ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ
РЕСПУБЛИКИ УЗБЕКИСТАН

САМАРКАНДСКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ И ПЕДАГОГИЧЕСКИХ ТЕХНОЛОГИЙ

КАФЕДРА ОБЩЕПРОФЕССИОНАЛЬНЫХ ДИСЦИПЛИН

ВЫПУСКНАЯ

КВАЛИФИКАЦИОННАЯ РАБОТА

для получения академической степени бакалавриата по направлению 5521900 - “Информатика и информационные технологии”

***ТЕМА*: Разработка алгоритма и программы расчета оценки ситуационной игры на примере игры в шахматы**

Работа рекомендована к
защите заседанием кафедры
№ ___ от ___ мая 2013 года
Заведующий кафедрой

_____доц. Рахимов Н.О.
“_____” _____ 2013 г

Выполнил: студент 4-курса

_____Нахалов Зариф

Научный руководитель:

_____доц. Абдукаримов А.

САМАРКАНД – 2013

СОДЕРЖАНИЕ

Введение

Актуальность темы: Всемирный финансово-экономический кризис, начавшийся в конце 2008 года и приобретающий сегодня большие масштабы и глубину в оценках многих международных экспертов и специалистов, получает больше вопросов, чем ответов о причинах и прогнозах его дальнейшего развития.

С учетом растущей интегрированности нашей экономики в мировое хозяйство и ее финансово-экономические связи показали, что мировой кризис, прежде всего его последствия, оказал и продолжает оказывать на Узбекистан негативное воздействие. В Узбекистане созданы достаточный запас прочности и необходимая ресурсная база для того, чтобы обеспечить устойчивую и бесперебойную работу нашей финансово-экономической, бюджетной, банковско-кредитной системы, а также предприятий и отраслей реальной экономики. В связи с чем, большой интерес проявляется на развитие ИТ-индустрии в Узбекистане. Все больше товаров и услуг теперь можно найти в Интернете.

Основные задачи исследования. В настоящее время в системах управления и обработки данных все чаще применяются микроконтроллеры, решающие широкий спектр задач. Однокристальные микроконтроллеры (ОМК) являются наиболее массовым видом устройств современной микропроцессорной техники, годовой объем выпуска, которых составляет более 2,5 млрд. штук. Интегрируя на одном кристалле высокопроизводительный процессор, память и набор периферийных схем, однокристальные микроконтроллеры позволяют с минимальными затратами реализовать высокоэффективные системы и устройства управления различными объектами (процессами). В отличие от обычных микропроцессоров, для работы которых необходимы внешние интерфейсные схемы, в корпусе однокристальных микроконтроллеров наряду с основными функциональными узлами размещены такие вспомогательные узлы, как тактовый генератор, таймер, контроллер прерываний, цифро-аналоговый и

аналого-цифровой преобразователи, порты ввода-вывода.

Благодаря этим качествам однокристальные микроконтроллеры находят широкое применение в системах промышленной автоматики, контрольно-измерительных приборах и системах, аппаратуре связи, автомобильной электронике, медицинском оборудовании, бытовой технике и многих других областях.

Применение однокристальных микроконтроллеров позволяет перенести основные затраты, связанные с разработкой встраиваемых систем управления, из аппаратной в программную область. Это неминуемо влечет за собой увеличение сложности программного обеспечения микро-контроллеров.

Особенностью разработки программного обеспечения для однокристальных микроконтроллеров является использование языка низкого уровня - языка Си. Это связано с тем, что при реализации встраиваемых систем критичными являются время реакции на внешние воздействия, время выполнения заданных процедур обработки данных, размер программного кода и области данных.

Мировая промышленность выпускает огромную номенклатуру микроконтроллеров. По области применения их можно разделить на два класса: специализированные, предназначенные для применения в какой-либо одной конкретной области (контроллер для телевизора, контроллер для модема, контроллер для компьютерной мышки) и универсальные, которые не имеют конкретной специализации и могут применяться в самых различных областях микроэлектроники, с помощью которых можно создать как любое из перечисленных выше устройств, так и принципиально новое устройство.

