КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Гумеров Р.И.

Практикум по микропроцессорам

Часть первая: микроконтроллеры AVR

Руководство

Казань – 2009

УДК 681.3.068

Печатается по решению редакционно-издательского совета

физического факультета

Казанского государственного университета

Рецензент

Кандидат физ.-мат. наук, доцент Никитин С.И.

Гумеров Р.И.

Практикум по микропроцессорам. Часть первая: микроконтроллеры AVR. Руководство. – Казань: КГУ, 2009, -37 стр.

Аннотация:

В руководстве компактно изложены вопросы практического применения микроконтроллеров AVR Atmel, архитектурные особенности и средства разработки. На конкретных примерах рассмотрены возможности системы команд и периферии микроконтроллеров, среды разработки и отладки приложений. Практические задания даются в порядке нарастающей сложности и связаны с разработками законченных цифровых устройств.

Предназначено для студентов обучающихся по направлению «010800 – Радиофизика и электроника»

СОДЕРЖАНИЕ

Практикум по микропроцессорам.

Часть первая: микроконтроллеры AVR.

Введение

Руководство предназначено для изучения архитектурных особенностей современных микропроцессоров, средств ввода/вывода, интерфейсов на практике в процессе проектирования простых устройств на основе микроконтроллеров. Микропроцессоры в своем развитии разделились на несколько разновидностей в основном относящимся к двум главным группам: собственно микропроцессорам и микроконтроллерам. Первые предназначены для использования в составе вычислительных систем, например, в персональных компьютерах (процессоры для PC). К этой же группе относят высокопроизводительные процессоры для серверов и некоторые другие.

Микроконтроллеры отличаются от микропроцессоров тем, что они предназначены для управления различными системами. При относительно более слабом вычислительном ядре микроконтроллеры включают в себя много дополнительных блоков, обеспечивающих взаимодействие с внешними устройствами: память, порты ввода/вывода, таймеры, контроллеры прерываний и прямого доступа к памяти, устройства аналогового ввода/вывода и многое другое. Для построения полностью функционирующего устройства достаточно единственной микросхемы микроконтроллера.

Первая часть практикума базируется на микроконтроллерах AVR фирмы Atmel. Эти устройства представляют собой RISC микропроцессоры с гарвардской архитектурой и разнообразными ресурсами для ввода/вывода (в том числе и аналоговыми). Основой лабораторной установки является

отладочный набор AS-megaM, который в сочетании с прилагаемым пакетом разработчика «AVR-Studio» и программатором позволяет создавать рабочие приложения. Язык ассемблера для данного типа микропроцессоров весьма нагляден и прост, и в сочетании с «AVR-Studio» дает возможность пользователю детально разобраться, как работают ядро процессора и периферийные устройства.

1. Общие сведения

Микроконтроллеры AVR представляют собой однокристальные 8-разрядные RISC-контроллеры (Reduced Instruction Set Computer – компьютер с сокращённым набором команд), обладающие программируемой FLASH – памятью и EPROM памятью (Erasable Programmable Read Only Memory – стираемое программируемое постоянное запоминающее устройство), выпускаемые фирмой ATMEL (Advanced Technology Memory and Logic). Название AVR они получили в честь двух студентов, участвовавших в разработке микропроцессоров и выдвинувших идею восьмиразрядного RISC ядра: Альфа Богена и Вергарда Воллена. Alf + Vergard +Risc.

Представители семейства AVR обладают ограниченным набором из 118 высокоэффективных команд. Благодаря особой архитектуре (рис.1), в вычислениях, кроме накапливающего сумматора, задействованы 32 равноправных рабочих регистра, напрямую связанных с арифметико-логическим устройством, что исключает необходимость после завершения вычислительных операций обращения к вспомогательным регистрам или промежуточным запоминающим устройствам. Микроконтроллеры AVR построены в соответствии с Гарвардской архитектурой, что подразумевает разделение памяти на две части: для программ и данных; обращение к каждой части по своей шине. Микроконтроллеры используют одноступенчатую

конвейерную обработку. Это означает, что во время выполнения текущей команды выполняется загрузка следующей команды из памяти программ. Благодаря этому достигается возможность выполнения команды в течение одного тактового цикла. Поэтому при использовании в качестве системного тактирования, тактовой частоты непосредственно сгенерированной кварцевым осциллятором (у большинства процессоров и контроллеров для получения собственно системного такта она делится на заданный коэффициент), достигается быстродействие равное частоте используемого кварца.

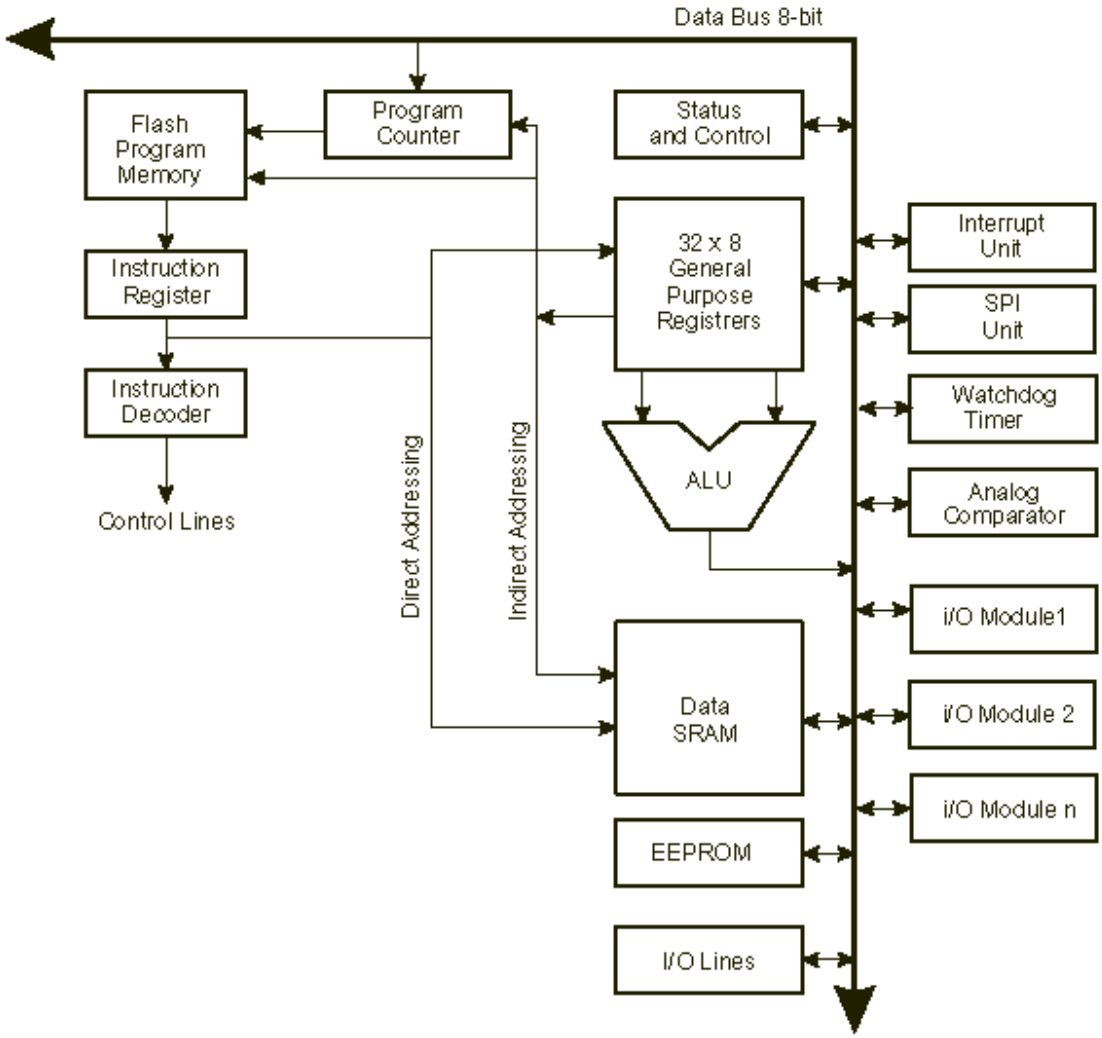


Рис.1. Архитектура ядра контроллера AVR

Так при работе с частотой кварца 7МГц получаем около 7 миллионов выполненных команд в течение одной секунды (0,14 микросекунды на команду). За немногими исключениями, в которых нарушается конвейер (первая команда в программе, команды условного и безусловного перехода, входы и выходы в прерывания и процедуры …), микроконтроллеры семейства AVR действительно обрабатывают все команды в течение единственного системного такта.

В рамках единой базовой архитектуры (рис.1) микроконтроллеры AVR подразделяются на три семейства:

- *Tiny AVR*
- *Classic AVR*
- *Mega AVR*

Они различаются степенью развития периферии и объёмом памяти программ и данных, причём контроллеры семейства Mega имеют наилучшие показатели. Микроконтроллеры каждого семейства поддерживают несколько режимов энергопотребления, имеют блок прерываний, сторожевой таймер и допускают программирование непосредственно в готовом устройстве. Мы в своих заданиях будем использовать микропроцессорные контроллеры семейства Mega, как функционально наиболее полные.

1.1. Архитектура семейства Mega

Как и все микроконтроллеры AVR представители семейства Mega являются 8 - разрядными микроконтроллерами, предназначенными для встраиваемых приложений. Они имеют электрически стираемую память программ (FLASH) и данных (EEPROM), а также разнообразные периферийные устройства. Они изготавливаются по мало потребляющей КМОП – технологии,

которая в сочетании с усовершенствованной RISC – архитектурой позволяет достичь наилучшего соотношения быстродействие/энергопотребление. Микроконтроллеры данного семейства являются наиболее развитыми представителями AVR (рис. 2).

Отличительные особенности:

● FLASH – память объёмом 8…128 Кбайт (128Кбайт – у ATmega128). Число циклов стирания/записи не менее 10000.

● Оперативная память (статическое ОЗУ) объёмом 1…4Кбайт (4Кбайт – у ATmega128).

● Память данных на основе EEPROM объёмом 512байт…4Кбайт (4Кбайт – у ATmega128).Число циклов стирания/записи не менее 100000.

● Возможность защиты от чтения и модификации памяти программ и данных.

