# Микропроцессоры.

**Микропроцессор** — это центральный блок персонального компьютера или вычислительной системы, предназначенный для выполнения арифметических и логических операций над информацией и управления работой всех остальных блоков. Микропроцессор (**МП**) реализуется в виде одной микросхемы.

Структурная схема микропроцессорной системы или ЭВМ представлена на рис 1.

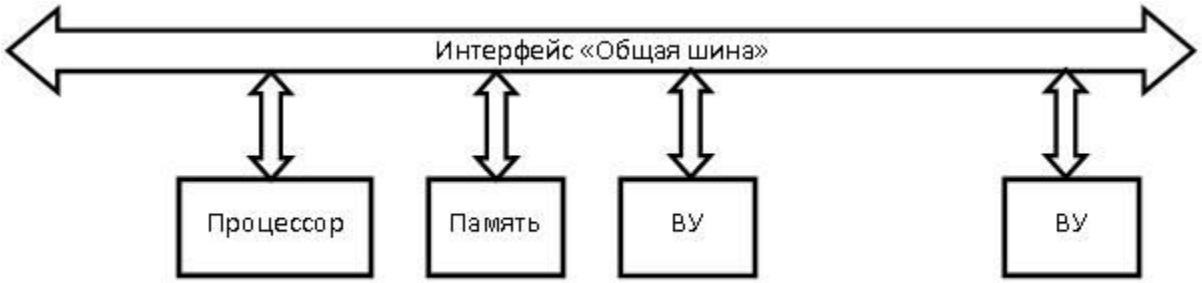



Рис. 1. Структурная схема микропроцессорной системы

В настоящее время существует несколько архитектур вычислительных машин. Термин «архитектура» носит двойной смысл.

В первом случае архитектура охватывает понятие организации системы, включающее структуру памяти, системной шины, организацию ввода/вывода и т.п.

Здесь при реализации микропроцессоров традиционно используется два подхода к построению архитектуры:

- Архитектура фон Неймана;
- Гарвардская архитектура.

Во втором случае под архитектурой понимается архитектура набора команд, исполняемых микропроцессором.

Здесь различают две основных архитектуры:

- CISC архитектура – архитектура с расширенным набором команд (Complex Instruction Set Computer);
- RISC архитектура – архитектура с сокращенным набором команд (Reduced Instruction Set Computer).

Архитектура ЭВМ фон Неймана базируется на двух основных идеях:

1. программа вычислений вводится в ЭВМ и хранится в той же памяти, что и обрабатываемые данные;
2. Команда представляется в виде двоичного кода и ничем не отличается от обрабатываемых данных.

Выборка команды и ее выполнение в ЭВМ фон-неймановского типа описывается с помощью так называемого цикла управления фон Неймана.

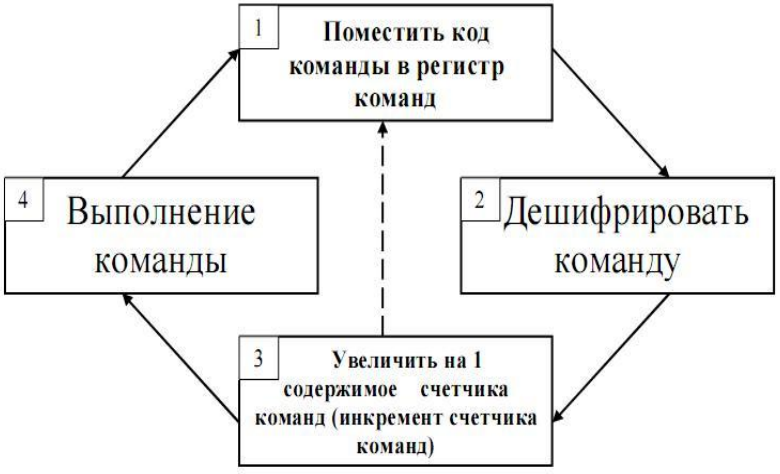



Рис.2. Цикл управления фон Неймана

В фазе выборки команды 1 содержимое ячейки памяти (ЯП) выбирается из памяти и помещается в регистр команд (РК). В фазе 2 код команды поступает из РК в дешифратор команд, где он преобразуется в систему управляющих сигналов. После этого счетчик команд инкрементируется в фазе 3, по его содержимому будет адресоваться уже следующая ячейка памяти. Если команда состоит из нескольких байтов, то фазы 1 - 3 повторяются до полного извлечения команды из памяти. После этого процессор переходит к фазе 4 выполнения команды. Когда команда выполнена, происходит возврат к фазе 1 и осуществляется выборка следующей команды.

Таким образом, процессор ЭВМ работает циклично, то есть непрерывно выполняет одни и те же действия: извлекает команды из последовательно расположенных ячеек памяти и выполняет их.

В следствии выше сказанного любой процессор фоннеймановского типа должен содержать счетчик команд.

**В качестве недостатка архитектуры фон Неймана можно назвать возможность непреднамеренного нарушения работоспособности системы (программные ошибки) и преднамеренное уничтожение ее работы (вирусные атаки).**

В Гарвардской архитектуре принципиально различаются два вида памяти микропроцессора:

- память программ (для хранения инструкций микропроцессора);
- память данных (для временного хранения и обработки переменных).

В связи с раздельной памятью в Гарвардской архитектуре используются две шины

- шина команд;
- шина данных.

В гарвардской архитектуре принципиально невозможно осуществить операцию записи в память программ, что исключает возможность случайного разрушения управляющей программы в случае ошибки программы при работе с данными или атаки третьих лиц.

Достоинства гарвардской архитектуры следующие :

1. Применение небольшой по объему памяти данных способствует ускорению поиска информации в памяти и увеличивает быстродействие МП.
2. Гарвардская архитектура позволяет организовать параллельное выполнение программ – выборка следующей команды может происходить одновременно с выполнением предыдущей в результате чего сокращается время выборки команды.

Недостатком гарвардской архитектуры является усложнение архитектуры МП и необходимость генерации дополнительных управляющих сигналов для памяти команд и памяти данных.

Для CISC-архитектуры типичны разнообразие способов адресации, различное число байт в команде и в следствие этого различное время выполнения команд.

В системах с RISK-архитектурой длина всех команд одинакова и они все выполняются за одинаковое число тактов (обычно за один такт).

До появления микропроцессоров процессор состоял из нескольких плат со многими элементами. Развитие интегральной технологии позволило все эти платы и элементы разместить на одном кристалле – так появился микропроцессор.

Первыми появились микропроцессоры фирмы Intel – 4 разрядный МП Intel 4004 в 1971 г. и 8 разрядный МП Intel 8080 в 1974 г. (отечественный аналог КР580ВМ80А - 1977 г.) На МП Intel 8080 уже можно было создавать микро-ЭВМ. Эти микро-ЭВМ имели значительно меньшие скорость и объем памяти, чем большие ЭВМ. На больших ЭВМ процессор состоял из нескольких плат. В микро-ЭВМ все эти платы заменила одна микросхема называемая микропроцессор – процессор выполненный на одном кристалле.

Первый МП 8080 - результат предельного упрощения структуры процессора ЭВМ до уровня, который мог быть реализован средствами интегральной технологии того времени. Процессор выпускался по новейшей тогда 6-микронной NMOS технологии, что позволило разместить на кристалле 6000 транзисторов.

С развитием интегральной технологии число транзисторов на кристалле росло. Увеличивались частота и скорость обработки данных. Появились множество типов МП, предназначенных для разных целей.

В настоящее время выпускается большое количество типов МП (около 500) и каждый тип микропроцессора находит применение в своей области. Все типы микропроцессоров можно разбить на 3 группы:

- универсальные микропроцессоры;
- микроконтроллеры;
- цифровые сигнальные процессоры (DSP)

Универсальные микропроцессоры имеют широкий набор команд, большое число методов адресации, возможность адресации достаточно большого объема памяти. Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. Эти процессоры в основном предназначены для построения ЭВМ.

Первый универсальный процессор – Intel 8080. На базе микропроцессора Intel 8080 фирмой MITS был выпущен «первый в мире миникомпьютерный комплект» (персональный компьютер) Altair-8800, который пользовался невероятно большой по тем временам популярностью (MITS не успевала даже вовремя обрабатывать).

На основе процессора Intel 8080 в дальнейшем строились процессоры линейки X86 фирмы Intel по принципу обратной совместимости. При этом увеличивалась разрядность шин и регистров, расширялся набор команд.

В связи с этим, для понимания работы микропроцессоров и их отличий рассмотрим структурную схему и работу МП Intel 8080.

Микропроцессор имеет 8-разрядную шину данных (поэтому он относится к 8-разрядным МП) и 16-разрядную шину адреса. 16-разрядная шина адреса позволяет адресоваться к $2^{16}$ = 65536 ячейкам памяти. Адресация к внешним устройствам (портам) осуществляется отдельным от памяти адресным пространством по 8-разрядной шине. Возможно подключение к МП $2^8$ = 256 внешних устройств.

Структурная схема МП Intel 8080 приведена на рис. 3.

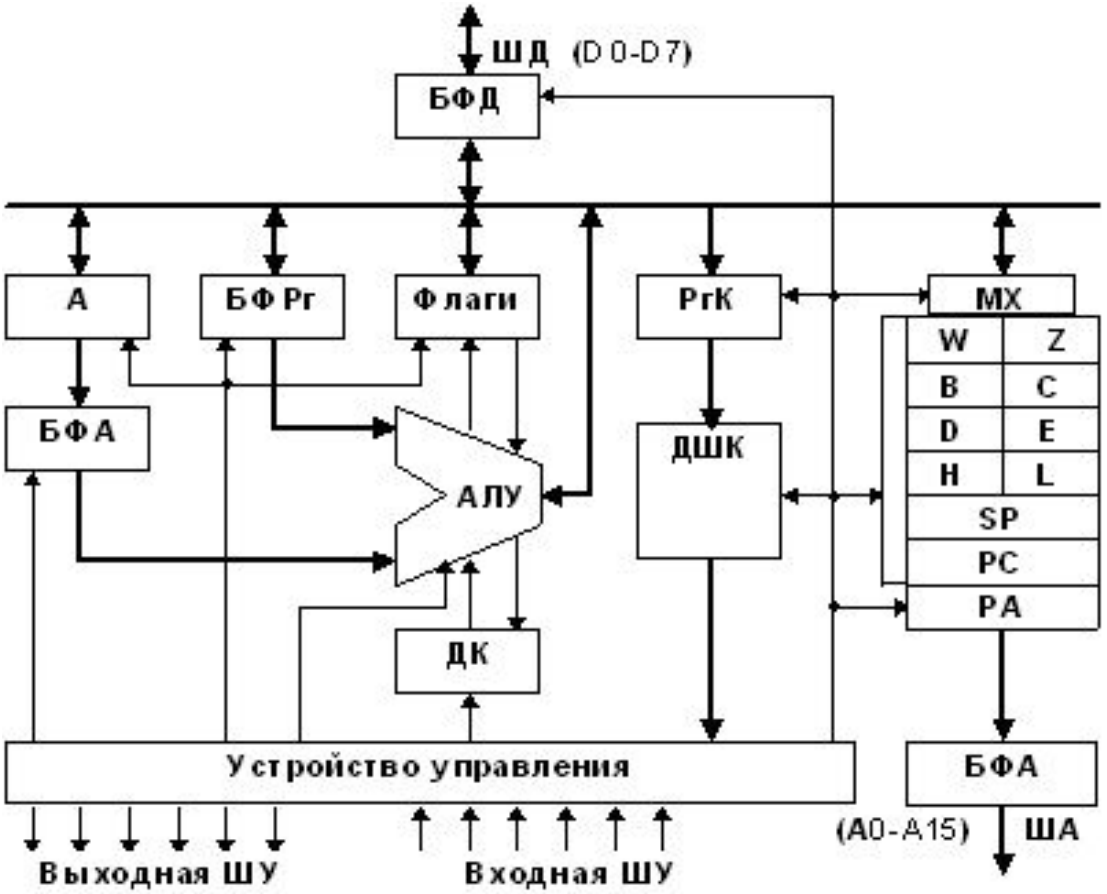



Рис 3. Структурная схема микропроцессора Intel 8080

МП состоит из Операционного устройства (ОУ) и Устройства управления (УУ). В ОУ осуществляются все операции с данными( арифметические, логические, пересылка данных из регистра в регистр и т.д.) Основными элементами ОУ являются арифметико-логическое устройство (АЛУ) и набор из 7 регистров.