Разработка программ для создания автоматизированных систем управления температурными режимами помещений – весьма достойная сфера приложения усилий. Развитие как технических, так и программных средств на современном этапе обеспечивает возможности создания терморегуляторов с такими особенностями, как простота схем при

существенно более широких, чем у распространенных аналоговых, функциональных возможностях, отсутствие необходимости регулировки и настройки при изготовлении и эксплуатации.

Но наиболее существенным достоинством таких регуляторов является их исключительно простая модификация, - на основе практически одинаковых схемных и конструктивных решениях, могут быть построены регуляторы для самых различных применений, что резко упрощает их разработку, а, следовательно, и стоимость. Требуется лишь изменение программного обеспечения и, возможно, исполнительных узлов.

Среднетемпературные терморегуляторы предназначены для автоматического измерения и поддержания стабильной температуры, например, в термостатах, инкубаторах, теплицах и т.п.

Регуляторы температуры, или, как их еще называют, терморегуляторы, предназначены для поддержания заданной температуры жидкости (например, фото раствора, воды в аквариуме, воды в системе электрического водяного отопления), воздуха в теплице, в жилом помещении и пр. Принцип работы любого терморегулятора состоит в плавном или скачкообразном изменении мощности нагревательного элемента в соответствии с температурой датчика.

Существуют терморегуляторы со скачкообразным изменением мощности, при нагрузке которых нагревательный элемент отключается, как только температура датчика достигает определенного значения, и выключается при понижении температуры до ее заданного значения. Нагревательный элемент при этом находится в одном из двух состояний: включен или выключен, поэтому регулятор с таким законом управления часто называют релейным.

В настоящее время более тридцати зарубежных фирм выпускают микроконтроллеры массового применения с разрядностью 8 бит, недорогие и пригодные для использования в самых разнообразных приложениях. Однако именно микроконтроллеры серии AVR фирмы Atmel переживают последние

три-четыре года в развитых странах поистине взрывной рост популярности. Эти микроконтроллеры также крайне популярны во всем мире, как у производителей электронной техники, так и среди радиолюбителей.

В чем же причина такой популярности? Конечно, не последнюю роль сыграли правильная маркетинговая политика, мощная и продуманная поддержка разработчиков со стороны фирмы и низкая стоимость микросхем. Кроме этого, сам продукт обладает целым рядом неоспоримых достоинств. Микроконтроллеры AVR фирмы Atmel объединили в себе все передовые технологии, применяемые в производстве микроконтроллеров: развитую RISC-архитектуру, минимальное энергопотребление при высоком быстродействии, ППЗУ, программируемое пользователем, функциональную законченность.

Четкая и продуманная внутренняя структура контроллеров и небольшая, но мощная система команд с интуитивно понятной мнемоникой значительно облегчают процесс изучения контроллеров AVR и написание для них программ.

Цель работы. Разработка прибора, предназначенного для автоматического регулирования температуры. Главная особенностью терморегуляторов - простота схем при существенно более широких, чем у распространенных аналоговых, функциональных возможностях, отсутствие необходимости регулировки и настройки при изготовлении и эксплуатации.

В выпускной квалификационной работе рассмотрены общие подходы к реализации микропроцессорной системы на базе микроконтроллера для терморегулятора помещений. В процессе написания квалификационной работы автором велась разработка реализации микропроцессорной системы на базе микроконтроллера U1 типа Atmega8.

Данная выпускная квалификационная работа состоит из 77 страниц, включающих в себя введение, три глав, заключение и приложения 1-2, включающего показ запрограммированного микроконтроллера в ситеме Electronics Labcenter (Proteus 7.6) предназначенный для проектирования

многослойных печатных плат (ПП) аналоговых, цифровых и аналого-цифровых устройств.