● Возможность программирования непосредственно в системе через последовательные интерфейсы SPI и JTAG.

● Возможность самопрограммирования.

● Возможность внутрисхемной отладки в соответствии со стандартом IEEE 1149.1 (JTAG).

● Различные способы синхронизации: встроенный RC – генератор с внутренней или внешней времязадающей RC – цепочкой или с внешним резонатором (пьезокерамическим или кварцевым), внешний сигнал синхронизации.

● Наличие нескольких режимов пониженного энергопотребления.

● Наличие детектора снижения напряжения питания.

● Возможность программного снижения частоты тактового генератора.

Периферийные устройства микроконтроллера ATmega128:

● два 8 – разрядных таймера/счётчика

● два 16 – разрядных таймера/счётчика

● сторожевой таймер

● аналоговый компаратор

● многоканальный 10 – разрядный АЦП

●полнодуплексный универсальный синхронный/асинхронный приёмопередатчик

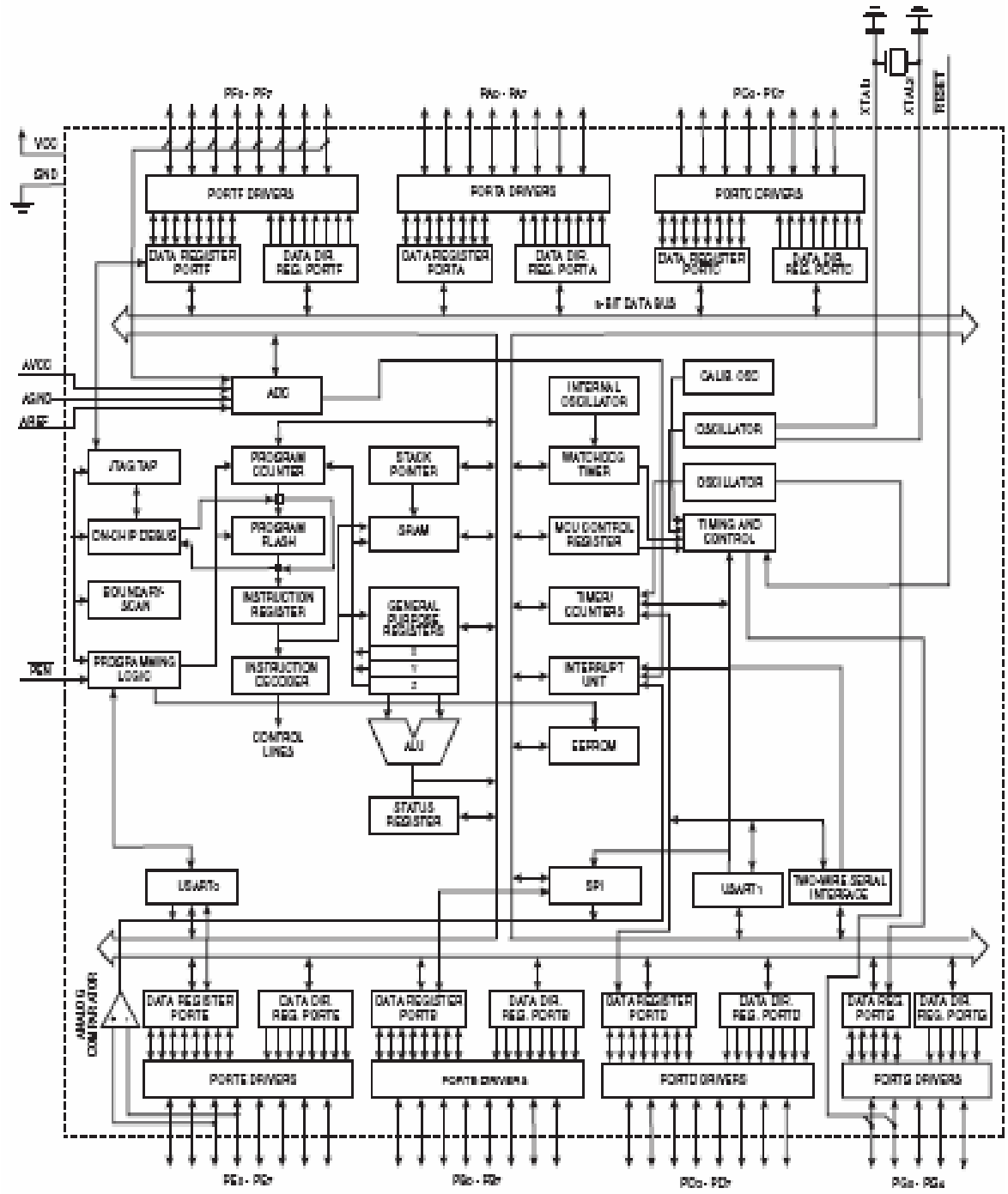


Рис. 2. Структурная схема микроконтроллера ATmega128

● последовательный синхронный интерфейс

● последовательный двухпроводной интерфейс

1.1.1. Центральный процессор и память

Характеристики процессора:

● полностью статическая архитектура (минимальная тактовая частота равна нулю).

● АЛУ подключено непосредственно к регистрам общего назначения.

● большинство команд выполняется за один машинный цикл.

● многоуровневая система прерываний.

● поддержка очереди прерываний.

● 27 источников прерываний, из них 8 внешних.

● наличие программного стека.

● наличие аппаратного умножителя.

В микроконтроллерах AVR семейства Mega реализована Гарвардская архитектура, в соответствии с которой разделены не только адресные пространства памяти программ и памяти данных, но и шины доступа к ним (рис. 2). То есть способы адресации и доступа к этим областям памяти также различны. Такая структура позволяет центральному процессору работать одновременно и с памятью программ и с памятью данных, что существенно увеличивает производительность.

Память программ предназначена для хранения команд, управляющих функционированием микроконтроллера и таблиц констант, не меняющихся во время работы. Как уже было сказано, она представляет собой электрически стираемое ППЗУ (FLASH – ПЗУ). В связи с тем что длина всех команд кратна одному 16-битному слову, память программ имеет 16-разрядную организацию. Соответственно, объём памяти микроконтроллера ATmega128 составляет 64×1024 16-разрядных слов. Карта памяти приведена на рис.3.

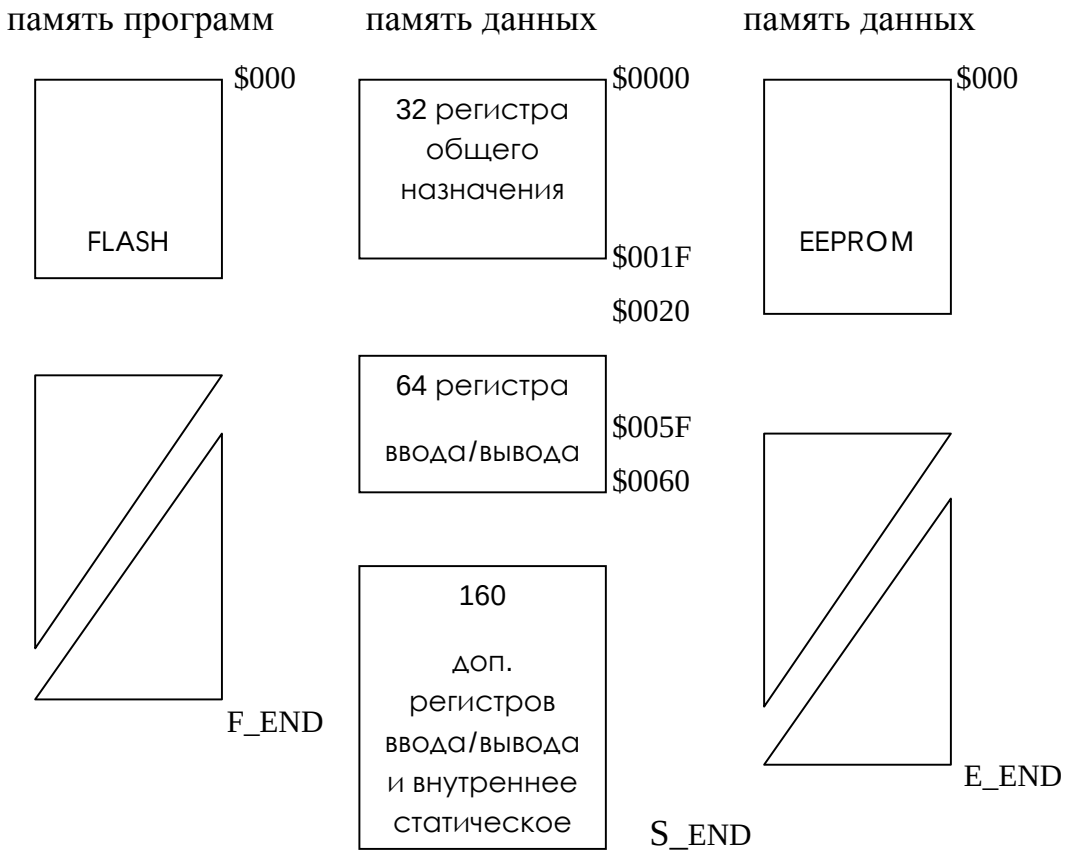


Рис. 3. Обобщённая карта памяти ATMega 128.

Для адресации памяти программ используется счётчик команд (PC – Program Counter). В ATmega128 размер счётчика команд составляет 16 бит. При нормальном выполнении программ содержимое счётчика команд автоматически увеличивается на 1 или на 2 (в зависимости от размера выполняемой команды) в каждом машинном цикле. Этот порядок нарушается при выполнении команд перехода, вызова и возврата из подпрограмм, а также при возникновении прерываний. По адресу $0000 памяти программ находится вектор сброса, то есть после сброса микроконтроллера выполнение программы начинается с этого адреса, и по нему должна размещаться команда перехода к инициализационной

части программы. Начиная с адреса $0001 памяти программ, располагается таблица векторов прерываний. При возникновении прерывания после сохранения в стеке текущего значения счётчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. По этим адресам должны располагаться команды перехода к подпрограммам обработки соответствующих прерываний.