Регистр А имеет особое значение – в нем сохраняются (аккумулируются) большинство арифметических и логических операций. Также через него происходит обмен данными с ячейками памяти. Поэтому этот регистр имеет название – аккумулятор. Остальные шесть регистров называются регистрами общего применения (РОН) и могут составлять пары B-C, D-E, H-L для работы с 16-разрядными числами и адресами.

Флаги на схеме – это регистр признаков. После арифметических и логических операций формируются признаки результата. Всего 5 признаков . Например, если при вычитании результат больше нуля, то триггер нуля установится в «1», если меньше нуля, то в «0». Признаки, их называют флагами, используются в основном в командах условного перехода.

Далее есть регистр стека **SP** – в него записывается адрес вершины стека. Стек это особое место в памяти, куда записывается для хранения адрес команды перед переходом к подпрограмме или перед выполнением прерывания.

**PC** – счетчик команд. Счетчик команд всегда содержит адрес следующей выполняемой команды. Со счетчика адреса адрес поступает в регистр адреса **PA** и через буферный регистр **БФА** на шину адреса. После выполнения каждой команды счетчик команд автоматически инкрементируется и указывает на следующую выполняемую команду. Таким образом достигается автоматизм выполнения программы.

Прерывания. Любое внешнее устройство может запросить МП, чтобы МП его обслужил. Для этого внешнее устройство выставляет сигнал запроса на специальный вход МП - вход запроса прерывания. Если МП в это время выполнял какую-то программу, то он прерывает ее выполнение, запоминает в стеке адрес команды, на которой он прервался, и выполняет подпрограмму прерывания.

После выполнения подпрограммы прерывания МП извлекает из стека адрес команды, на которой он остановился и как ни в чем небывало продолжает выполнять основную программу.

Еще один важный элемент МП – это дешифратор команд ДШК. Каждая команда состоит из кода операции (КОП) и операнда. В коде операции закодировано, что должен делать МП в данной команде. При извлечении команды из памяти КОП заносится в ДШК, где он дешифруется и поступает в управляющее устройство (УУ). Получив расшифрованный КОП УУ выдает управляющие сигналы на все внутренние узлы, а так же и на внешние устройства. Команда выполняется за несколько тактов и в каждом такте выдаются свои сигналы управления. Команда может состоять из 1-го, 2-х или 3- х байт, но первый байт всегда КОП. Команды могут выполняться за время от 4–х до 10 тактов задающего генератора. Универсальные процессоры относятся к микропроцессорам с CISC архитектурой.

## МИКРОКОНТРОЛЛЕРЫ

На МП 8080, как упоминалось выше, собирались микрокомпьютеры, но он также широко применялся в различных семах управления, например в светофорах. Но к этому МП для работы нужны были дополнительные устройства – микросхемы постоянной памяти, ОЗУ, буферные регистры, интерфейсы портов и т.п. И, к тому же, не требовался микропроцессор такой универсальности и большим объемом памяти. Поэтому появились микросхемы, где на одном кристалле были сам процессор, небольшая память (ПЗУ и ОЗУ), таймеры, последовательный интерфейс, порты ввода вывода и другие устройства. Такое устройство стали называть однокристальной микроЭВМ. Но потом, так как они применялись в различных устройствах управления, контроля, стали называть микроконтроллерами.

Однокристальные микроконтроллеры или просто микроконтроллеры (МК) предназначены для использования в системах промышленной и бытовой автоматики. Они представляют собой большие интегральные схемы, которые включают в себя все устройства, необходимые для реализации цифровой системы управления минимальной конфигурации: процессор (как правило, целочисленный), ЗУ команд, ЗУ данных, генератор тактовых сигналов, программируемые устройства для связи с внешней средой (контроллер прерывания, таймеры-счетчики, разнообразные порты ввода/вывода), иногда аналого-цифровые и цифро-аналоговые преобразователи и т. д.

Первым микроконтроллером фирмы Intel был микроконтроллер 8051, который нашел очень широкое применение и выпускался в вариантах с различным набором внешних устройств. МК стали выпускать фирма Моторола и другие фирмы. В настоящее время различными фирмами выпускается несколько сотен различных типов микроконтроллеров. Среди разработчиков систем управления, цифровых устройств обработки данных можно выделить три типа наиболее популярных микроконтроллеров:

- микроконтроллеры с ядром MCS-51;
- PIC микроконтроллеры фирмы MicroChip;
- AVR микроконтроллеры фирмы Atmel.

Микроконтроллеры с ядром MCS-51 – микроконтроллеры, архитектура и система команд которых подобна микроконтроллеру 8051.

PIC микроконтроллеры – это уже микроконтроллеры с Гарвардской архитектурой и RISK системой команд, несколько более удобные для программирования и для

PIC микроконтроллеры – это уже микроконтроллеры с Гарвардской архитектурой и RISK системой команд, несколько более удобные для программирования и для использования в различных встраиваемых системах.

AVR микроконтроллеры - это  микроконтроллеры тоже с Гарвардской архитектурой и более развитой RISK системой команд, более быстродействующие. Эти МК в настоящее время наиболее часто используются разработчиками. Микроконтроллеры AVR разрабатывались с учетом того, чтобы у них получался более эффективный код при компиляции  с языка Си. Поэтому при разработке больших программ удобнее писать программы на Си, чем на ассемблере. Для AVR микроконтроллеров разработаны мощные и удобные системы программирования и отладки.

## AVR микроконтроллеры

Компания Atmel выпускает обширную линейку восьмиразрядных микроконтроллеров на базе AVR ядра, разбитые на несколько подсемейств, различающиеся по техническим характеристикам, области применения, цене :

ATtiny – семейство AVR микроконтроллеров оптимизированных для приложений, требующих относительно большой производительности (до 1.0 MIPS и способны работать на частотах до 20.0 МГц), энергоэффективности (ATtiny единственное семейство способное работать от 0,7В напряжения питания!) и компактности (есть микроконтроллеры в корпусе – всего 6 пин, и при этом каждый пин обладает несколькими функциями, к примеру: порт ввода/вывода, вход АЦП, вывод ШИМ и т.д.). Отсюда вырисовывается и область их применения : устройства критичные к цене, энергопотреблению, габаритам и т.д.

ATmega – семейство AVR микроконтроллеров предназначенных для использования в самых разнообразных областях, благодаря очень большому набору периферийных устройств, большому объему памяти программ, портов ввода/вывода и пр. Одним словом – есть где развернутся.

ATxmega – новое семейство AVR микроконтроллеров с еще большим набором периферийных устройств чем у ATmega (добавилось устройство прямого доступа к памяти, ЦАП, CRC-модуль, полноценный USB интерфейс, более быстрый АЦП и др.), с рабочими частотами до 32.0МГц.

Все вышеперечисленные семейства имеют единую архитектуру (что позволяет с легкостью переносить код с одного микроконтроллера на другой), выпускаются как в DIP, так и SMD корпусах.

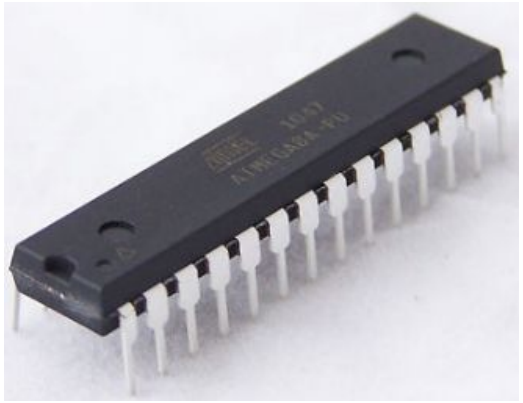


Рис.4. ATMega8 в DIP-корпусе

Рис.5. ATtiny2313 в DIP-корпусе

Для первоначального программирования МК выберем МК ATtiny2313 (AT90S2313), имеющий 20 вывода.

Микросхема ATtiny2313 имеет следующие особенности:

- 2 Кбайт системной программируемой Flash-памяти программ;

- 128 байт EEPROM;
- 128 байт SRAM (ОЗУ);
- 18 линий ввода-вывода (I/O);
- 32 рабочих регистра;
- однопроводной интерфейс для внутрисхемной отладки;
- два многофункциональных таймера/счетчика с функцией совпадения;
- два многофункциональных таймера/счетчика с функцией совпадения;
- поддержка внешних и внутренних прерываний;
- последовательный программируемый USART-порт;
- программируемый сторожевой таймер с внутренним генератором;
- три программно изменяемых режима энергосбережения.

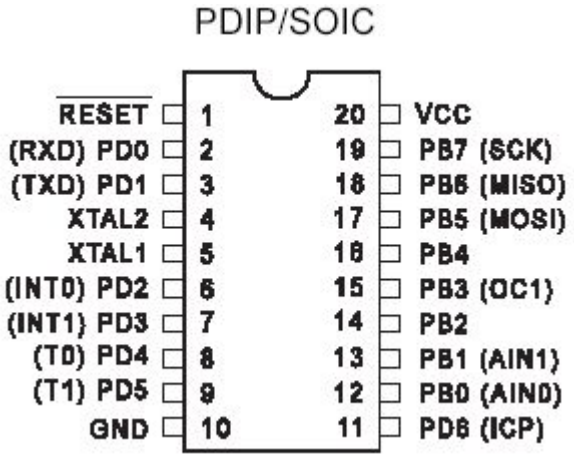



Рис.6. Расположение выводов AT90S2313

**Описание выводов.**

GND - земля (-)

Port B, Port D являются двунаправленнымы портами ввода/вывода с внутренними подтягивающими резисторами.

RESET - Вход сброса. Удержание на входе низкого уровня в течение двух машинных циклов (если работает тактовый генератор), сбрасывает устройство.

XTAL1 - Вход инвертирующего усилителя генератора и вход внешнего тактового сигнала.

XTAL2 - Выход инвертирующего усилителя генератора.

VCC - Вывод источника питания (+5 В).

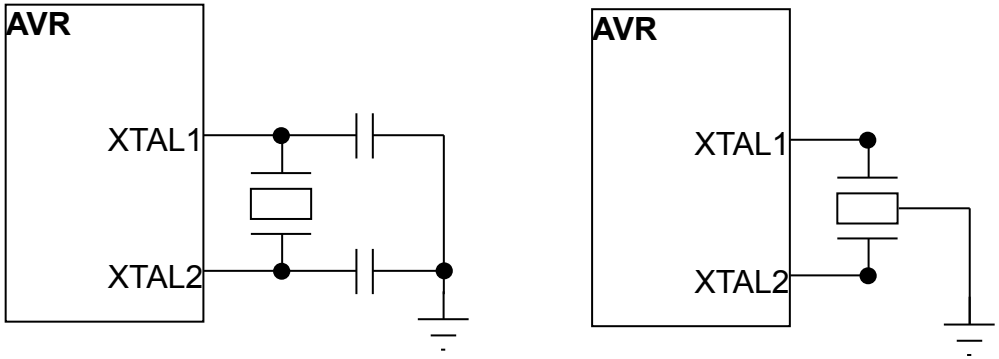



**Рис. 7.** Схемы подключения к AVR керамического и кварцевого резонаторов
а) без встроенных конденсаторов и б) со встроенными конденсаторами

Также AVR может синхронизироваться внешним тактовым сигналом, поступающим на вход XTAL1. Последний способ синхронизации – использование встроенного RC-генератора. Такой генератор на данный момент есть только в AT90S1200, он обеспечивает тактовую частоту 1 МГц. Этот режим задается при помощи одного из битов конфигурации (fuse bit) только при параллельном программировании МК.

