сайте Atmel.

В настоящее время представлено достаточно много компиляторов Си для AVR. Самым мощным из них считается компилятор фирмы IAR Systems. Именно ее сотрудники в середине 90-х годов участвовали в разработке системы команд AVR. IAR C Compiler имеет широкие возможности по оптимизации кода и поставляется в составе интегрированной среды разработки IAR Embedded Workbench (EWB), включающей в себя также компилятор ассемблера, линкер, менеджер проектов и библиотек, а также отладчик.

Фирмой Image Craft выпускается компилятор языка Си, получивший достаточно широкую популярность. Image Craft C Compiler обладает неплохим уровнем оптимизации кода.

Не меньшую популярность завоевал Code Vision AVR C Compiler. Компилятор поставляется вместе с интегрированной средой разработки, в которую, помимо стандартных возможностей, включена достаточно интересная функция - CodeWizardAVR Automatic Program Generator. Наличие в среде разработки последовательного терминала позволяет производить отладку программ с использованием последовательного порта микроконтроллера.

Поистине культовой стала интегрированная среда разработки WinAVR. Она включает мощные компиляторы Си и ассемблера, программатор AVRDUDE, отладчик, симулятор и множество других вспомогательных программ и утилит. WinAVR прекрасно интегрируется со средой разработки AVR Studio от Atmel. Ассемблер идентичен по входному коду ассемблеру AVR Studio. Компиляторы Си и ассемблера имеют возможность создания отладочных файлов в формате COFF, что позволяет применять не только встроенные средства, но и использовать мощный симулятор AVR Studio. Еще одним немаловажным плюсом является то, что WinAVR распространяется свободно без ограничений (производители поддерживают GNU General Public License).

Отдельно следует выделить среду графического программирования **Algorithm Builder.** Среда предназначена для производства полного цикла разработки начиная от ввода алгоритма, включая процесс отладки и заканчивая программированием кристалла. Разработка программы может быть как на уровне ассемблера, так и на макроуровне с манипуляцией многобайтными величинами со знаком. В отличие от классического ассемблера программа вводится в виде алгоритма с древовидными ветвлениями и отображается на плоскости, в двух измерениях. Сеть условных и безусловных переходов отображается графически, в удобной векторной форме. Это к тому же освобождает программу от бесчисленных имен меток, которые в классическом ассемблере являются неизбеж-

ным балластом. Вся логическая структура программы становися наглядной. Графические технологии раскрывают новые возможности для программистов. Визуальность логической структуры уменьшает вероятность ошибок и сокращает сроки разработки.

В качестве резюме стоит сказать, что WinAVR и Algorithm Builder, наверное, являются идеальным выбором для тех, кто начинает осваивать микроконтроллеры AVR.

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ.
Приборы и оборудование.

Комплекс лабораторных работ по исследованию программирования микроконтроллеров на различных языках программирования выполняется на учебном макетном лабораторном комплексе РТМТЛ-4.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микроконтроллер (Atmega8535, Atmega16, Atmega32 или совместимый); резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; набор из 8 светодиодов, подключенных к выводам 33 – 40, см. рис. 2.1; набор из 8 кнопок с фиксацией нормально разомкнутых (кнопка нажата — замкнуто; отжата — разомкнуто), подключенных к выводам 22 — 29; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к COM – порту ПЭВМ ТОЛЬКО проводом типа COM 9m/9f («мама» - «папа»).

В данной работе на примере микроконтроллера AVR (Atmega8535, Atmega16, Atmega32 или совместимого) предлагается освоить простейшие приемы программирования на следующих языках:

1) Программирование микроконтроллера на языке ассемблер в интегральной среде разработки AVR Studio
2) Программирование микроконтроллера на языке Basic в интегральной среде разработки Bascom-AVR
3) Программирование микроконтроллера на языке Си в интегральной среде разработки AVR Studio с подключенным Си компилятором GCC из WinAVR
4) Программирование микроконтроллера в среде графического ассемблера Algorithm Builder.

Естественно, что в данном кратком методическом руководстве невозможно описать все тонкости работы с той или иной средой программирования или с тем или иным языком программирования. Поэтому, перед началом работы необходимо ознакомиться с документацией на исследуемый микроконтроллер, документацией на исследуемую среду программирования, а также со статьями по данным вопросам, размещенными в соответствующих папках на прилагаемом к учебной установке диске или Flash карте.

Принципиальная электрическая схема прибора приведена на рис. 2.1.