Память данных микроконтроллеров семейства Mega линейно организована и разделена на три части: регистровая память, оперативная память (статическое ОЗУ) и энергонезависимое ЭСППЗУ – электрически стираемое программируемое постоянное запоминающее устройство(EEPROM). Регистровая память включает 32 регистра общего назначения (РОН), объединённых в файл, служебные регистры ввода/вывода (РВВ) и область дополнительных регистров ввода/вывода (ДРВВ). Под РВВ в памяти контроллера отводится 64 байт, под ДРВВ – 160 байт.

Все регистры общего назначения объединены в регистровый файл быстрого доступа. Как уже было сказано, в микроконтроллерах AVR все 32 РОН непосредственно доступны АЛУ, что позволяет использовать любой РОН практически во всех командах и как операнд-источник и как операнд-приёмник. *Такое решение (в сочетании с конвейерной обработкой) позволяет АЛУ выполнять одну операцию (извлечение операндов из регистрового файла, выполнение команды и запись результата обратно в регистровый файл) за один машинный цикл.*

В обеих областях регистров ввода/вывода располагаются различные служебные регистры (регистр управления микроконтроллером, регистр состояния и т.п.), а также регистры управления периферийными устройствами, входящими в состав микроконтроллера. Эти регистры располагаются в так

называемом пространстве ввода/вывода, то есть существуют команды быстрого доступа к ним, использующие отдельную адресацию. Введение дополнительных РВВ связано с тем, что для поддержки всех периферийных устройств ATmega128 обычных 64-х РВВ не хватает.

Для хранения переменных программ помимо РОН может использоваться статическое ОЗУ объёмом 4 Кб (ATmega128). В адресном пространстве ОЗУ также расположены все регистры микроконтроллера, под них отведены младшие 256 адресов. Что позволяет обращаться к РОН и РВВ как к ячейкам ОЗУ, хотя физически они ими не являются. Такое решение увеличивает эффективность подсистемы адресации.

Для долговременного хранения различной информации в микроконтроллере используется EEPROM – память, размер которой у ATmega128 равен 4Кб. Эта память расположена в отдельном адресном пространстве, а доступ к ней осуществляется с помощью определённых РВВ.

1.1.2. Порты ввода/вывода

Характеристики подсистемы ввода/вывода:

- программное конфигурирование и выбор портов ввода/вывода;
- выводы могут быть запрограммированы как входные или как выходные независимо друг от друга;
- входные буферы с триггером Шмидта на всех выводах;
- возможность подключения ко всем входам внутренних подтягивающих резисторов(35…120 кОм).

Каждый порт микроконтроллера состоит из определённого числа выводов, через которые он может осуществлять приём и передачу цифровых сигналов. Задание направления передачи данных через любой контакт ввода/вывода может быть произведено программно в любой момент времени.

Выходные буферы всех портов, имея симметричные нагрузочные характеристики, обеспечивают высокую нагрузочную способность при любом уровне сигнала.

Входные буферы всех выводов построены по схеме триггера Шмидта. Для всех входов имеется возможность подключения внутреннего подтягивающего резистора между входом и шиной питания.

Отличительной особенностью портов микроконтроллеров AVR при использовании их в качестве цифровых портов ввода/вывода общего назначения является возможность выполнять операции над любым выводом (с помощью команд CBI и SBI), не влияя на другие выводы порта. Это относится к изменению режима работы контакта ввода/вывода, к изменению состояния выходного буфера (для выходов) и к изменению состояния внутреннего подтягивающего резистора (для входов).

Микроконтроллер ATmega128 имеет шесть 8-разрядных портов ввода/вывода и один 5-разрядный, то есть всего 53 контакта ввода/вывода.

Управление, конфигурирование и обращение к портам ввода/вывода осуществляется с помощью соответствующих регистров ввода/вывода (РВВ).

В ATmega128 предусмотрены 8 внешних прерываний, связанных с контактами ввода/вывода и несколько условий их генерации: по низкому уровню на выводе, по спадающему фронту сигнала, по нарастающему фронту сигнала на выводе. Управление обработкой прерываний от контактов ввода/вывода осуществляется при помощи соответствующих РВВ.

Необходимо отметить, что линии портов ввода/вывода несут в себе несколько функций, зависящих от режима работы контроллера и периферийных устройств, подключенных к этим линиям.

1.1.3. Таймеры/счётчики.

В микроконтроллере ATmega128 присутствуют 4 таймера/счётчика. И обозначаются они как T/C0, T/C1, T/C2, T/C3. Кроме того, в составе микроконтроллера имеется еще и, так называемый, сторожевой таймер, позволяющий избежать зависания программы.

8 –ми разрядные таймеры/счётчики T/C0 и T/C2 идентичны, равно как идентичны и 16-ти разрядные T/C1 и T/C2. Схематично таймер/счетчик представляет собой:

1. счетный регистр, который инкрементируется (или декрементируется), в зависимости от режима работы таймера в каждом такте счётчика;
2. несколько вспомогательных регистров, необходимых для задания режимов работы счётчика (например, регистр блока сравнения или блока захвата);
3. линии входа и выхода;
4. механизм генерации прерываний на события, связанные с изменением состояния счётного регистра.

В таймерах/счетчиках T/C0 и T/C2 счётные регистры имеют по 8 разрядов и соответственно максимальное разрешение этих счётчиков – 256 тактов таймера. Счётные регистры T/C1 и T/C3 16 – разрядные и максимальный временной интервал, который они непосредственно могут измерить 65536 тактов таймера. Для получения тактового сигнала таймеров/счётчиков используются предделители, которые делят системный тактовый сигнал на программно-задаваемый коэффициент. В ATmega128 имеется 2 предделителя: один для T/C0, другой используется совместно таймерами T/C1,T/C2,T/C3. В качестве тактового сигнала таймеров может быть использован либо непосредственно системный тактовый сигнал микропроцессора clc_0, либо внешний сигнал с частотой не более, чем $clc_0/2$.

В ATmega128 предусмотрены несколько режимов работы таймеров/счётчиков и связанных с ними прерываний:

1. Normal(обычный)

2. CTC(сброс при совпадении)
3. Fast PWM(быстродействующий ШИМ)
4. Phase Correct PWM (ШИМ с точной фазой).

В режиме Normal по каждому импульсу тактового сигнала осуществляется инкремент счётного регистра. При переходе через максимально возможное значение ($FF- для T/C0 и T/C2, $FFFF- для T/C1 и T/C2) возникает переполнение, и счёт продолжается с $0000. В том же такте сигнала, в котором обнуляется счётный регистр, генерируется запрос на прерывание по переполнению. В этом режиме также возможна генерация прерываний по совпадению: когда значение счётного регистра совпадает со значением, находящимся в регистре сравнения (в случае 16-разрядных таймеров существует несколько прерываний по совпадению, так как в них реализованы несколько блоков сравнения, точнее 3). Блоки сравнения таймеров могут также использоваться для изменения состояний линий выходов соответствующих таймеров/счётчиков.

В режиме CTC, также как и в предыдущем случае, инкремент счётного регистра осуществляется по каждому импульсу тактового сигнала, однако максимальное значение счётного регистра и, следовательно, разрешающая способность счётчика определяется регистром сравнения (в случае 16-разрядных таймеров регистром сравнения первого (A) блока). После достижения значения, записанного в регистре сравнения, счёт продолжается со значения $0000. В этом же такте сигнала, происходит генерация запросов на прерывания по переполнению и по совпадению соответствующего таймера/счётчика. Одновременно с генерацией прерываний может изменяться состояние выхода соответствующего таймера.

Режимы Fast PWM и Phase Correct PWM предназначены для генерации сигналов с широтно-импульсной модуляцией.

Доступ к таймерам/счётчикам и управление связанными с ними прерываниями и предделителями осуществляется с помощью соответствующих РВВ.

1.1.4. Универсальный синхронный/асинхронный приёмопередатчик

Микроконтроллер ATmega128 имеет два модуля последовательной связи, называемые универсальными синхронно/асинхронными приёмопередатчиками (USART). Оба модуля USART обеспечивают полнодуплексный обмен по последовательному каналу, при этом скорость передачи данных может варьироваться в довольно широких пределах. Длина посылки может составлять от 5 до 9 разрядов с контролем четности, или без него. Кроме того, модули USART могут обнаруживать различные внештатные ситуации, такие как – переполнение буфера приёмника, ошибка кадрирования, неверный старт бит. Для уменьшения вероятности сбоев в модулях реализована функция фильтрации помех.

Для взаимодействия с программой в модулях предусмотрены 3 прерывания, запрос на генерацию которых генерируется при наступлении событий: передача завершена, регистр данных передатчика пуст, приём завершён.

Выводы микроконтроллера, используемые модулями USART, являются линиями портов ввода/вывода общего назначения, функционирование которых переопределяется при включении приёмника или передатчика.

Каждый модуль USART состоит из трёх основных частей: блока тактирования, блока передатчика и блока приёмника.

Блок тактирования включает в себя схему синхронизации, которая используется при работе в синхронном режиме, и контроллер скорости передачи.

Блок передатчика включает одноуровневый буфер, сдвиговый регистр, схему формирования бита чётности и схему управления.

Блок приёмника, в свою очередь, включает схемы восстановления тактового сигнала и данных, схему контроля чётности, двухуровневый FIFO-буфер, сдвиговый регистр, а также схему управления.

Доступ и управление модулями USART, как обычно, осуществляется с помощью соответствующих РВВ.

1.2. Программирование AVR

Микроконтроллер ATmega128 поддерживает 3 режима программирования:

- последовательное программирование при низком напряжении (по интерфейсу SPI)
- параллельное программирование при высоком напряжении
- программирование по интерфейсу JTAG.

А также имеет возможность самопрограммирования. Под этим термином понимается изменение содержимого памяти программ, управляемое самим микроконтроллером.

В процессе программирования могут выполняться следующие операции:

- стирание кристалла
- чтение/запись памяти программ
- чтение/запись памяти данных
- чтение/запись ячеек защиты
- чтение/запись конфигурационных ячеек
- чтение ячеек идентификатора
- чтение калибровочного байта.

Содержимое памяти программ и данных может быть защищено от чтения или/и записи посредством программирования соответствующих ячеек защиты. После программирования этих ячеек память контроллера блокируется для

соответствующих операций. Ячейки защиты могут быть перезаписаны только после выполнения команды стирания кристалла, что также уничтожит содержимое памяти программ и данных микроконтроллера.