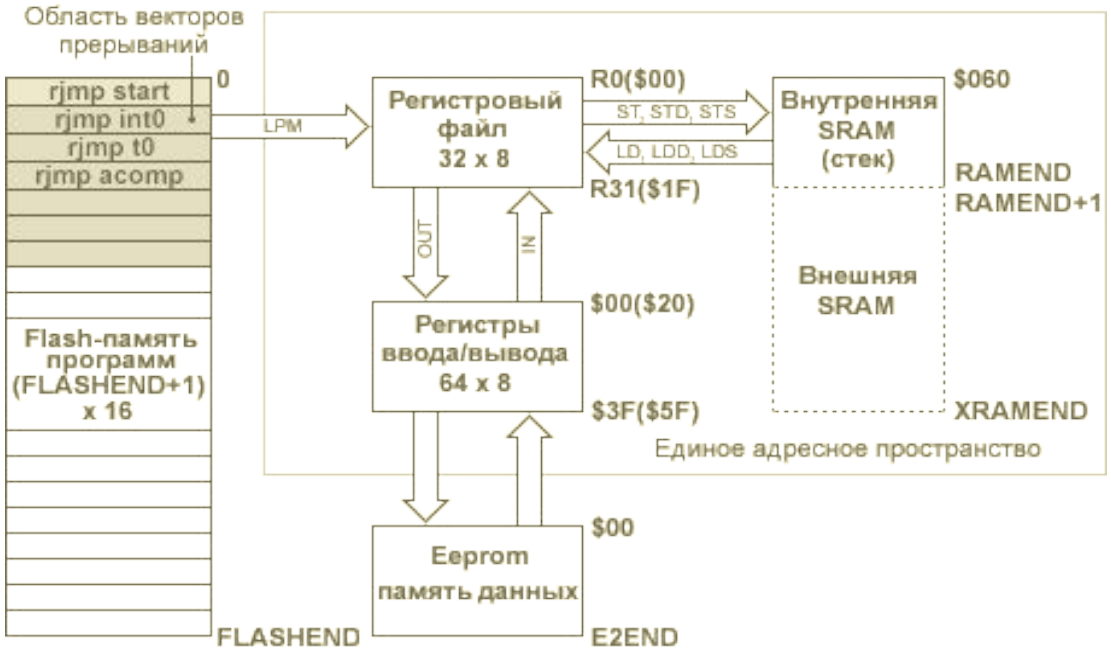



**Рис. 8.** Программная модель AVR-микроконтроллеров

Регистровый файл, блок регистров ввода/вывода и оперативная память, как показано на рис.8., образуют единое адресное пространство, что дает возможность при программировании обращаться к 32 оперативным регистрам и к регистрам ввода/вывода как к ячейкам памяти, используя команды доступа к SRAM (в том числе и с косвенной адресацией).

Все арифметические и логические операции, а также часть операций работы с битами выполняются в АЛУ только над содержимым РОН. Следует обратить внимание, что команды, которые в качестве второго операнда имеют константу (SUBI, SBCI, ANDI, ORI, SBR, CBR), могут использовать в качестве первого операнда только регистры из второй половины РОН (R16-R31).

Команды 16-разрядного сложения с константой ADIW и вычитания константы SBIW в качестве первого операнда используют только регистры R24, R26, R28, R30.

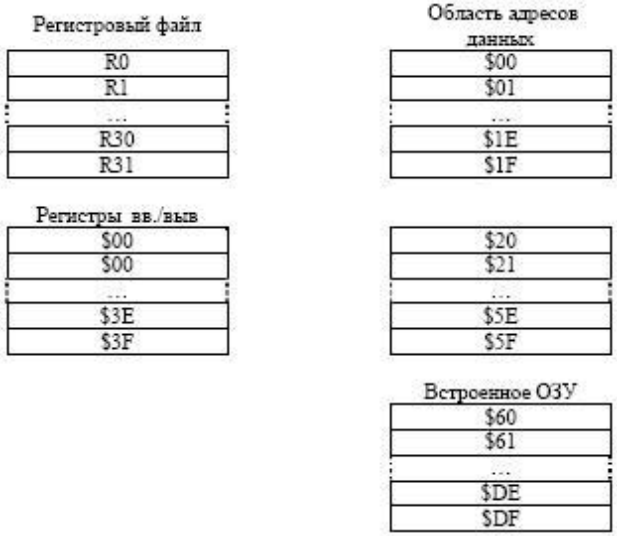



Рис. 9. Память данных

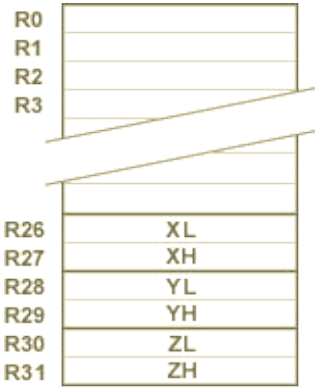



Рис. 10. Регистровый файл или РОН.

Все РОН непосредственно доступны АЛУ. Старшие регистры (см. рис. 9.) объединены парами и образуют три 16-разрядных регистра, предназначенных для косвенной адресации ячеек памяти (AVR без SRAM имеют только один 16-битный регистр Z).

Младшие 32 адреса ($0 - $1F) соответствуют оперативным регистрам т.е. РОН. Следующие 64 адреса ($20 - $5F) зарезервированы для регистров ввода/вывода. Внутренняя SRAM у всех AVR начинается с адреса $60. Таким образом, регистры ввода/вывода имеют двойную нумерацию. Если используются команды IN, OUT, SBI, CBI, SBIC, SBIS, то следует использовать нумерацию регистров ввода/вывода, начинающуюся с нуля (назовем ее основной). Если же к регистрам ввода/вывода доступ осуществляется как к ячейкам памяти, то необходимо использовать нумерацию единого адресного пространства оперативной памяти данных AVR.

Кроме оперативной памяти программно доступными ресурсами микроконтроллера являются энергонезависимые, электрически программируемые FLASH и EEPROM блоки памяти, которые имеют отдельные адресные пространства.

Младшие адреса памяти программ имеют специальное назначение. Адрес $0000 является адресом, с которого начинает выполняться программа после сброса процессора. Начиная со следующего адреса $0001, ячейки памяти программ образуют область векторов прерывания. В этой области для каждого возможного источника прерывания отведен свой адрес, по которому (в случае использования данного прерывания) размещают команду относительного перехода RJMP на подпрограмму обработки прерывания (см. рис. 8). Следует помнить, что адреса векторов прерывания одних и тех же аппаратных узлов для разных типов AVR могут иметь разное значение. Поэтому для обеспечения переносимости программного обеспечения удобно, так же как и в случае с регистрами ввода/вывода, использовать символические имена адресов векторов прерывания, которые определены в соответствующем **inc**-файле.

EEPROM блок электрически стираемой памяти данных AVR предназначен для хранения энергонезависимых данных, которые могут изменяться непосредственно на объекте. Это калибровочные коэффициенты, различные установки, конфигурационные параметры системы и т. п. EEPROM-память данных может быть программным путем как считана, так и записана. Однако специальных команд обращения к EEPROM-памяти нет. Чтение и запись ячеек EEPROM выполняется через регистры ввода/вывода EEAR (регистр адреса), EEDR (регистр данных) и EECR (регистр управления).

## Признаки результата операции - флаги

Во время выполнения арифметических и логических операций или операций работы с битами АЛУ формирует те или иные признаки результата операции, то есть устанавливает или сбрасывает биты в регистре состояния SREG (Status Register) (см. рис. 11).

Признаки результата операции могут быть затем использованы в программе для выполнения дальнейших арифметико-логических операций или команд условных переходов.

## Регистр состояния

Регистр состояния – SREG является частью пространства ввода/вывода и расположен по адресу $3F. В нем устанавливаются признаки результата арифметических операций. Отдельные биты регистра имеют следующее назначение (см. рис. 2.5)

| Символьное обозначение | I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|---|
| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Рис. 11.** Регистр состояния SREG (Status Register)

Рассмотрим некоторые из признаков:

Бит 0 – **C** (carry) Флаг переноса. Устанавливается, если во время выполнения операции был перенос из старшего разряда результата;

Бит 1 – **Z** (zero) Флаг нулевого результата. Устанавливается, если результат операции равен 0;

Бит 2 – **N** Флаг отрицательного результата. Устанавливается, если MSB (Most Significant Bit - старший бит) результата равен 1 (правильно показывает знак результата, если не было переполнения разрядной сетки знакового числа);

Бит 3 – **V** Флаг переполнения дополнения до двух. Устанавливается, если во время выполнения операции было переполнение разрядной сетки знакового результата;

Бит 4 – **S** Бит знака, **S = N** XOR **V**. Бит **S** всегда равен исключающему ИЛИ между флагами N (отрицательный результат) и V (переполнение дополнения до двух). Правильно показывает знак результата и при переполнении разрядной сетки знакового числа;

Бит 5 – **H** Флаг половинного переноса. устанавливается, если во время выполнения операции был перенос из 3-го разряда результата.

Бит 6 - **T** Хранение копируемого бита. Команды копирования битов BLD (Bit LoaD) и BST (Bit STore) используют этот бит как источник и приемник обрабатываемого бита. Бит из регистра регистрового файла может быть скопирован в **T** командой BST, бит **T** может быть скопирован в бит регистрового файла командой BLD.

Бит 7 **- I** Общее разрешение прерываний. Для разрешения прерываний этот бит должен быть установлен в единицу. Управление отдельными прерываниями производится регистром маски прерываний - GIMSK/TIMSK. Если флаг сброшен (0), независимо от состояния GIMSK/TIMSK, прерывания не разрешены. Бит **I** очищается аппаратно после входа в прерывание и восстанавливается командой RETI, для разрешения обработки следующих прерываний

## Прерывания.

Если произошел запрос прерывания, и флаг **I** в регистре состояния установлен в 1, то адрес следующий команды сохраняется в стеке, а выполнения программы продолжается с адреса, хранящимся в соответствующем векторе прерывания. Когда запрос прерывания получен, и программа перешла по этому вектору (адресу), флаг **I** сбрасывается в 0, чтобы предотвратить возможность вызова нового прерывания во время обработки текущего прерывания.

## Таймеры.

Источником сигнала переключения таймеров/счетчиков является либо тактовая частота процессора, либо внешний синхросигнал. Тактовая частота процессора может использоваться непосредственно или предварительно делиться. Выбор источника сигнала и коэффициента деления производится с помощью мультиплексора Биты CSxn, управляющие мультиплексором, расположены в регистре управления таймера TCCR0. В AT90S2313, который содержит два таймерных блока, имеется два мультиплексора – по одному на каждый блок. В AT90S1200 есть только один таймерный блок и один мультиплексор. Содержимое таймера инкрементируется при поступлении переднего фронта переключающего сигнала. Поэтому синхросигнал со внешнего вывода МК поступает в мультиплексор в прямом и инвертированном. Значение внешнего сигнала проверяется при поступлении переднего фронта тактового сигнала процессора.

## Сторожевой таймер.

Сторожевой таймер представляет собой отдельный таймер с собственной частотой на 1 МГц, который при включении будет отсчитывать нужный интервал времени. Если произойдет переполнение до того, как команда WDR сбросит таймер в 0, то производиться перезапуск МК. Сторожевой таймер включается при установке в 1 бита WDE в регистре управления сторожевого таймера WDTCR. Содержимое битов WDRn этого регистра определяет интервал времени до того, как сторожевой таймер произведет перезапуск МК.