Демонстрационный пример готовой программы, на основе которого могут быть построены задания по программированию микроконтроллера, приведены в папке Examples_RTMTL-4. Пример выполнен на четырех распространенных языков программирования для микроконтроллера (ASM, Си, Basic, Графический ASM в среде Algorithm Builder).

1) Пример 1. В данном примере кнопки с фиксацией подключены к порту A (выводы PA0-PA7), логическое состояние которого представляет собой 1 байт данных. При этом состояние нажатой кнопки это логический 0, состояние отжатой кнопки — логическая 1 соответствующего вывода. К данным порта A прибавляется константа (число 2 в среде графического ассемблера Algorithm Builder; число 3 в среде разработки AVR Studio ассемблер; число 4 в среде разработки Bascom-AVR и число 5 в среде разработки AVR Studio Си с подключенным компилятором GCC из WinAVR). Полученный результат копируется в порт C (PC0-PC7), к которому подключены светодиоды, визуализирующие логическое состояние соответствующих выводов (PC0-PC7).

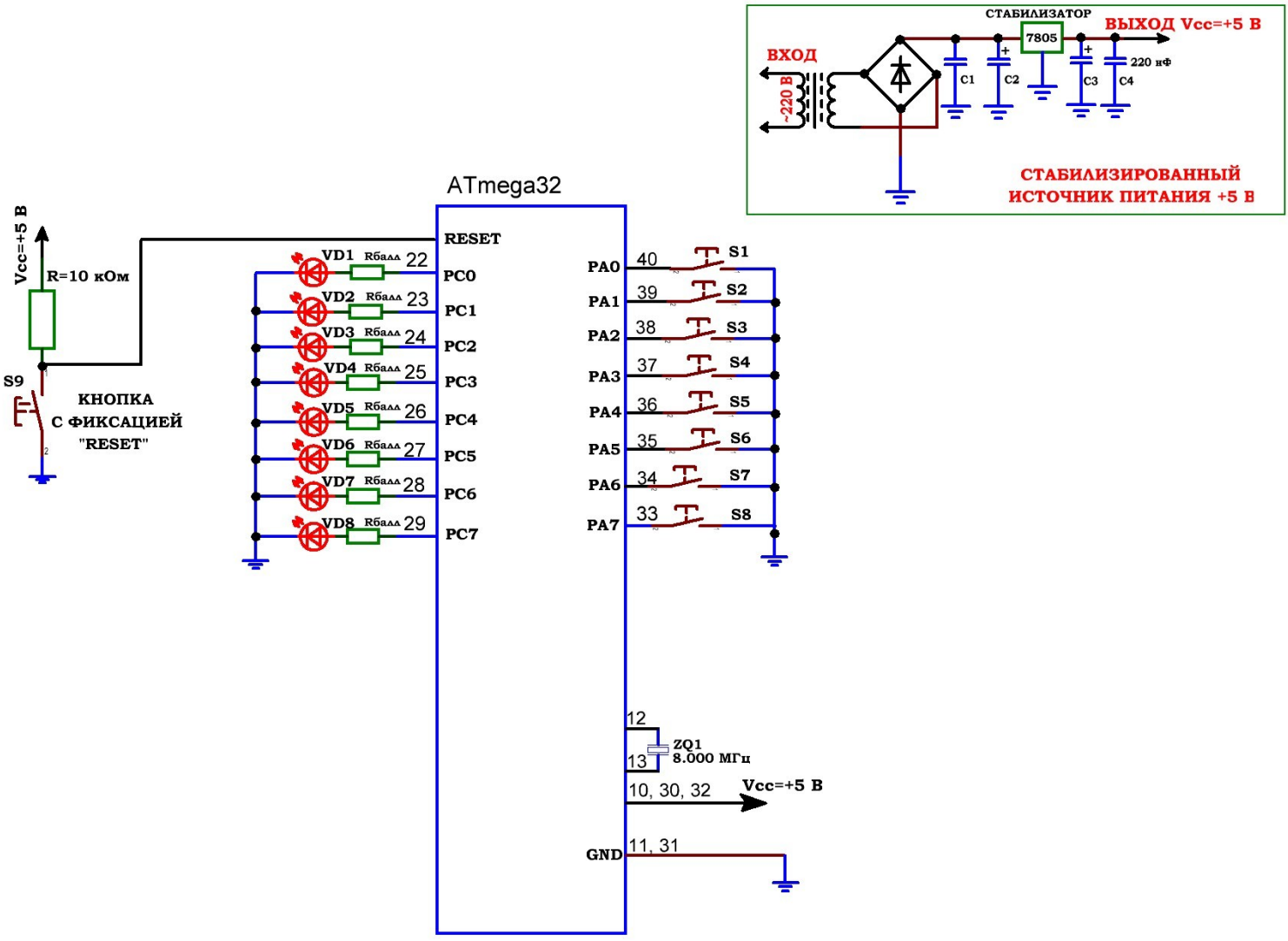


Рис. 2.1. Принципиальная электрическая схема учебного стенда для изучения языков программирования на примере микроконтроллеров серии Atmega8535/16/32.