Конфигурационные ячейки определяют некоторые параметры конфигурации микроконтроллера. Эти ячейки расположены в отдельном адресном пространстве, доступном только при программировании. Команда стирание кристалла на состояние этих ячеек не влияет.

Все микроконтроллеры AVR имеют три 8-разрядные ячейки, позволяющие идентифицировать устройство (ячейки идентификатора). Они доступны только в режиме программирования и только для чтения.

В калибровочную ячейку при изготовлении микроконтроллера заносится калибровочная константа RC-генератора. При работе от внутреннего RC-генератора необходимо воспользоваться содержимым этой ячейки для подстройки внутреннего генератора на номинальную частоту.

В данной работе для программирования микроконтроллера использовался программатор AS-2, работающий по интерфейсу SPI, а для написания и отладки программы на ассемблере использовалась интегрированная среда разработки «AVR-Studio», разработанная компанией ATMEL для поддержки выпускаемых устройств.

Программа «AVR-Studio» (рис. 4), предназначенная для операционных систем Windows, позволяет создавать, редактировать и отлаживать программы для микроконтроллеров AVR на языках ассемблера и Си. Она свободно доступна с сайта компании ATMEL. Благодаря наличию для каждой модели микроконтроллеров своих заголовочных файлов, где определены характерные константы, такие как максимальный адрес памяти программ и данных, адреса регистров ввода/вывода и т.д., процесс программирования существенно упрощается. Программный симулятор всех моделей позволяет моделировать поведение микроконтроллера при выполнении отлаживаемой программы, что

делает возможным поиск ошибок в ней ещё до записи непосредственно в микроконтроллер.

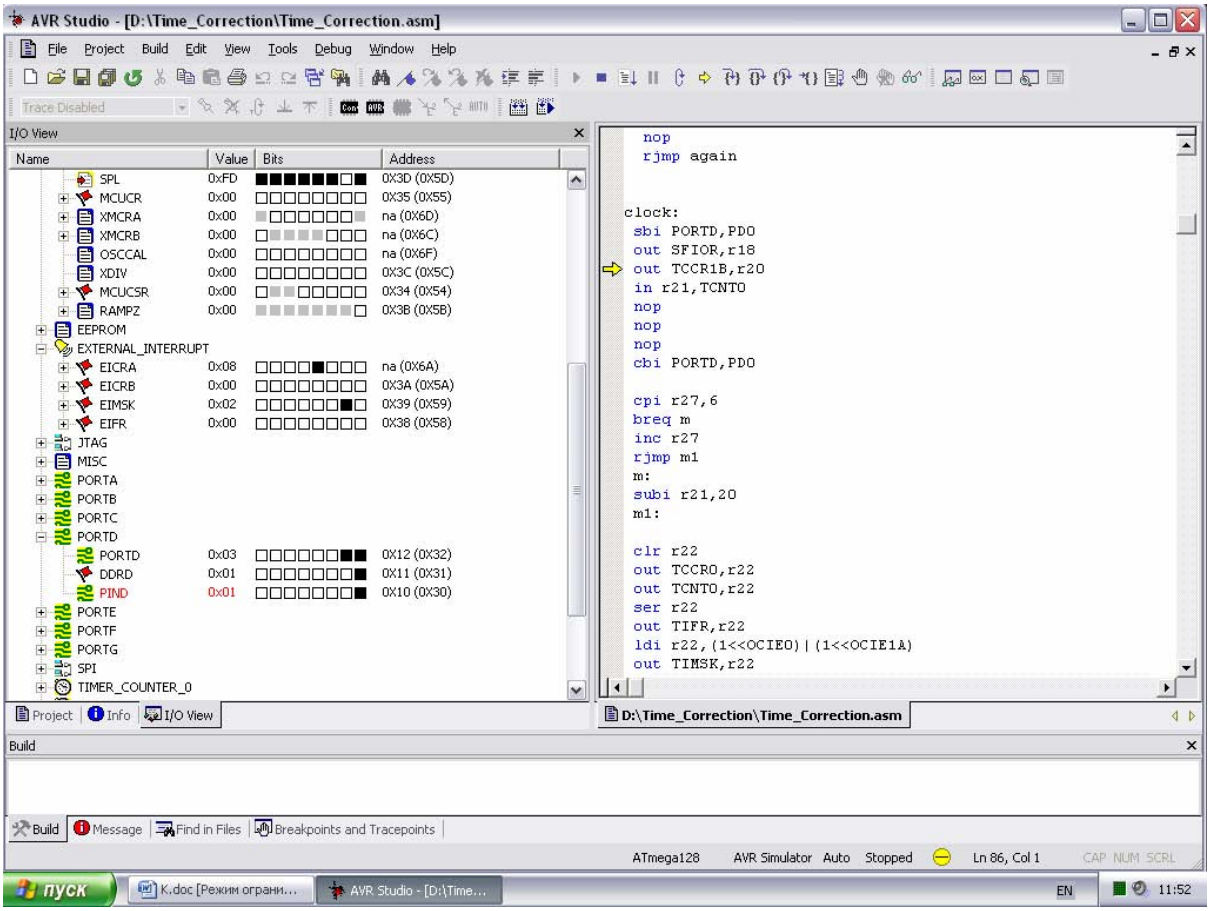


Рис. 4. Консоль программы «AVR-Studio».

Результатом работы «AVR-Studio» является HEX файл с кодами команд для микроконтроллера. Для записи этого файла в память микроконтроллера можно использовать программатор AS-2 фирмы Аргуссофт вместе с поставляемой для него программы управления.

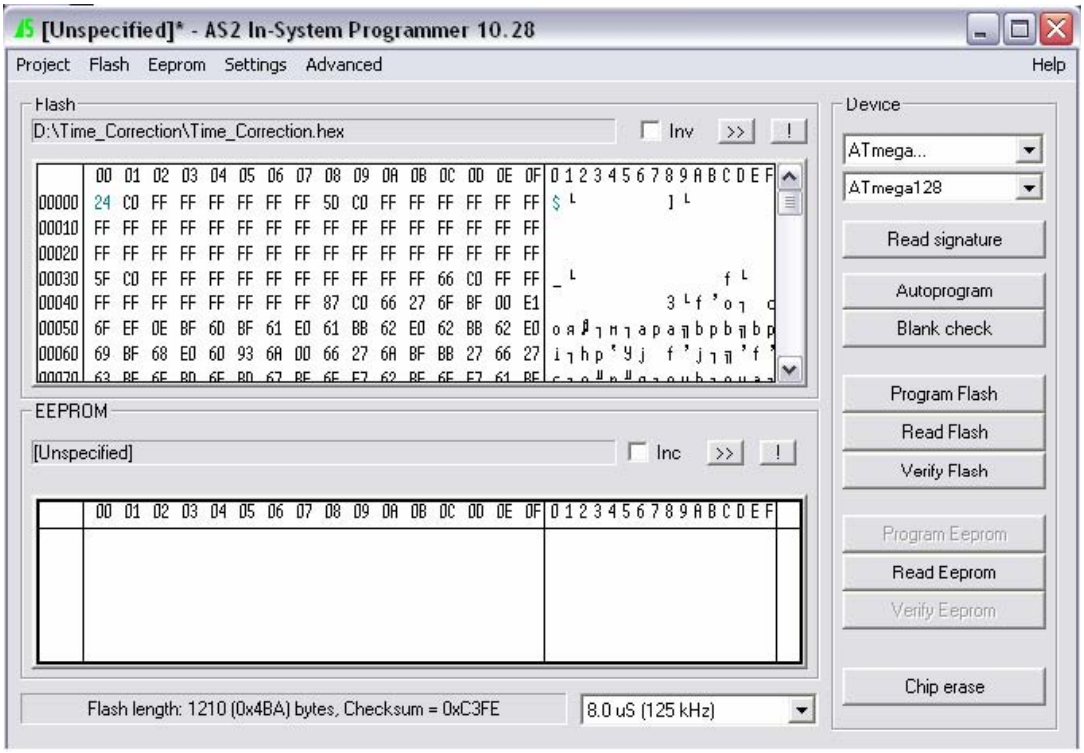


Рис. 5. Рабочее окно программатора AS-2.

2. Пример построения приложения

Работу пакета «AVR-Studio» удобно рассматривать на примере решения какой-либо конкретной задачи. В качестве иллюстрации рассмотрим создание проекта простой программы, реализующей поочередное включение/выключение двух светодиодов. Для отладочной платы AS-megaM (см. Приложение 1) подключим два светодиода к разрядам 6 и 7 порта D (к выводам 31 и 32 микросхемы ATmega128). AVR-контроллеры имеют мощные выходные каскады, типовое значение тока каждого вывода составляет 20 мА, максимальный ток вывода – 40 мА, причем это относится как к втекающему, так и к вытекающему току. В нашем примере светодиоды подключены анодами к выводам контроллера, а катоды через гасящие резисторы соединены с землей. Это означает, что светодиод зажигается подачей логической «1» на соответствующий вывод порта. Принципиальная схема приведена на рисунке.

На схеме также показаны две кнопки, которые будут использованы в программе.