# Программирование AVR микроконтроллера

Для программирования микроконтроллеров AVR и отладки программ разработаны интегрированные среды разработки программ, повышающие производительность труда. Наиболее мощной и удобной является AVR Studio. Этот программный продукт включает в себя:

- встроенный редактор текста, для набора исходного кода программы,
- транслятор с языка ассемблер,
- программный симулятор ЦПУ, памяти и устройств ввода/вывода
- поддержку внешних устройств, таких как – внутрисхемный эмулятор (ВСЭ) ICEPRO, ICE200 и программатора типа AVRISP и STK500/501.

Компания Atmel распространяет данный продукт совершенно бесплатно, что является огромным преимуществом в начале освоения программирования МК и изучении его архитектуры.

Здесь можно легко писать исходный текст программы на ассемблере, подключить внешний компилятор для С, отлаживать текст написанной программы используя встроенный программный симулятор. И в конечном итоге получается hex-файл, которым можно запрограммировать свой МК программатором AVRISP или STK500/501.

В настоящее время очень удобно и быстро можно разработать устройство и отладить программу работы с этим устройством, используя программы электронного моделирования. Одна из таких программ - Proteus позволяет моделировать схемы с использованием микроконтроллеров.

Для создания первой программы соберем в программе моделирования Proteus простую схему – подключим к одному из выводов(портов) МК светодиод(LED), и будем его включать-выключать тумблером.

# Порты ввода/вывода

В МК AT90S2313 имеются три двунаправленных параллельных портов ввода/вывода: порт B и портD. Для обслуживания каждого порта отведено по три регистра.

Для порта B: регистр данных PORTB, регистр направления данных - DDRB и выводы порта B – PINB.

Регистр PORTB предназначен для чтения и записи данных, регистр DDRB определяет направление передачи данных. Если в бит порта DDRB записать 0 – вывод порта сконфигурирован как вход, если записать 1 - вывод порта сконфигурирован как выход.

Если вывод порта сконфигурирован как вход и в соответствующий бит порта PORTB записана 1, то на этом входе включается подтягивающий резистор.

PINB не является регистром, по этому адресу осуществляется доступ к физическим значениям каждого порта B.

Для порта D также имеются три таких же регистра.

Непосредственная запись константы в эти регистры невозможна, поэтому константа вначале записывается в какой-либо регистр регистрового файла, например R16, командой:

```
ldi R16,0b11111111   ;определяем все выводы порта B, как выходы. Затем R16
```

записываем в регистр направления DDRB командой:

```
out DDRB,R16        ; DDRB – регистр направления передачи порта B.
```

Такие же команды надо сделать для порта D в нужной ему конфигурации.

Одни выводы порта B (и D) можно запрограммировать как входы, другие как выходы. Почти все выводы (ножки) микроконтроллера имеют еще альтернативные функции, как показано на рис.6.

# Схема включения-выключения светодиода в Proteus

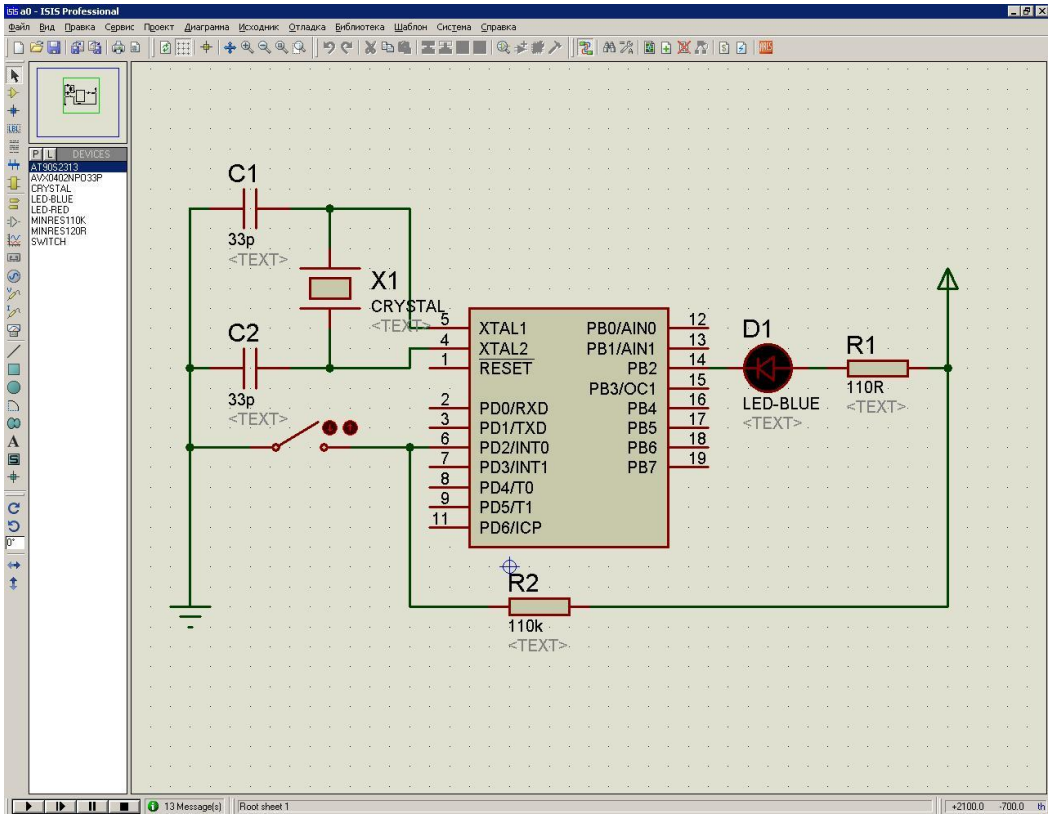



Рис.12. Включение светодиода тумблером

Здесь резистор R2 поставили для того, чтобы при разомкнутом тумблере на выводе PD2 была лог.1(+5В).

При выводе на ножку PB2 порта B лог.0 светодиод загорается.

# Включение и выключение светодиода тумблером через ввод-вывод

В программе будем считывать данные с ножки PD2 и выводит на ножку PB2.

**Программа 1**

```
.nolist                    ; чтобы не выводил в листинг многие  equ =…
.include "2313def.inc"  ; подключает файл "перевода имен"
.list                      ; делает листинг программы
.def     temp=R16          регистру R16 дается имя  temp

    ldi R16,0b11111111   ;определяем порт B как выход - светодиод
      out DDRB,R16         ; DDRB – регистр направления передачи  порта B
    ldi temp,0b11111011    ;порт D2-вход  – тумблер, остальные выходы.
      out DDRD,temp        ; DDRD – регистр направления передачи  порта D

m:     in R17,PinD          ; ввод с порта D
       out PortB,R17        ; вывод в порт B
rjmp m
```

# Использование подтягивающего резистора

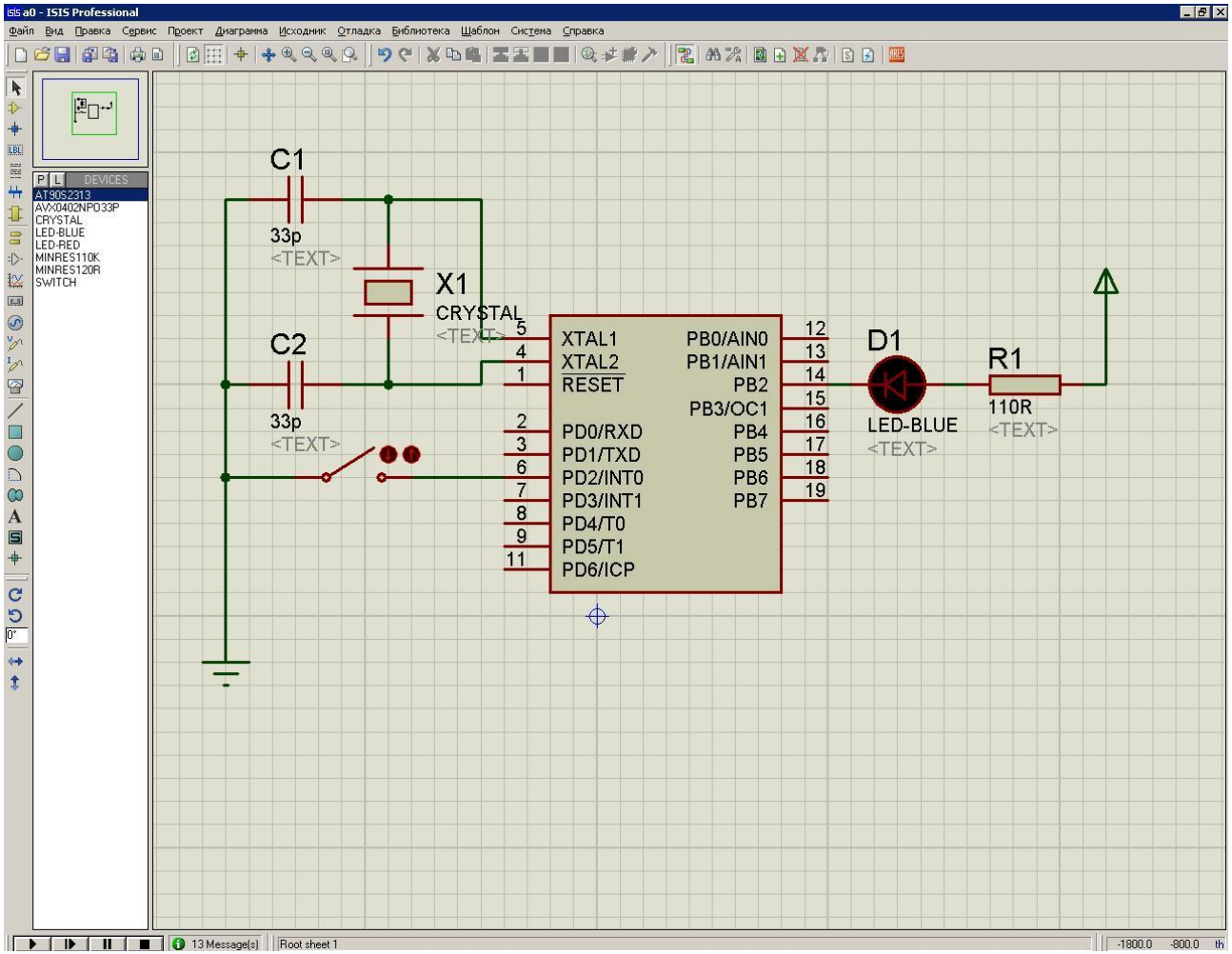



Рис.13. Включение светодиода тумблером с подтягивающим резистором

**Программа 2** Использование подтягивающих резисторов
(убрали резистор с вывода D2)

```
.nolist                    ; чтобы не выводил в листинг многие  equ =…
.include "2313def.inc"  ; подключает файл "перевода имен"
.list                      ; делает листинг программы

   ldi R16,0b11111111   ;определяем порт B как вых. - светодиод
     out DDRB,R16        ; DDRB – регистр направления передачи  порта B
   ldi R16,0b11111011   ;порт D2-вх – тумблер, стальные вых.
     out DDRD,R16        ; DDRD – регистр направления передачи  порта D
   ldi R16,0b00000100    ; подключение подтягивающих резисторов к выводу
D2
     out PortD,R16
m:    in R17,PinD          ; ввод с порта D
      out PortB,R17        ; вывод в порт B
rjmp m
```