Для прошивки откомпилированного HEX-файла используется графическая оболочка SinaPROG для программы AVRdude, включающая в себя простой и функциональный AVR fuse-калькулятор рис. 2.2.

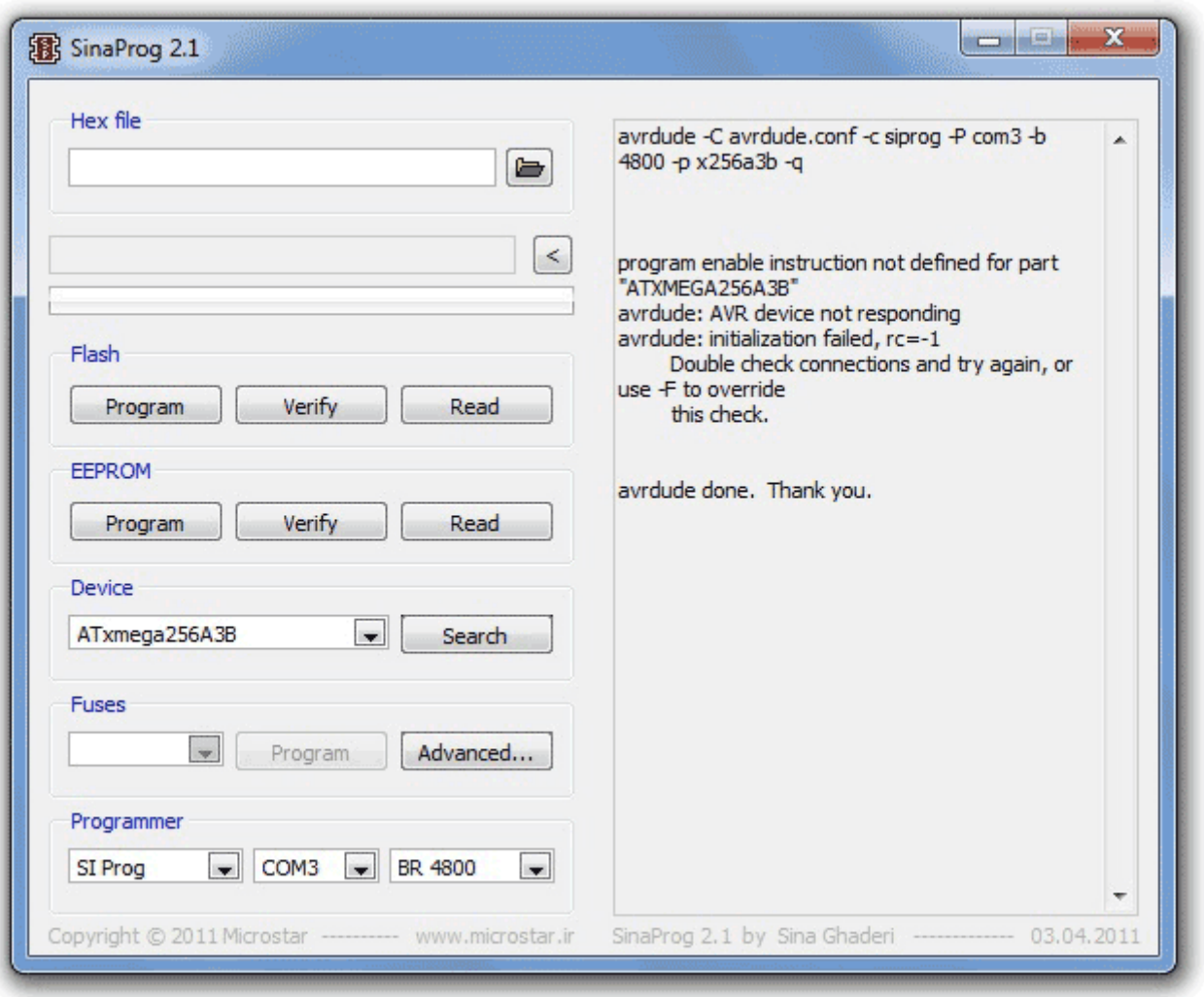


Рис. 2.2. Графическая утилита SinaPROG для программирования микроконтроллеров.