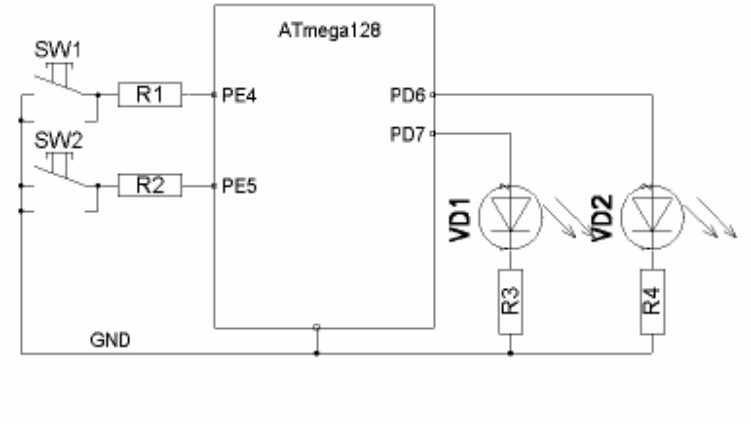


Рис.6. Схема подключения светодиодов

Здесь уместно сделать небольшое отступление о выборе типа микросхемы для простейшего примера. Действительно, с первого взгляда может показаться странным, зачем нужен такой мощный кристалл в 64-выводном корпусе там, где хватит и 8-выводной микросхемы ATtiny12? Однако в таком подходе есть логика. Известно, что в основе практически любого AVR-контроллера лежит одинаковое ядро. В основном контроллеры различаются объемом памяти, количеством портов ввода/вывода и набором периферийных модулей. Особенности каждого конкретного контроллера – привязка логических имен регистров ввода/вывода к физическим адресам, адреса векторов прерываний, определения битов портов и т.д. описаны в “include” файлах (расширением .inc), которые входят в состав пакета «AVR-Studio». Следовательно, используя конкретный тип кристалла, можно отлаживать программу как собственно для него, так и для любого младшего кристалла. И если использовать в качестве отладочного старший кристалл, например, ATmega128, то можно отлаживать программу практически для любого AVR-контроллера. Надо просто не использовать аппаратные ресурсы, которые отсутствуют у целевого микроконтроллера. Таким образом, можно отлаживать на ATmega128

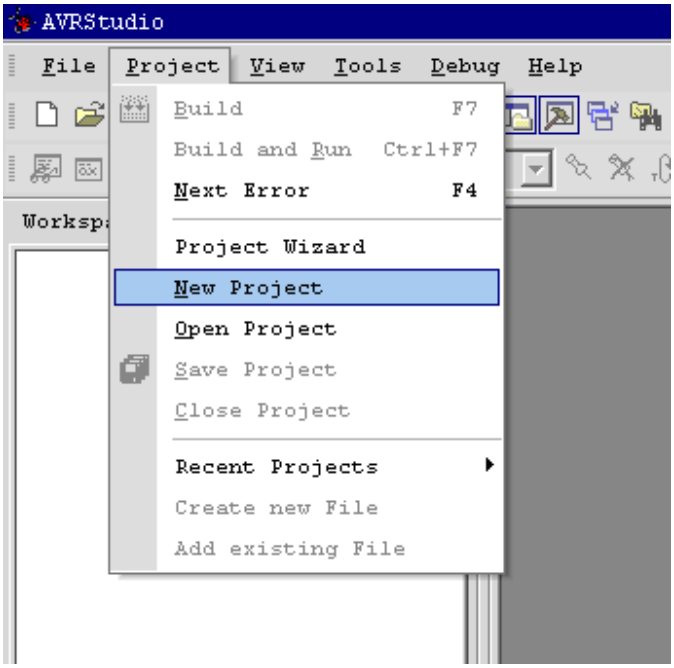
программу, которая будет выполняться, например, на ATtiny13. При этом исходный код останется практически тем же, изменится лишь имя подключаемого файла с 128def.inc на tn13def.inc. У такого подхода также есть свои преимущества. Например, «лишние» порты ввода/вывода можно использовать для подключения ЖК-индикатора, на который можно выводить отладочную информацию. Или, воспользоваться внутрисхемным эмулятором, который подключается к JTAG-порту микросхемы ATmega128 (контроллер ATtiny13 такой порт не имеет). Таким образом, можно использовать отладочную плату AS-megaM для отладки любых вновь разрабатываемых систем, естественно, базирующихся также на AVR-микроконтроллерах. На плате помимо контроллера ATmega128 имеется внешнее ОЗУ, два порта RS-232, порт для подключения ЖК-индикатора, внутрисхемного программатора и эмулятора AT JTAGICE. На плате также есть место для распайки микросхемы FLASH-ПЗУ серии AT45 в корпусах TSOP32/40/48 и двухканального ЦАП серии AD5302/12/22.

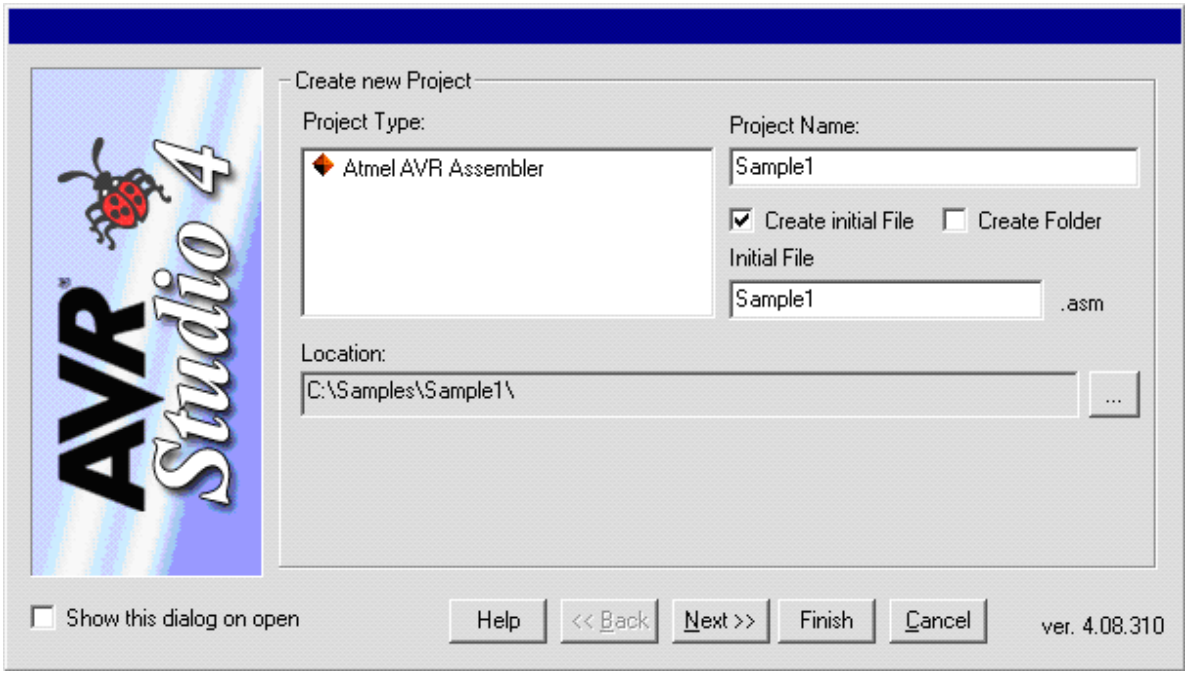
При программировании в среде «AVR-Studio» надо выполнить стандартную последовательность действий:

- с*оздание проекта*
- *загрузка файла*
- *компиляция*
- *симуляция*
- *загрузка hex-кода в микроконтроллер*

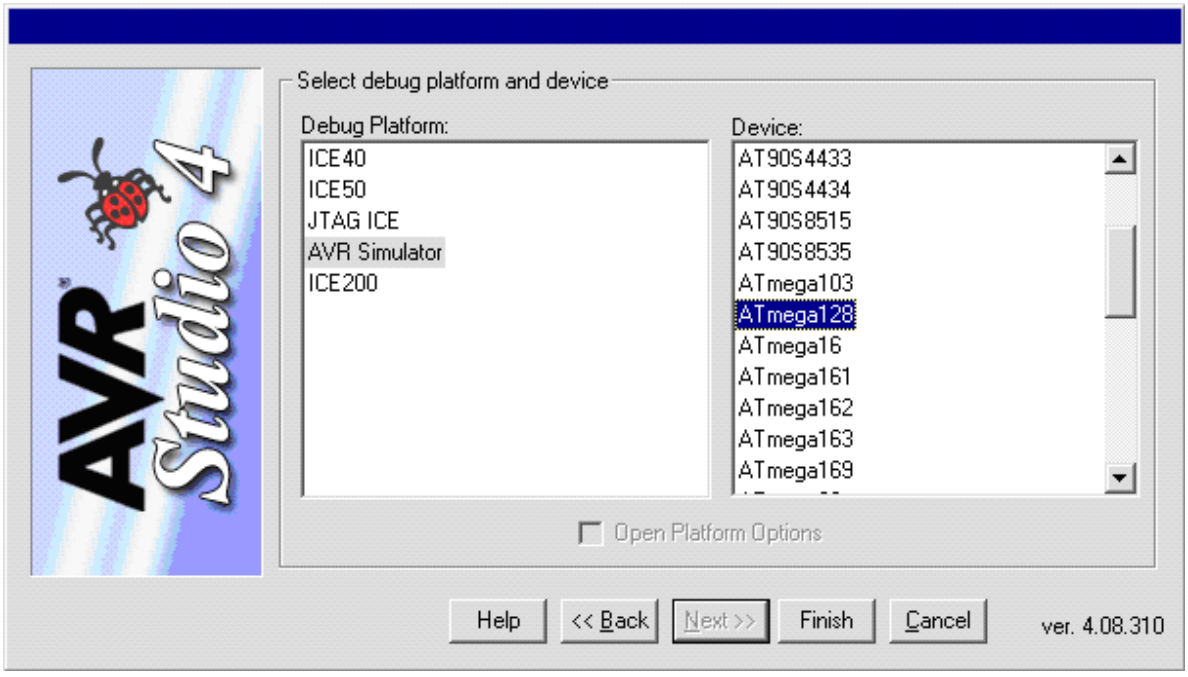
Эти действия выполняются в представленных ниже окнах и в особых пояснениях не нуждаются.

Открыть новый проект





Выбрать требуемый тип микроконтроллера



Создание проекта начинается с выбора строки меню Project\New Project. В открывшемся окне “Create new Project” надо указать имя проекта, (в нашем случае – sample1) и имя файла инициализации. После нажатия кнопки «Next» открывается окно «Select debug platform and device», где выбирается отладочная платформа (симулятор или эмулятор) и тип микроконтроллера.