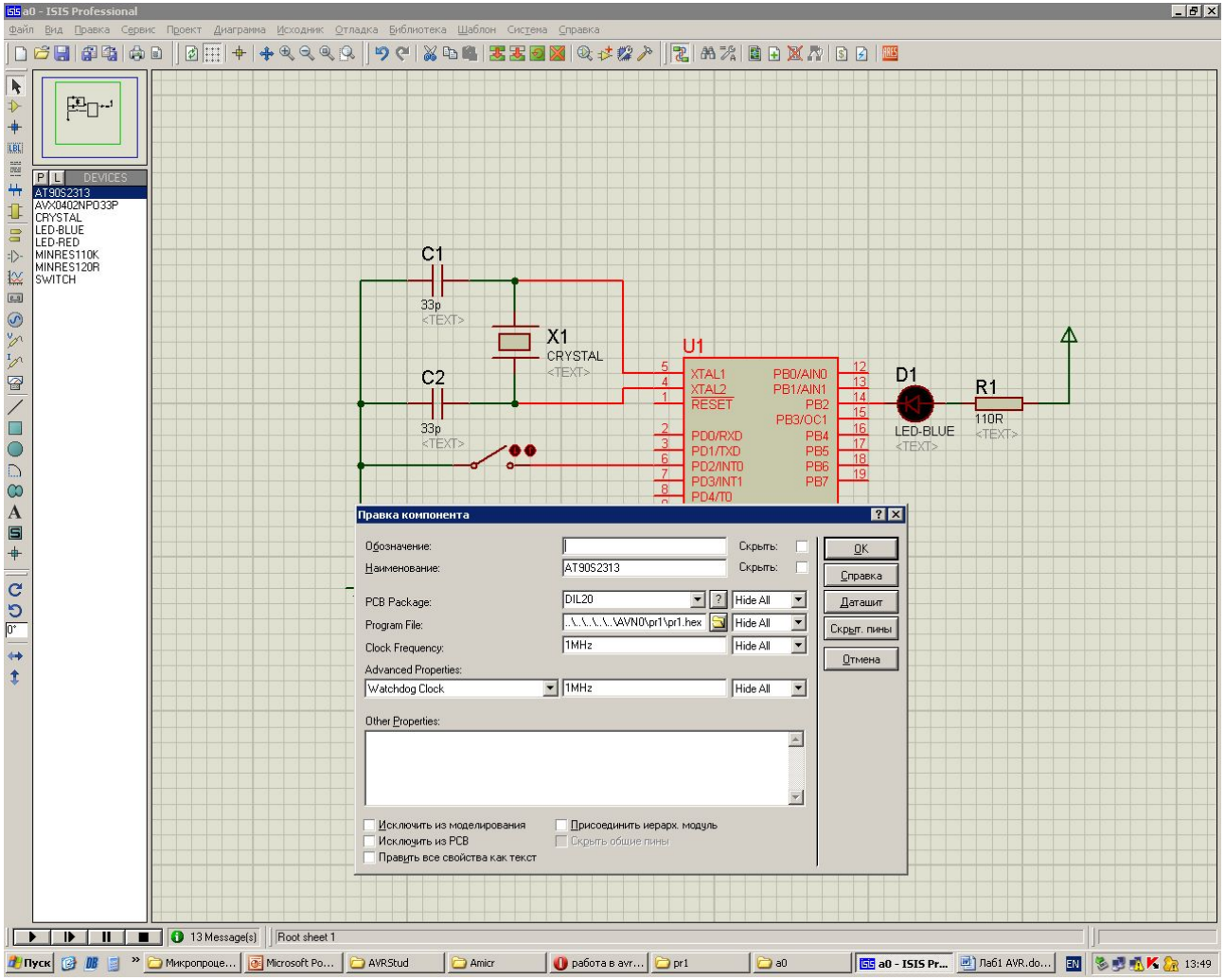



Рис.14. Загрузка hex – файла в микропроцессор

## Создание и компиляция программы в AVR-Studio

В меню нужно открыть Project – New Project.

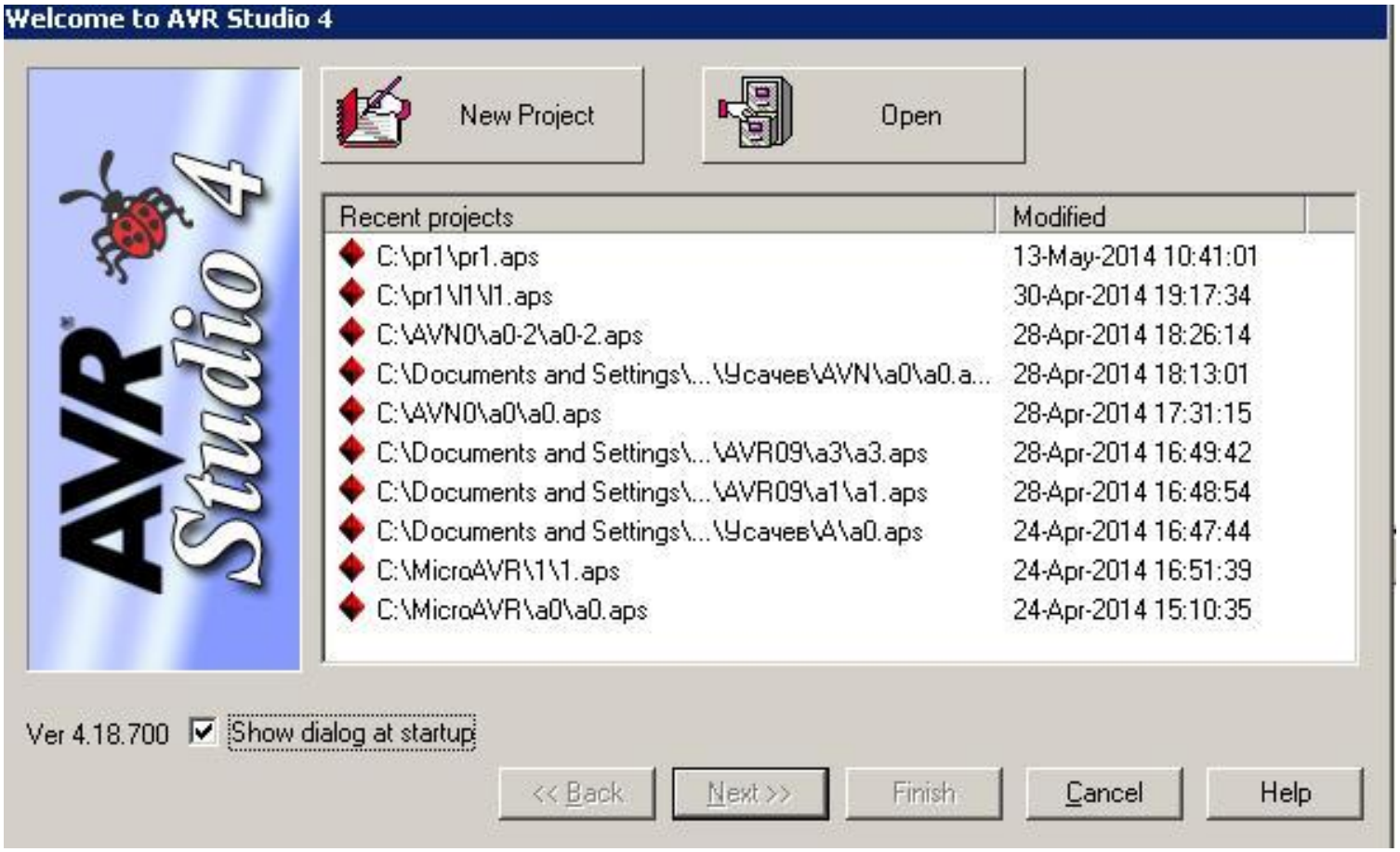



Рис.15. Окно

Выбираем язык программирования Atmel AVR Assembler и

задаем имя проекту и имя файлу с программой на ассемблере (с расширением .asm)

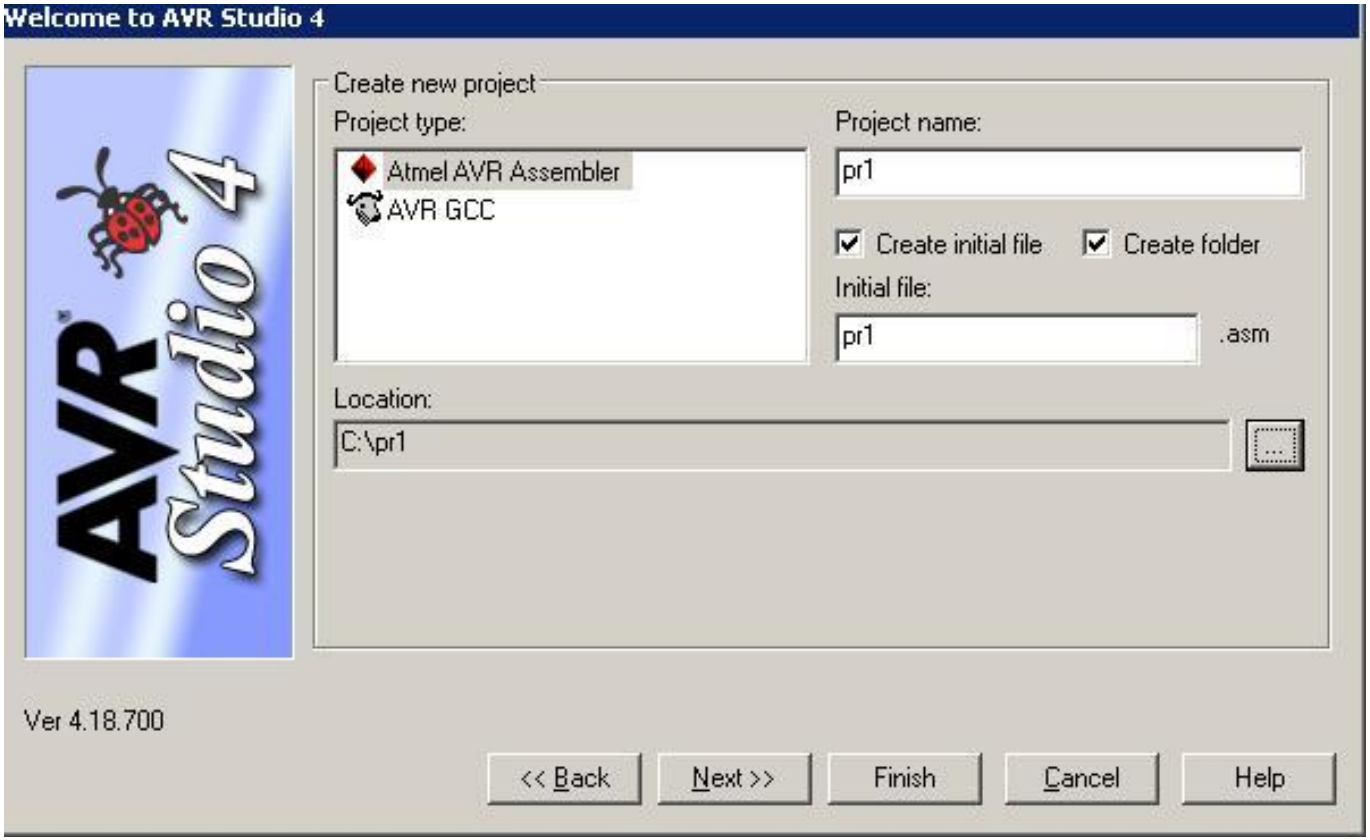



Рис.16.