В отличие от аналогичного программного обеспечения SinaProg имеет поддержку огромного количества программаторов. Она работает одинаково хорошо и с мощными атмеловскими устройствами, и с простейшими аппаратами, состоящими всего из нескольких проводков. Любой поддерживаемый через AVRdude программатор, даже если его нет в списке доступных устройств SinaProg, легко может быть добавлен в базу и настроен.

Интерфейс графической среды выглядит просто и удобно, нет ничего лишнего, все понятно и предельно ясно. Разработчику предлагается выбрать необходимый для прошивки hex-файл и указать нужную память – EEPROM или Flash. Для работы с памятью доступны операции: программирования, проверки (верификация содержимого памяти и hex-файла) и чтения. Помимо этого присутствует поле выбора микроконтроллера из выпадающего списка, с которым будет работать программатор. Для контроля работоспособности линии «программатор-шлейф-микроконтроллер» можно запустить процесс проверки соответствия выбранного микрочипа, тому, что подсоединен в реальности. В отдельном окне предлагается установить тип программатора, порт к которому он подключен и скорость его работы. Индикатор состояния показывает прогресс выполнения

операций с микроконтроллером. Для поиска ошибок присутствует возможность просмотреть логи системных сообщений.

В программе SinaProg присутствует целая секция, предназначенная для установки и конфигурации fuse-битов рис. 2.3. Данные автоматически считываются с микроконтроллера. Утилита позволяет посмотреть: сигнатуру микропроцессора, калибрационные величины для генератора, fuse-биты, разделенные на четыре байта (High Fuse, Lock Bits, Low Fuse и Ext. Fuse). Кроме этого программа выдает удобные для анализа и редактирования выпадающие списки с подробным описанием fuse-битов – так называемый fuse-калькулятор. Запись fuse-битов возможна либо в виде значений всего байта (шестнадцатеричное число), либо при помощи fuse-калькулятора с выбором нужных режимов работы. При необходимости все описания fuse-битов можно легко русифицировать. Все предустановки описываются в простом формате в файле Fuse.txt. Также необходимо отметить, что fuse-биты в данной среде являются инверсными.

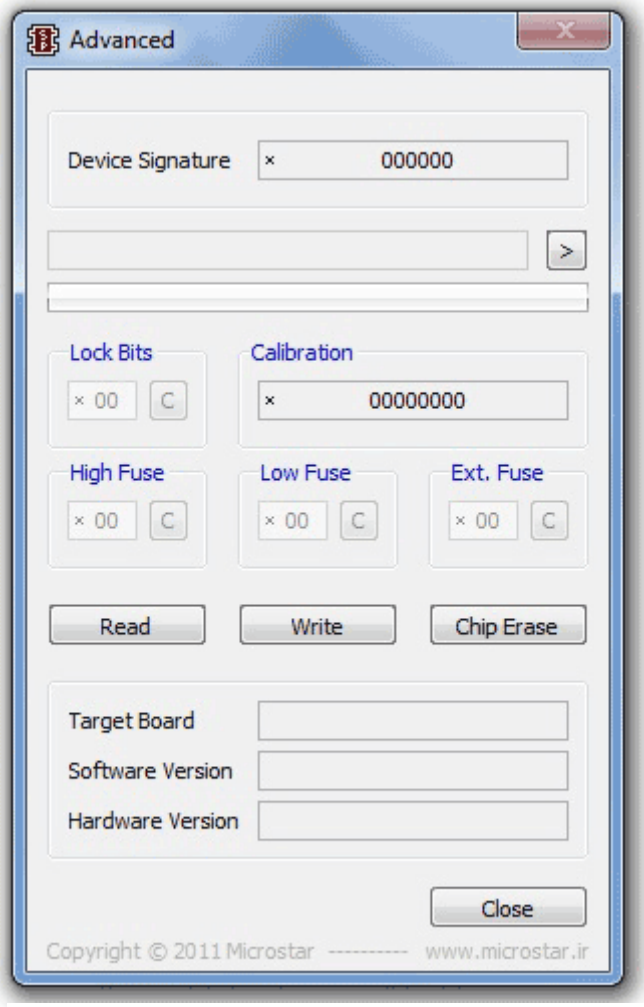


Рис. 2.3. Графическая утилита SinaPROG. Секция, предназначенная для установки и конфигурации fuse-битов

Порядок выполнения.