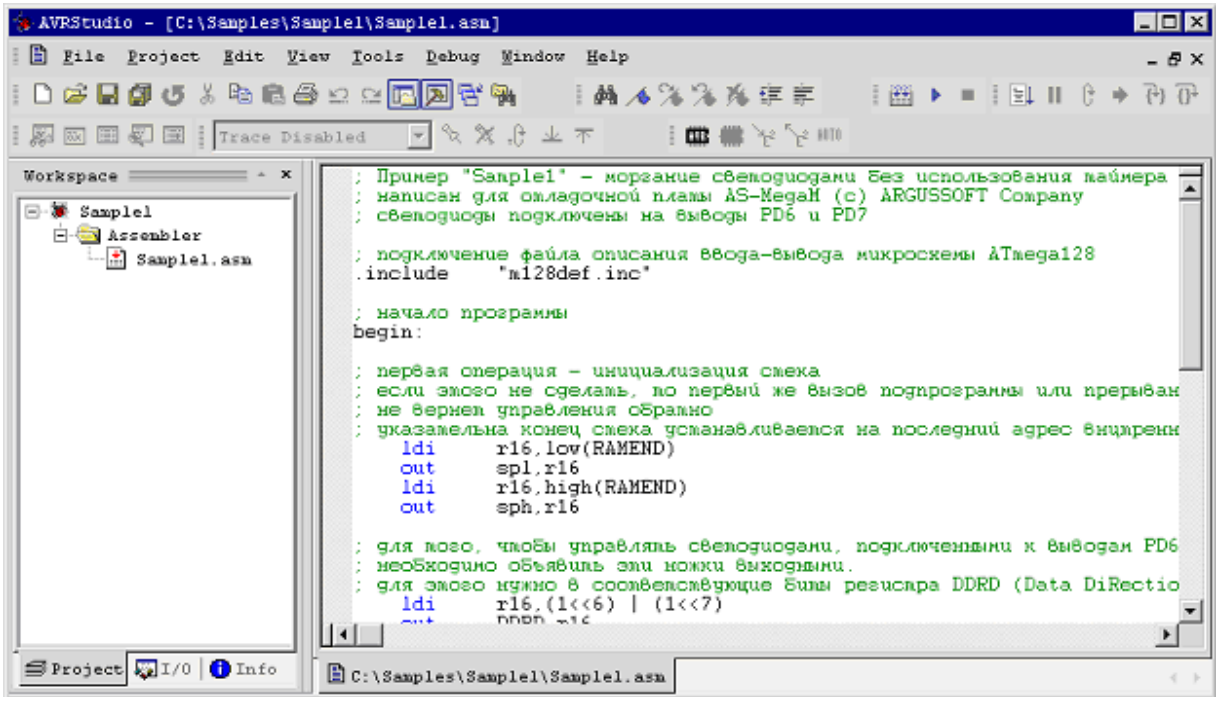
Можно выбрать один из предлагаемых внутрисхемных эмуляторов. Отметим, что у каждого эмулятора свой список поддерживаемых микросхем . Для рассматриваемого примера мы выбираем в качестве отладочной платформы AVR Simulator и микросхему ATmega128. После нажатия кнопки “Finish” мы увидим рабочие окна пакета «AVR-Studio», пока пустые. В правое окно следует поместить исходный текст программы. Это можно сделать двумя способами:

либо набрать весь текст непосредственно в окне редактора, либо загрузить уже существующий файл. Ниже приведен полный текст простой программы с комментариями.

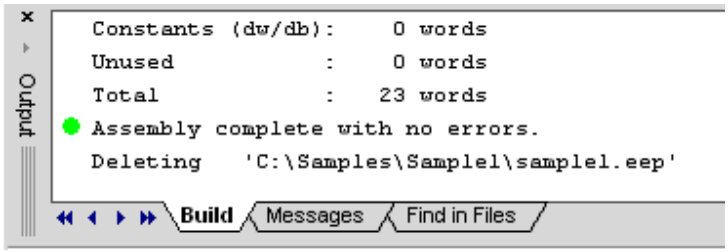
```
; Пример 1: «Управление светодиодами»
; написан для отладочной платы AS-MegaM
; Частота задающего генератора 7,37 МГц
; светодиоды подключены к выводам PD6 и PD7 и через резисторы - на общий провод.
; подключение файла описания ввода-вывода микросхемы ATmega128

.include "m128def.inc"
; начало программы
begin:
; первая операция - инициализация стека
; если этого не сделать, то вызов подпрограммы или прерывания
; не вернет управление обратно
; указатель на конец стека устанавливается на последний адрес внутреннего ОЗУ –
;RAMEND

ldi r16,low(RAMEND)
out spl,r16
ldi r16,high(RAMEND)
out sph,r16

; для того, чтобы управлять светодиодами, подключенными к выводам PD6 и PD7,
; необходимо объявить эти выводы выходными.
; для этого нужно записать "1" в соответствующие биты регистра DDRD
;(DataDiRection)
l
di r16,(1<<6) | (1<<7)
out DDRD,r16

; основной цикл программы
loop:
ldi r16,(1<<6) ; светится один светодиод
out PORTD,r16

rcall delay ; задержка

ldi r16,(1<<7) ; светится второй светодиод
out PORTD,r16

rcall delay ; задержка

rjmp loop ; повторение цикла
```

```
        ; процедура задержки
        ; примерно полсекунды при частоте 7,37 МГц
        ; три пустых вложенных цикла соответственно
        delay:
        ldi r16,30 ; 30
delay1:
        ldi r17,200 ; 200

delay2:
        ldi r18,200 ; и еще 200 итераций
delay3:
        dec r18
        brne delay3
        dec r17
        brne delay2
        dec r16
        brne delay1
        ret ; возврат в главную программу
```

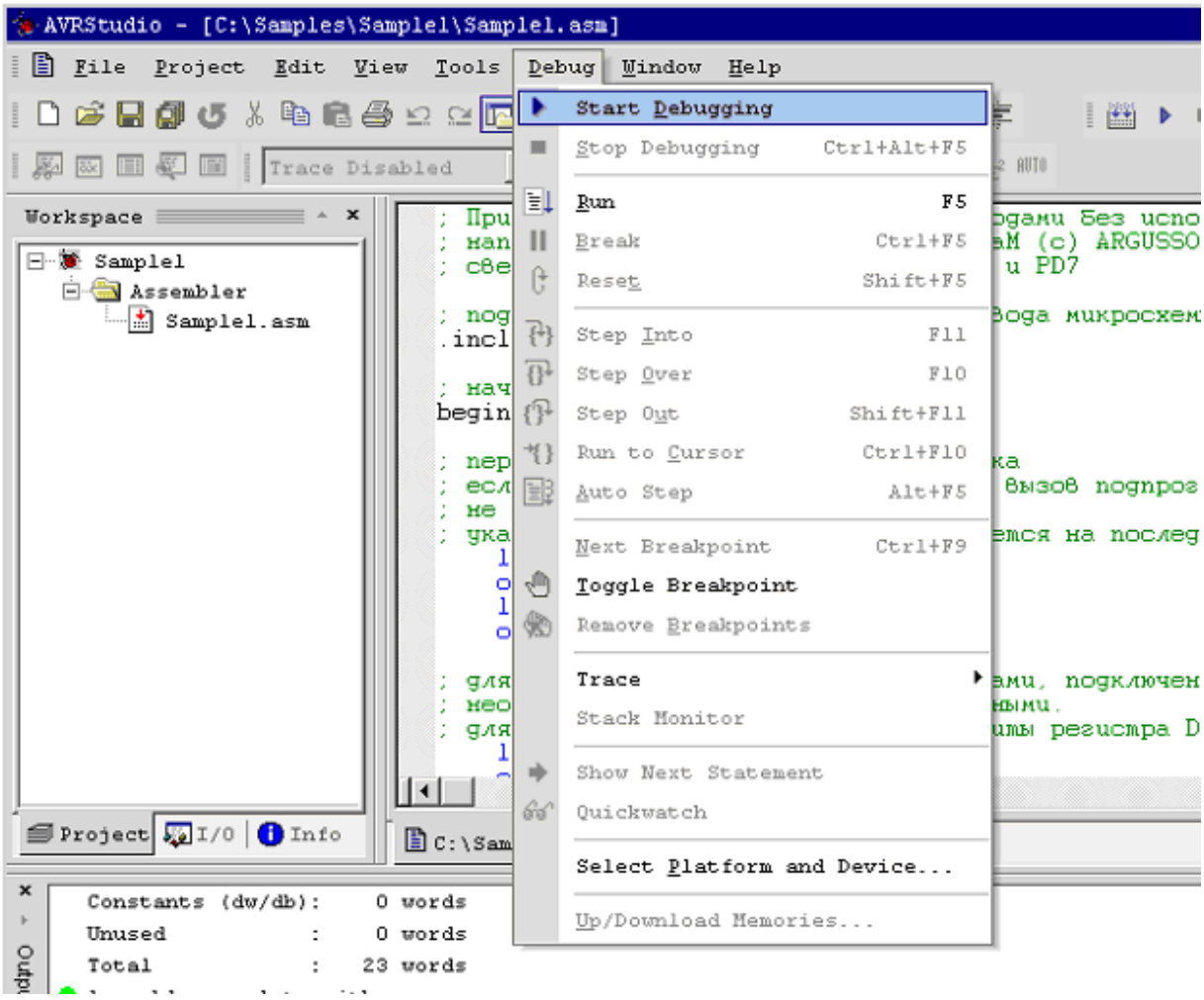
Проект может состоять из нескольких файлов, при этом один файл назначается основным. Все операции удобно производить, используя контекстную кнопку мыши. После подключения исходного файла окна имеют следующий вид.