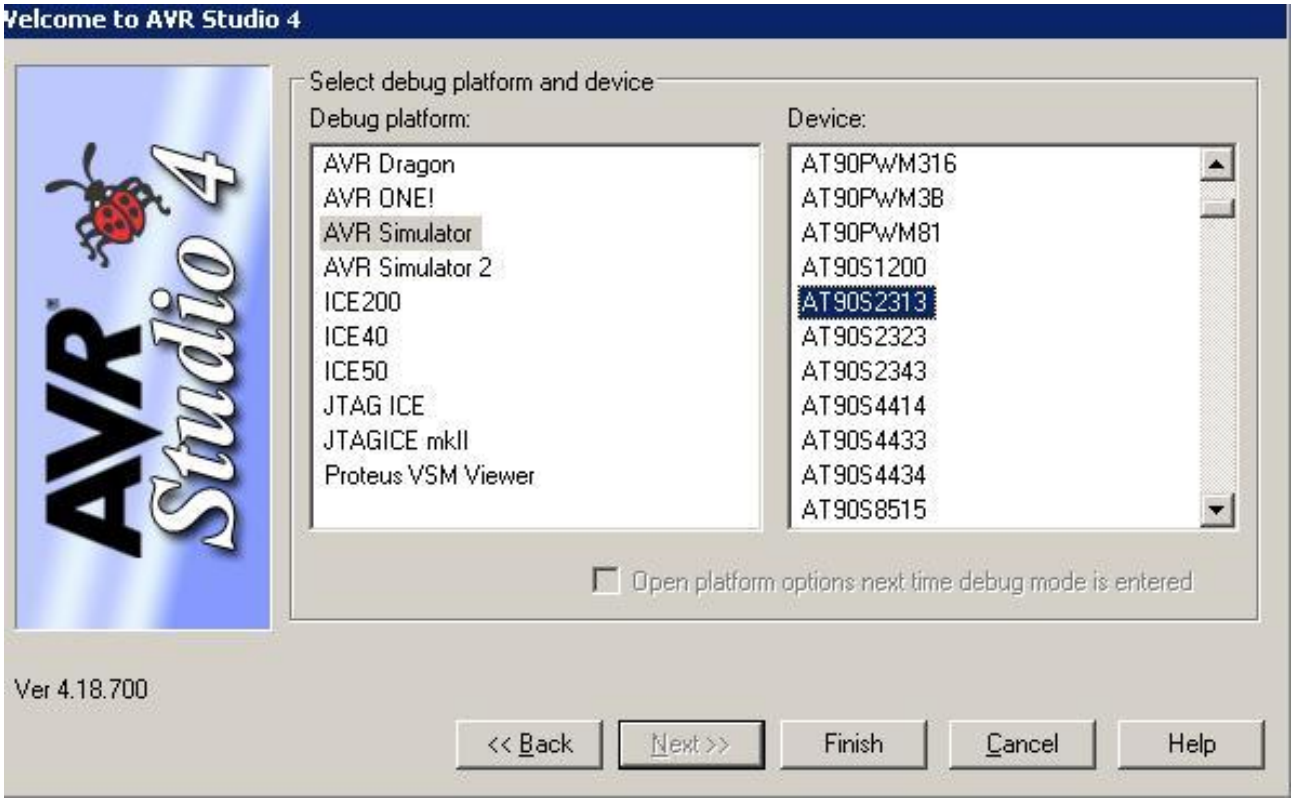



Рис.17. Выбираем  микропроцессор

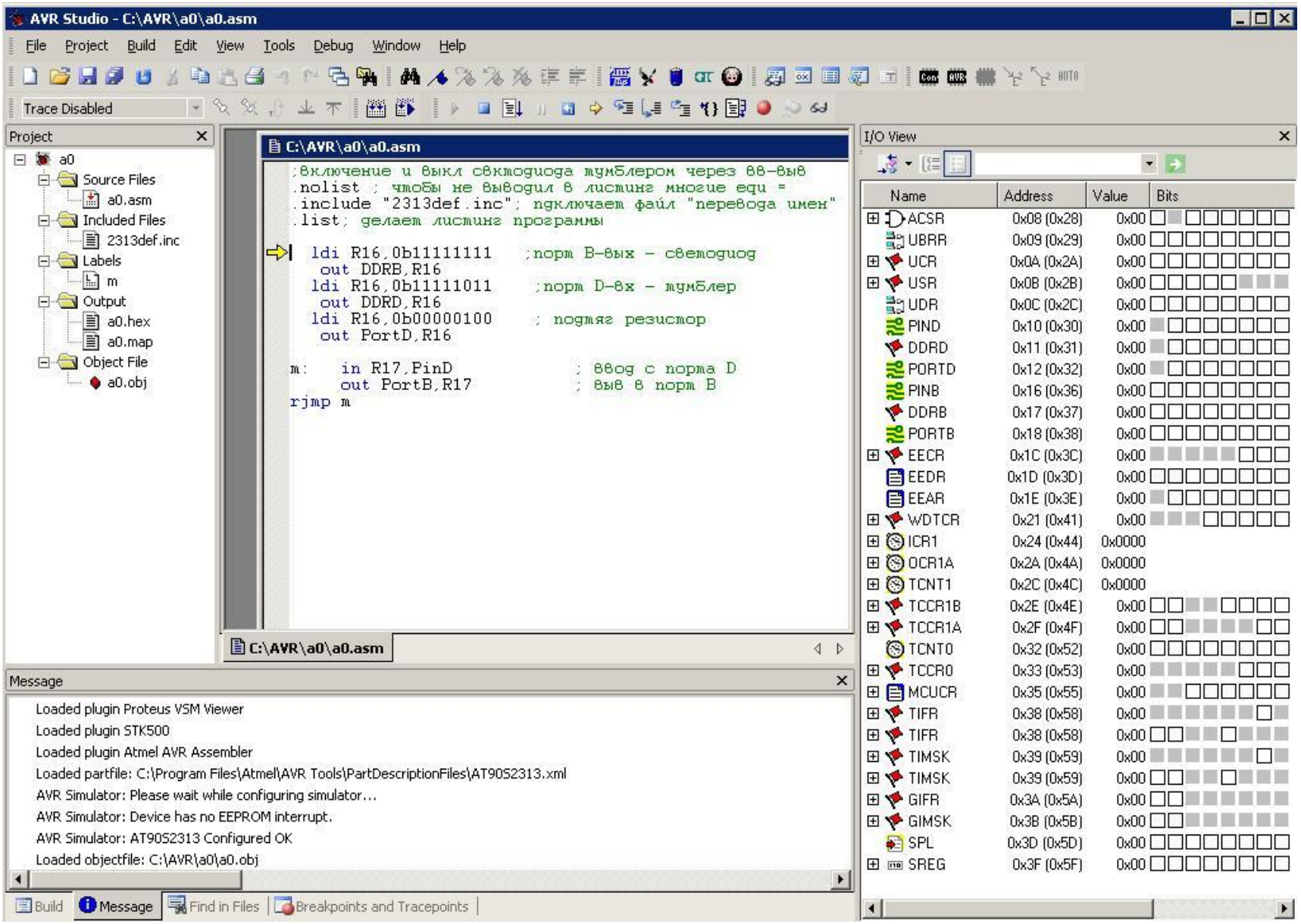



Рис. 18. Запись в AVR Studio программы .asm и ассемблирование - получение hex -файла

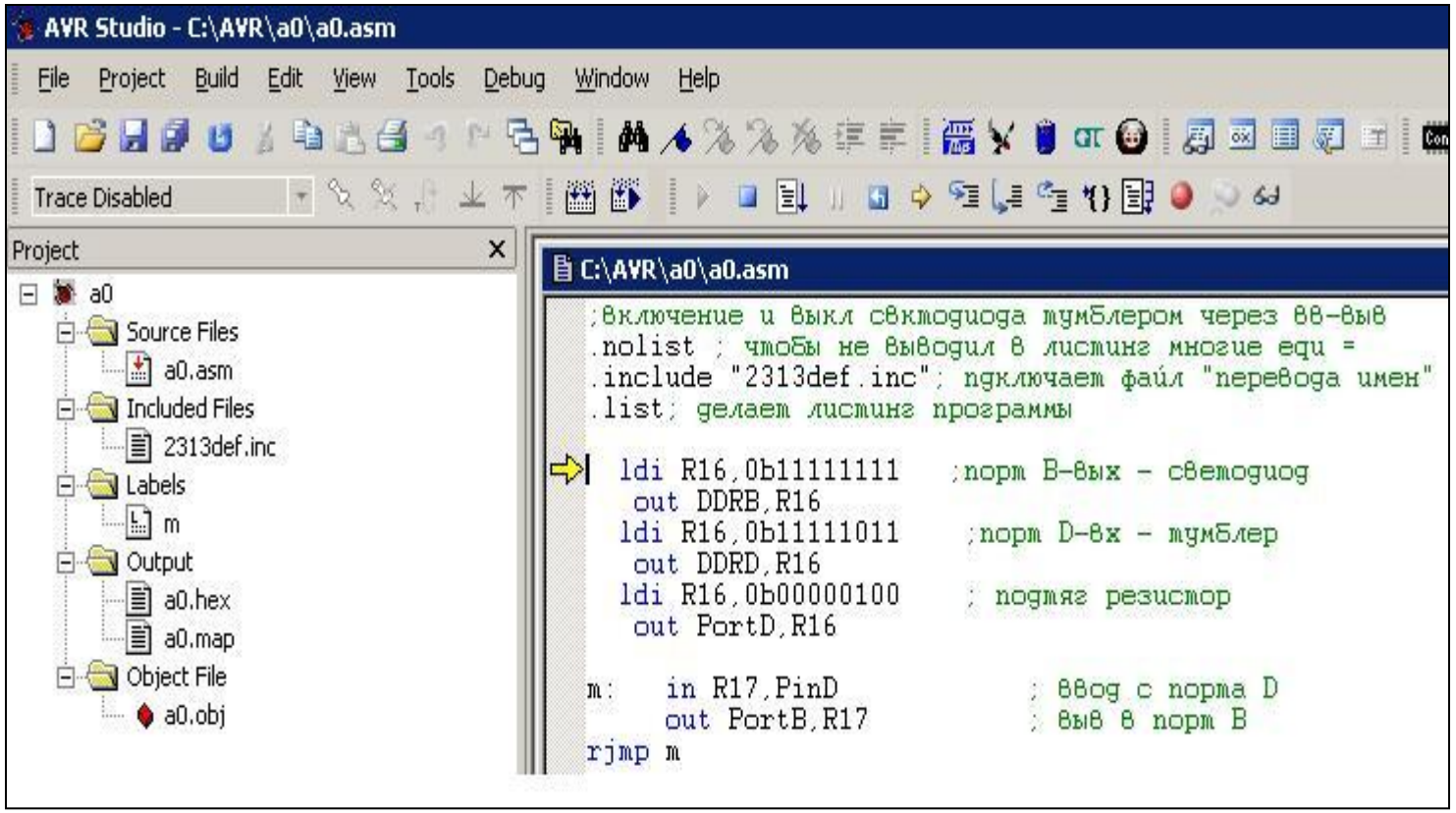


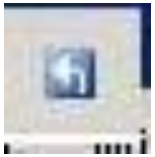

Рис.19. Компиляция программы и ее пошаговое выполнение

Панель инструментов AVR Studio

- Assemble (F7) ассемблирование программы

- Assemble and run (Ctrl+F7) ассемблирование программы и запуск

- Пошаговое выполнение программы



- Reset

Рис.20. Инструменты для компиляции программы и пошагового выполнения

# Моделирование устройств в программе Proteus

Запускаем “PROTEUS”. На рис.21. показано окно программы. Слева - панель Device, в которой выбираются элементы схемы , справа – окно , в котором собирается схема Для того, чтобы создать схему в программе Proteus нужно вначале выбрать микроконтроллер. Для этого нужно обратить внимание на панель  Device. Панель  Device находится слева и содержит две кнопки : “P” и “L”.