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.
2. **Перед началом работы необходимо ознакомиться с документацией на исследуемый микроконтроллер, документацией на исследуемую среду программирования, а также со статьями по данным вопросам, размещенными в соответствующих папках на прилагаемом к учебной установке диске или Flash карте.**
3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.
4. Загрузить операционную систему согласно стандартным процедурам загрузки.
5. Открыть демонстрационный пример (проект) в исследуемой среде программирования (AVR Studio, Bascom-AVR, WinAVR, Algorithm Builder). Откомпилировать проект в исследуемой среде и получить готовый бинарный HEX-файл.
6. Подключить разъём «ПРОГРАММАТОР» учебного прибора к COM – порту ПЭВМ проводом типа COM 9m/9f («мама» - «папа»).
7. Прошить тестовую программу (HEX-файл) в кристалл. Т. к. для программирования микроконтроллера (прошивки) используется резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор («ПРОГРАММАТОР ГРОМОВА»), то для прошивки hex-файлов, полученных в средах программирования AVR Studio, Bascom-AVR, WinAVR следует использовать утилиту SinaPROG, поддерживающую данный резистивно-диодный последовательный программатор. При работе в среде Algorithm Builder полученный HEX-файл может быть зашит в микроконтроллер непосредственно из среды.
8. **При работе с программой SinaPROG следует избегать пробелов и русских символов в полном имени HEX-файла (имя с учетом полного пути к файлу) и общей длины пути к HEX-файлу более 255 символов. Идеальным вариантом будет копирование полученного HEX-файла в корень диска перед прошивкой.**
9. **Перед прошивкой следует нажать кнопку с фиксацией «RESET» на учебном приборе.**
10. **Особенно внимательно следует отнестись к опциям «ЗАПИСЬ Fuse bit» и «ЗАПИСЬ блокирующих бит» в среде Algorithm Builder и в утилите SinaPROG. ФЛАЖКИ ЭТИХ ОПЦИЙ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ СНЯТЫ, И ЭТУ ОПЦИЮ В УЧЕБНЫХ ЦЕЛЯХ ИСПОЛЬЗОВАТЬ НЕ РЕКОМЕНДУЕТСЯ, Т. К. НЕ ПРАВИЛЬНО**

ПРОШИТЫЕ FUSE БИТЫ МОГУТ ПРИВЕСТИ К ДАЛЬНЕЙШЕЙ НЕВОЗМОЖНОСТИ РАБОТЫ С ДАННЫМ МИКРОКОНТРОЛЛЕРОМ!

11. **Отжать кнопку RESET и проверить работу тестовой программы.**
12. Исходя из данного учебного примера, составить свои программы в различных средах программирования. Например, программа может прибавлять к данным порта А (PA0-PA7) любое другое произвольное число. Также рекомендуется поработать с действиями «умножение» и «вычитание».
13. Для составления программ и подробного изучения работы микроконтроллеров AVR следует обращаться к учебной литературе, полному руководству к микроконтроллерам семейства AVR, а также к документации по средам программирования, используемым в работе.
14. По окончании работы следует закрыть программу и все открытые подпрограммы.
15. Выключить компьютер, нажав на кнопку, находящуюся в крайнем нижнем левом углу экрана. Из доступных действий выбрать «ВЫХОД»--> «ВЫКЛЮЧИТЬ КОМПЬЮТЕР».
16. Отключить установку от сети, поставив переключатели «СЕТЬ» на панели установки в положение «ВЫКЛ» и вынуть сетевые вилки из розеток.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА.

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.: Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярные цифровые микросхемы. – М.: Радио и связь, 1990. –350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь, 1990 (1982).
11. Большие интегральные схемы ЗУ./ Под ред. А.Ю. Гордонова, Ю.Н. Дьякова. Справочное пособие. – М: Радио и связь, 1990. – 286 с.
12. Федорков Б.Г., Телец В.А. Микросхемы ЦАП и АЦП: Функционирование, параметры, применение. – М.: Энергоатомиздат, 1990. – 320 с.
13. Гилмор Ч. Введение в микропроцессорную технику. Пер. с англ. – М.: Мир, 1984.

ДЛЯ СВОБОДНОГО РАСПРОСТРАНЕНИЯ
НПО УЧЕБНОЙ ТЕХНИКИ «ТУЛАНАУЧПРИБОР»