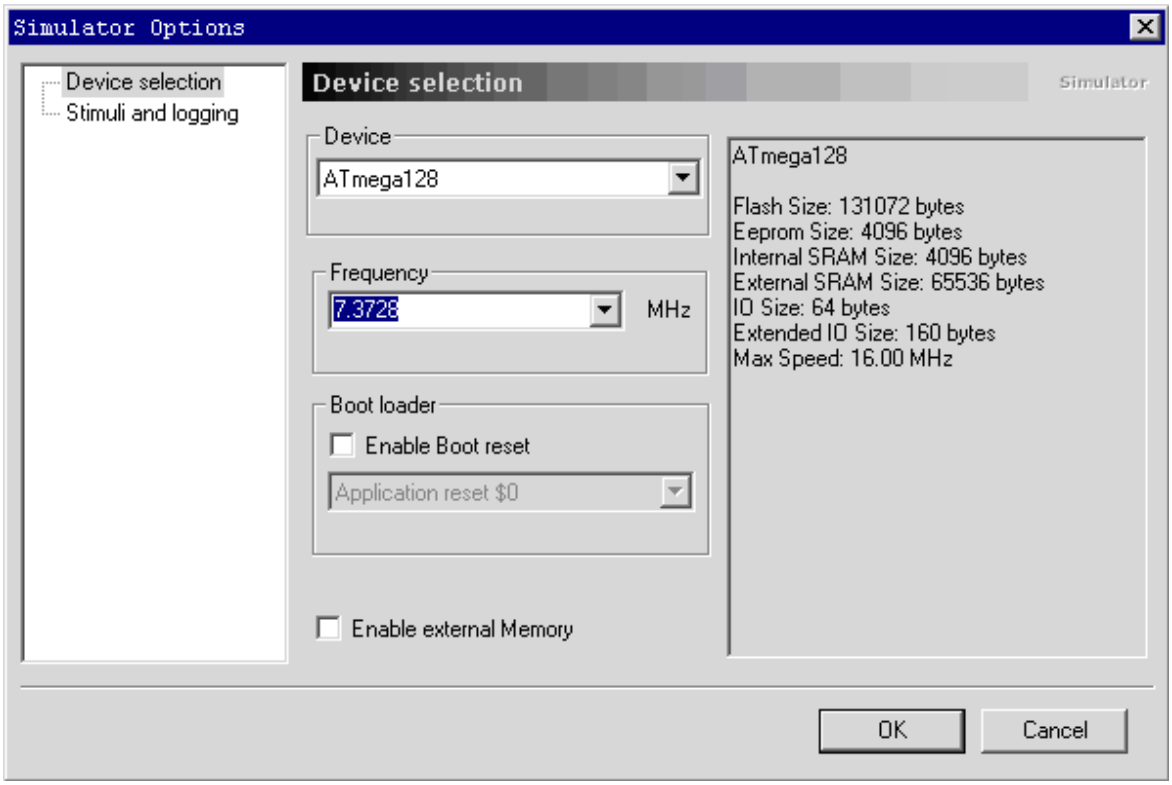
Компиляция проекта выполняется командой \Project\Build или нажатием клавиши F7. Процесс компиляции отображается в окне «Output». Это окно можно «вытащить» командой \View\Output.



В принципе, мы уже получили выходной файл в формате .hex, который можно загружать во флэш-память микросхемы и наблюдать работу нашего приложения: перемигивание светодиодов. Однако нам нужно показать полный цикл работы в среде «AVR-Studio», поэтому мы переходим к стадии отладки. Это делается при помощи команды \Debug\Start Debugging.

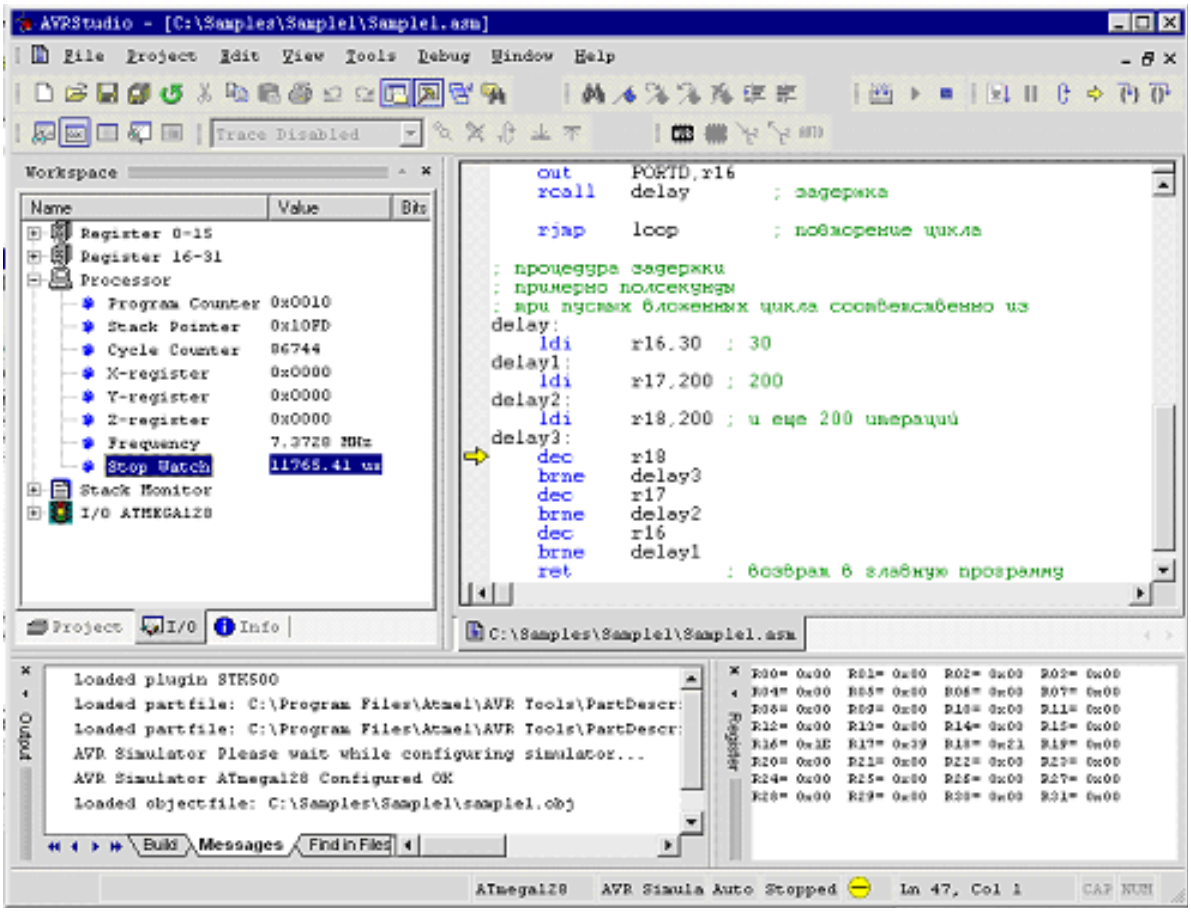


Теперь устанавливаем в окне Simulator Options” частоту кварца 7,3728 MHz для точного измерения времени выполнения программы .



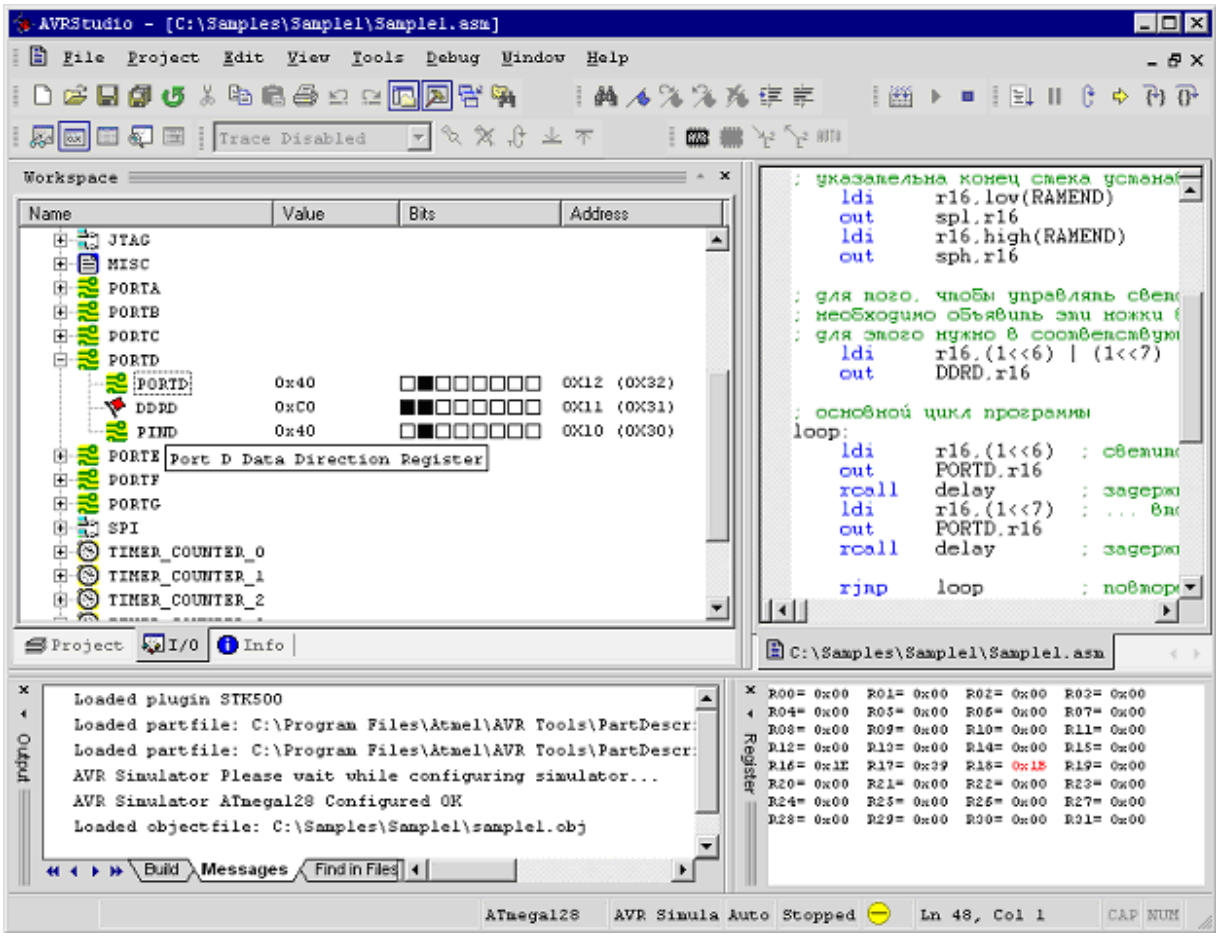
Остальные опции следует оставить без изменения. Программу удобно выполнять в пошаговом режиме при помощи мыши или клавиши F11.

Пакет «AVR-Studio» содержит мощные средства для просмотра и редактирования состояния внутренних регистров и портов ввода/вывода отлаживаемого микроконтроллера, а также время, выполнения программы . Доступ к ним осуществляется через окно “I/O”.



На самом деле, количество информации, доступное через окна просмотра пакета «AVR-Studio» настолько велико, что для получения максимального комфорта рекомендуется использовать компьютер в двухмониторной конфигурации.

Для отладки нашего примера, чтобы получить доступ к битам порта D, надо раскрыть строку I/O ATMEGA128 и затем строку PORTD. Теперь видны все три регистра этого порта, PORTD, DDRD и PIND. Чтобы увидеть поля Value, Bits и Address, придется расширить правую границу окна, потеснив при этом окно с исходным текстом программы.