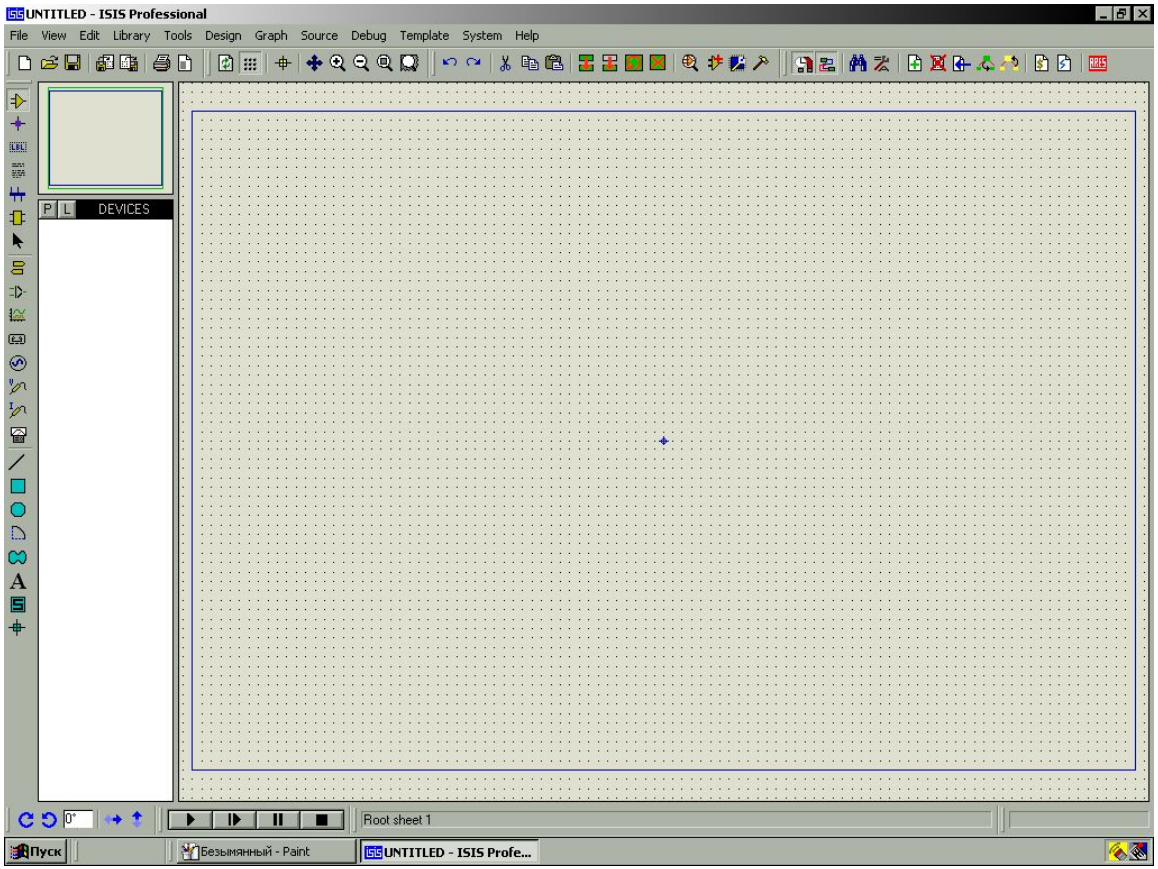



Рис.21. Окно  программы  Proteus

Чтобы выбрать нужный элемент нужно нажать кнопку “P”. При нажатии на кнопку “P”открывается окно Pick Device. В этом окне выбираем категорию ( те устройства, которые необходимы для создания схемы). Выбираем категорию “Microprocessorc Ics”. Появляется список микроконтроллеров , выбираем микроконтроллер AT90S2313.

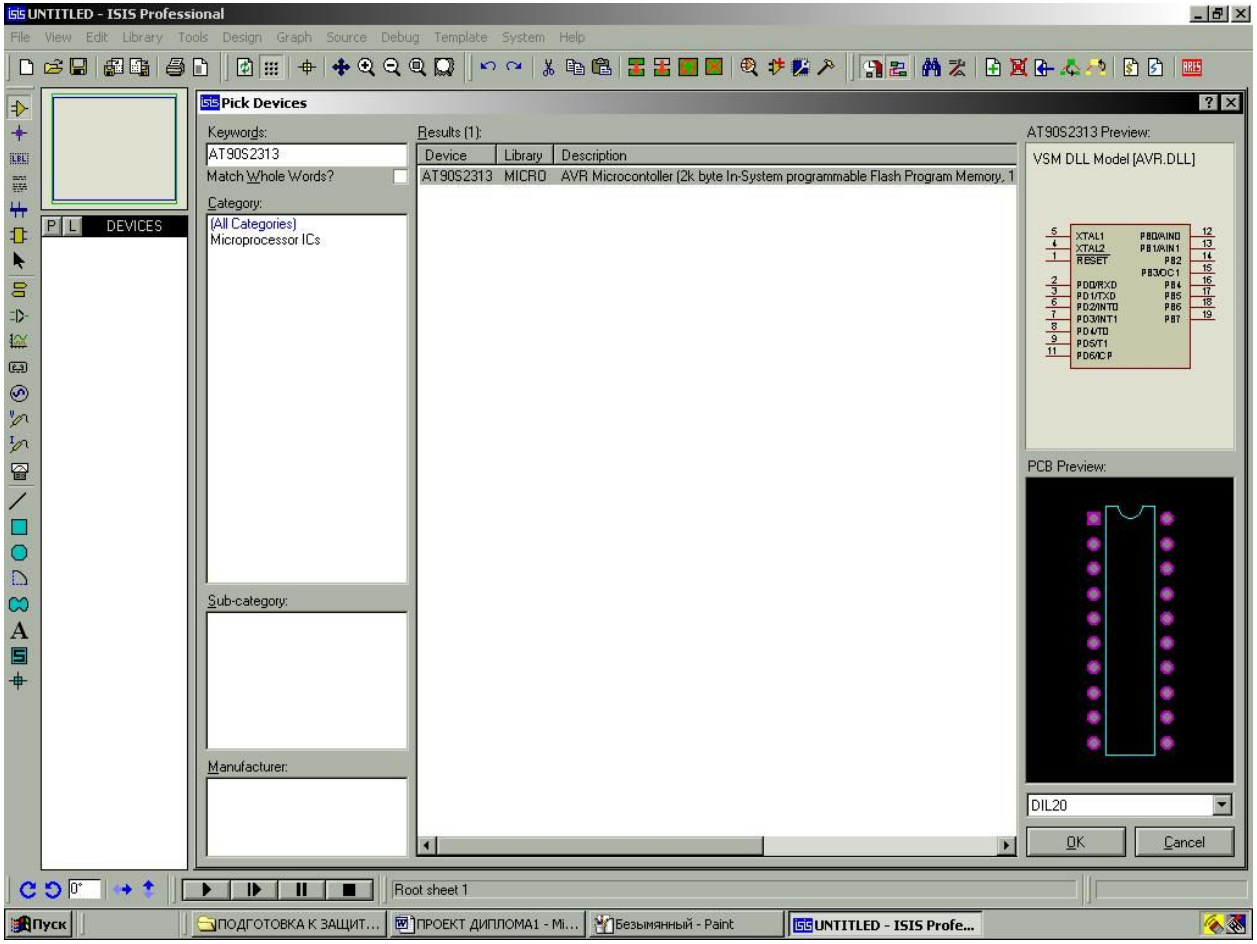



Рис.22. Поиск устройств.

Можно начать поиск устройства с помощью строки “Keyword”, которая находится также в окне Pick Device,написав например, “AT90S2313”. В окне «PCB Preview» появляется устройство, также показано расположение выводов этого устройства. Чтобы компонент оказался на схеме нажмем "ОК", затем указатель мыши поместим в нужное место на листе схемы и щелкнем левой кнопкой мыши , компонент окажется на схеме

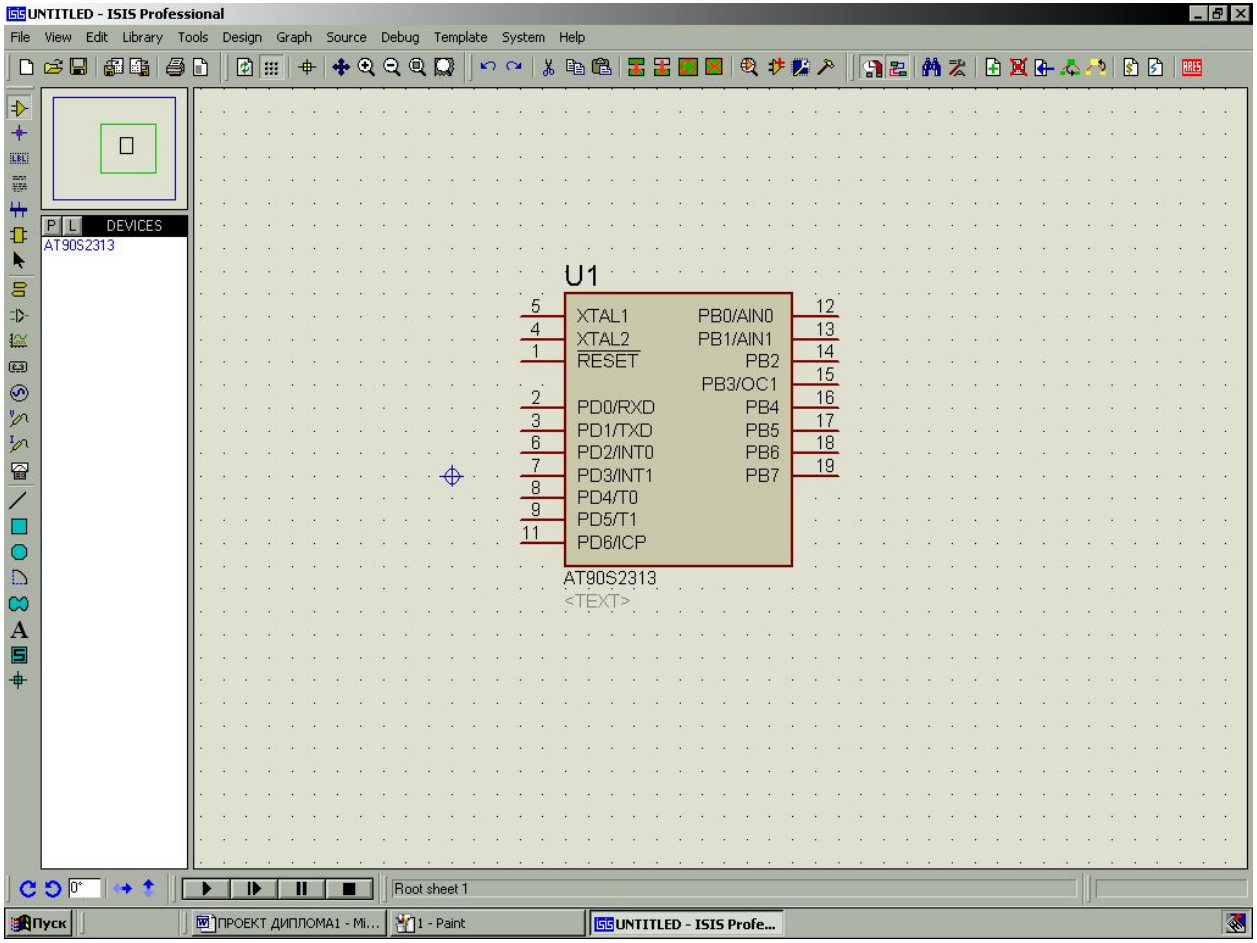



Рис.22. Выбор микроконтроллера

Выбираем резисторы . Для этого в окне Pick Device выбираем категорию “Resistors” или в строке “Keyword” пишем “110R” ( нам нужны резисторы на 110 Ом).

Выбираем конденсаторы. Для этого в окне Pick Device выбираем категорию “Capasitors” или в строке “Keyword” пишем “33Pf” ( нам нужны конденсаторы на 33 Пф).

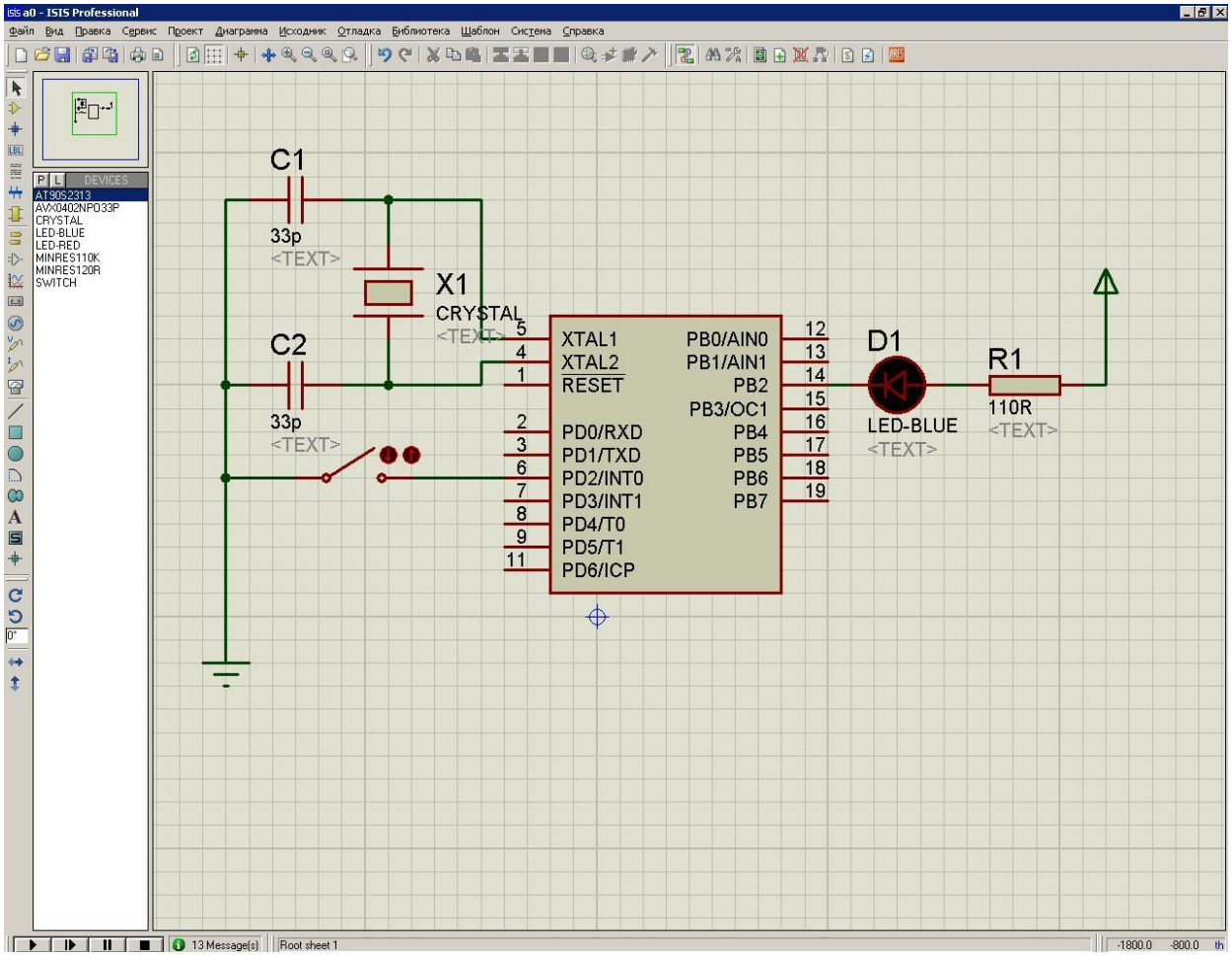



Рис.23. Монтаж схемы

Выбираем светодиод. Для этого в окне Pick Device выбираем категорию “Optoelectronics” или в строке “Keyword” пишем “Led” ” Выбираем цвет светодиодов (“Led red”).

Выбираем кварцевый резонатор. Для этого в окне Pick Device выбираем категорию
“Miscellaneous” или в строке “Keyword” пишем “Crystal device” ( нам нужен кварцевый резонатор).

Левой кнопкой мыши нажимаем на выводы устройства, тянем проводники к другим устройствам. Правой кнопкой можно сохранить положение проводника , до выбора следующего устройства. Правой кнопкой нажимаем на устройство на схеме, устройство выделяется красным .Нажимаем левую кнопку . Открывается окно «Edit component». В этом окне можно изменить параметры устройства,«Program file» - открывает папку , где находится файл прошивки микроконтроллера .hex- файл, где находится программный код.

Можно менять частоту тактирования микроконтроллера . В окне «Edit component» находим меню “Advanced Properties” и выбираем “Watchdog clock”.

"Hidden Pins" (скрытые выводы) - откроется дополнительное меню в котором показано как по умолчанию называются узлы (проводники) схемы к которым подключены питание МК - VCC и его общий провод - GND. Изменение этих названий может потребоваться вам при питании МК или других компонентов разными напряжениями или от разных источников.

# Запуск симулятора микроконтроллера

Для запуска симулятора имеются кнопки «Play, Pause, Stop», обозначенные соответствующими символами

Рис.24. Кнопки симулятора

“Пуск" - запуск симуляции или продолжение приостановленной симуляции.
"Шаг" - выполнить минимальный шаг по программе МК, обычно это одна инструкция на ассемблере. Этой кнопкой тоже можно начать симуляцию.
"Пауза" - пауза симуляции. Можно продолжить кнопками "Пуск" или "Шаг"
"Стоп" - остановка симуляции. После этого симуляция начнется сначала кнопками "Пуск" или "Шаг".
Чтобы симулировать в PROTEUS работу микроконтроллера достаточно
1) найти его в библиотеках и поместить на схему
2) указать какую программу он должен выполнять (как описано чуть выше)
3) указать частоту тактирования МК.

## Программирование микроконтроллера

Чтобы смоделировать схему нужно запрограммировать микроконтроллер. Микроконтроллер программируется на языках Си или Ассемблере. Программа для микроконтроллера пишется в специальных программах – компиляторах. В компиляторе есть текстовый редактор , где пишется текст программы .Далее программа проверяется на наличие ошибок ( компилируется). После компиляции создается файл с расширением “hex” , где находится программный код , необходимый для работы контроллера. Для программирования на Ассемблере используют AVR Studio , на языке Си - компилятор Code Vision AVR. Его можно использовать также при программировании на Ассемблере. Программирование микропроцессора в Proteus осуществляется просто – подключается hex –файл, как показано на рис.25.

После создания проекта в AVR Studio создается папка с именем этого проекта. В этой папке будут находиться программы .asm, .hex а также программа с расширением .aps, при запуске которой открывается в AVR Studio созданный проект.

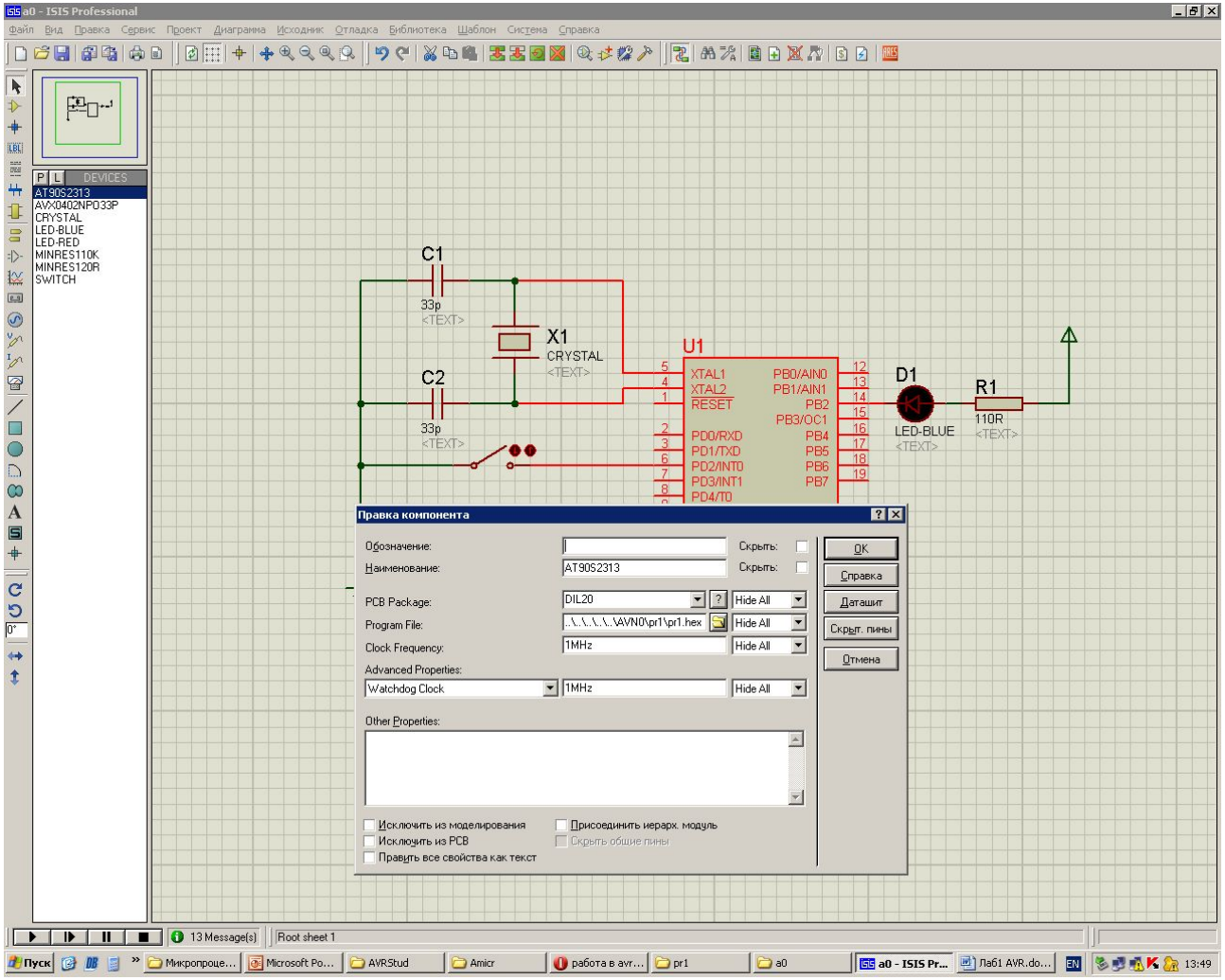



Рис.25. Загрузка hex – файла в микропроцессор

# ЛИТЕРАТУРА


1. Евстифеев А.В. Микроконтроллеры AVR семейства Mega : Рук. пользователя. – М. : Издательский дом «Додэка-XXI», 2007. – 592 с.

2. Предко М. Руководство по микроконтроллерам. Том 1. / Пер. с англ. под ред. И. И. Шагурина и С. Б. Лужанского – М.: Постмаркет, 2001. – 416 с.

3. Предко М. Руководство по микроконтроллерам. Том 2. / Пер. с англ. под ред. И. И. Шагурина и С. Б. Лужанского – М.: Постмаркет, 2001. – 488 с.

4. Джон Мортон. Микроконтроллеры AVR. Вводный курс./Пер. с англ. – М. : Издательский дом «Додэка-XXI», 2006. – 272 с. (Серия «Мировая электроника).

5. ГОЛУБЦОВ м.с., Кириченкова А.В..Микроконтроллеры AVR:от простого к сложному- М.: СОЛОН-Пресс,2004.-304с. (Серия «Библиотека инженера»).

6. http://chipmk.ru/index.php?option=com_content&view=article&id=230:-c-avrstudio-5-6&catid=58:-avr-asm&Itemid=69 – программирование AVR.

7. http://proteus123.narod.ru/01.htm - PROTEUS.

8. http://www.gaw.ru/ - информация о МК фирмы ATMEL.