Теперь, проходя программу в пошаговом режиме, можно видеть изменение текущих состояний этих регистров в поле Bits. Есть возможность оперативного изменения состояния любого бита регистров порта, причем это можно делать либо записью нового кода в поле Value, либо непосредственно, щелкнув мышью на нужном бите регистра.

Для самостоятельных упражнений, предлагается следующая программа, которая отличается от предыдущей тем, что зажиганием светодиодов управляют две кнопки.

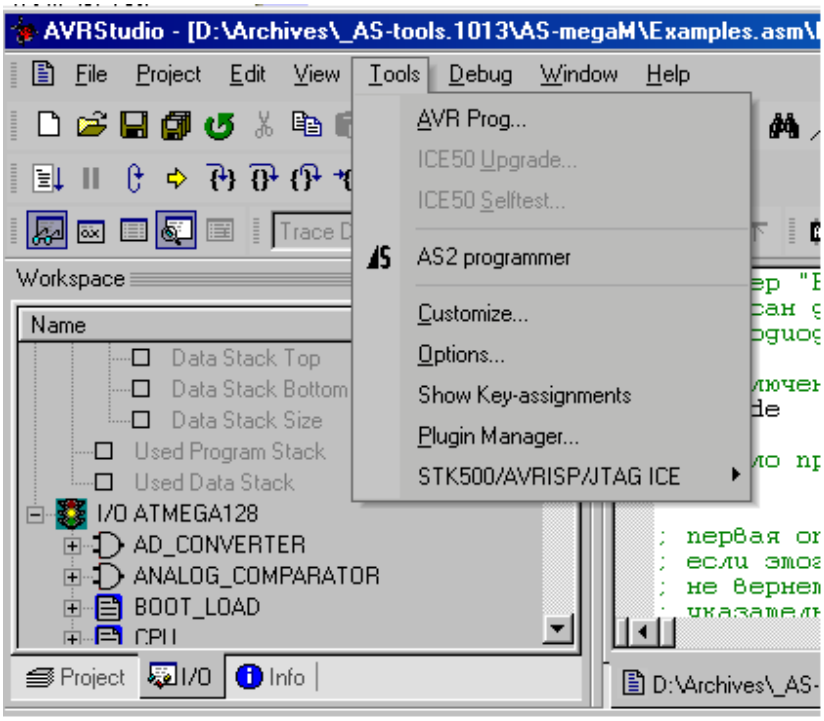
; **Пример 2**: «Управление светодиодами от кнопок »
; написан для отладочной платы AS-MegaM
; светодиоды подключены к выводам PD6 и PD7 и через резисторы - на общий провод.

```
; кнопки - на PE4 и PE5

.include "m128def.inc"

; основная программа
begin:

; инициализация стека
ldi r16,low(RAMEND)
out spl,r16
ldi r16,high(RAMEND)
out sph,r16

; инициализация светодиодов
ldi r16,(1<<6) | (1<<7)
out DDRD,r16

; инициализация выводов, к которым подключены кнопки (на вход)
; внутренние подтягивающие резисторы подключены
; для этого в PORTE нужно установить соответствующие биты в единицы
ldi r16,(1<<4) | (1<<5)
out PORTE,r16

; а в DDRE - в нули
ldi r16,0
out DDRE,r16

; бесконечный цикл
forever:
in r16,PINE ; теперь в r16 находится текущее "состояние" кнопок
com r16 ; кнопка "нажимается" нулем, поэтому инвертируем регистр
lsl r16 ; переносим биты 4,5 в позиции 6,7
lsl r16 ; и обновляем "показания" светодиодов
andi r16,(1<<6) | (1<<7)
out PORTD,r16
rjmp forever ; цикл выполняется бесконечно
```

Таким образом, на примере простейших программ показаны некоторые возможности пакета «AVR-Studio». Надо понимать, что это лишь первое знакомство, позволяющее быстрее освоиться с базовыми командами пакета. Между тем, возможности рассматриваемого пакета намного шире. Например, здесь можно отлаживать программы, написанные на языках высокого уровня. В частности, Си-компилятор фирмы ImageCraft пользуется отладчиком «AVR-

Studio» «как родным». Для этого при компиляции исходного кода надо установить опцию генерации выходного файла в формате, совместимом с «AVR-Studio». При этом появляется возможность производить отладку в исходных кодах.

Еще одна из многих характеристик пакета «AVR-Studio» - возможность подключения внешних программ. Например, для обеспечения вызова оболочки внутрисхемного программатора AS2 нужно выполнить несколько простых операций:



- В меню Tools главного окна «AVR-Studio» надо выбрать пункт Customize;
- в окне Customize выбрать пункт Tools;
- двойным нажатием кнопки мыши или нажав Insert на клавиатуре, добавить новую команду в список и назвать ее «Программатор AS2»;
- указать путь к исполняемому файлу программатора, введя его непосредственно в поле для ввода "Command", или нажав на кнопку "…" справа от этого поля.

Теперь в меню Tools появился пункт "Программатор AS2".

Средства пакета «AVR-Studio», начиная с версии 4.08, позволяют подключать вспомогательные программы – plugins. Первый plugin для «AVR-Studio» – это программа графического редактора, упрощающая процесс инициализации ЖК-индикатора, которым может непосредственно управлять AVR-контроллер ATmega169. Максимальный логический размер ЖК-индикатора составляет 100 сегментов, каждому элементу индикатора ставится в соответствие бит в специальном регистре контроллера. Чтобы упростить рутинную процедуру привязки определенных битов к каждому сегменту, можно использовать вышеупомянутую программу.

Литература

1. *Евстифеев А.В.* Микроконтроллеры AVRсемейств Tiny и Mega фирмы ATMEL, 3-е изд., М.: Издательский дом «Додэка-XXI», 2006.- 560 с.
2. *Ревич Ю.В.* Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. – СПб.: БХВ-Петербург, 2008. – 384 с.
3. *Николай Королев, Дмитрий Королев.* AVR-микроконтроллеры второго поколения: средства разработчика. // Компоненты и технологии, 2003, № 7.
4. *Трамперт В.* AVR-RISC микроконтроллеры.: Пер. с нем. – К.: “МК-Пресс”, 2006. – 464 с.

Задание 1. Управление светодиодами от кнопок, используя прерывание:

1. Взяв за основу *пример 2* составить блок-алгоритм программы.
2. Написать программу на ассемблере.
3. Провести отладку программы на симуляторе “AVR-Studio”.
4. Загрузить программу во флэш-память микроконтроллера с помощью программатора “AS-2”.
5. Проверить работу приложения.

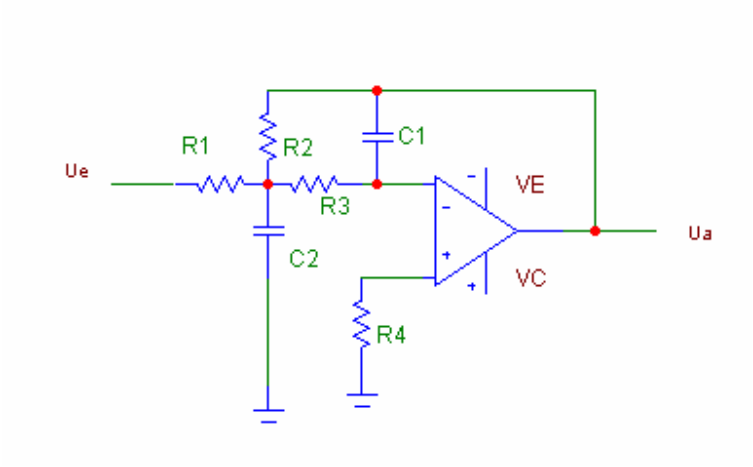
Задание 2. Управление частотой мигания светодиодов по интерфейсу RS-232:

1. В качестве консольной программы управления COM портом использовать утилиту «Com Port Toolkit» (ознакомиться с ее применением).
2. Для включения/выключения светодиодов за основу можно взять процедуры из *примера 2*.
3. Составить блок-алгоритм и написать программу на ассемблере.
4. Провести отладку программы на симуляторе “AVR-Studio”.
5. Загрузить программу во флэш-память микроконтроллера с помощью программатора “AS-2”.
6. Проверить работу приложения.

Задание 3. Формирование треугольного периодического сигнала с помощью таймера в режиме ШИМ:

1. Изучив работу счетчика-таймера в режиме широтно-импульсной модуляции (ШИМ), составить блок-алгоритм программы.
2. Написать программу на ассемблере.
3. Провести отладку программы на симуляторе “AVR-Studio”.

4. Загрузить программу во флэш-память микроконтроллера с помощью программатора “AS-2”.
5. Выход таймера подключить к фильтру Баттерворта (нижних частот, второго порядка), который монтируется на монтажной плате «WishBoard».



Усиление: $A_0 = -\frac{R_2}{R_1}$; Предельная частота: $f_g{}^2 = \frac{1}{4\pi^2 C_1 C_2 R_2 R_3}$; $C2 \geq 4C_1$

6. Сигнал наблюдать с помощью цифрового осциллографа “GDS-2102”.

Задание 4. Управление 3-х фазным шаговым электродвигателем:

1. С помощью счетчика-таймера сформировать опорную частоту, определяющую скорость вращения шагового двигателя.
2. Получить 3-х фазную последовательность прямоугольных импульсов со сдвигом между фазами 120 градусов. В качестве тактовой частоты используется опорная частота со счетчика-таймера.
3. С помощью USART контроллера и консоли на основе утилиты «ComPort Toolkit» реализовать управление шаговым двигателем, задавая скорость, число шагов и направление вращения.
4. Составить блок-алгоритм работы приложения.

5. Написать программу на ассемблере.
6. Провести отладку программы на симуляторе “AVR-Studio”.
7. Загрузить программу во флэш-память микроконтроллера с помощью программатора “AS-2”.
8. Проверить управляющие сигналы с выходов порта (частоту, фазовый сдвиг) с помощью цифрового осциллографа «GDS-2102».
9. Подключить драйвер двигателя ШД 300/300 к выходам контроллера, к обмоткам двигателя и источнику питания.
10. Проверить работу привода.

Приложение 1. Описание платы AS-megaM.

Приложение 2. Ассемблер для 8-разрядных AVR-контроллеров.