Глава 1. Теоретические основы разработки микропроцессорной системы. В этой главе объясняются основные понятия и термины, используемые в данной дипломной работе. Что собой вообще представляет микроконтроллеры, их назначение, область применения и их архитектура.

Глава 2. Разработка программы автоматизации расчета температурного режима помещений. Приводится постановка задачи, разработки структурной схемы устройства, функциональной спецификации и алгоритм управления.

Глава 3. Описание программного обеспечения и порядок его использования. В этой части для показа работы программного обеспечения микроконтроллера используется система Electronics Labcenter (Proteus 7.6) предназначенный для проектирования многослойных печатных плат (ПП) аналоговых, цифровых и аналого-цифровых устройств. Приводится порядок использования программного обеспечения.

В приложениях дано короткое описание выпускной работы и листинг.

Глава 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

1.1. Микроконтроллеры, назначение и область применения, их архитектура

Микропроцессорные системы на основе микроконтроллера используются чаще всего в качестве встроенных систем для решения задач управления некоторым объектом. Важной особенностью данного применения является работа в реальном времени, т.е. обеспечение реакции на внешние события в течение определенного временного интервала. Такие устройства получили название контроллеров.

Можно считать что микроконтроллер - это компьютер, разместившийся в одной микросхеме. Отсюда и его основные привлекательные качества: малые габариты; высокие производительность, надежность и способность быть адаптированным для выполнения самых различных задач.

Микроконтроллер помимо центрального процессора содержит память и многочисленные устройства ввода/вывода: аналого-цифровые преобразователи, последовательные и параллельные каналы передачи информации, таймеры реального времени, широтно-импульсные модуляторы, генераторы программируемых импульсов и т.д. Его основное назначение - использование в системах автоматического управления, встроенных в самые различные устройства: кредитные карточки, фотоаппараты, сотовые телефоны, музыкальные центры, телевизоры, видеомагнитофоны и видеокамеры, стиральные машины, микроволновые печи, системы охранной сигнализации, системы зажигания бензиновых двигателей, электроприводы локомотивов, ядерные реакторы и многое, многое другое. Встраиваемые системы управления стали настолько массовым явлением, что фактически сформировалась новая отрасль экономики, получившая название Embedded Systems (встраиваемые системы).

Достаточно широкое распространение имеют микроконтроллеры фирмы ATMEL, которые располагают большими функциональными возможностями.

Применение микроконтроллера можно разделить на два этапа: первый - программирование, когда пользователь разрабатывает программу и прошивает ее непосредственно в кристалл, и второй - согласование спроектированных исполнительных устройств с за программируемым микроконтроллером. Значительно облегчают отладку программы на первом этапе - симулятор, который наглядно моделирует работу микропроцессора. На втором этапе для отладки используется внутрисхемный эмулятор, который является сложным и дорогим устройством, зачастую недоступным рядовому пользователю.

В тоже время в литературе мало уделено внимания вопросам обучения программированию некоторых недорогих микроконтроллерах, в сочетании с реальными исполнительными устройствами.

Разработка макета программатора отличающегося простотой, наглядностью и низкой себестоимостью, становиться необходимой как для самого программирования кристаллов, так и для наглядного обучения широкого круга пользователей основам программирования микроконтроллера.

Микроконтроллер - компьютер на одной микросхеме. Предназначен для управления различными электронными устройствами и осуществления взаимодействия между ними в соответствии с заложенной в микроконтроллер программой. В отличие от микропроцессоров, используемых в персональных компьютерах, микроконтроллеры содержат встроенные дополнительные устройства. Эти устройства выполняют свои задачи под управлением микропроцессорного ядра микроконтроллера.

К наиболее распространенным встроенным устройствам относятся устройства памяти и порты ввода/вывода (I/O), интерфейсы связи, таймеры, системные часы. Устройства памяти включают оперативную память (RAM),

постоянные запоминающие устройства (ROM), перепрограммируемую ROM (EPROM), электрически перепрограммируемую ROM (EEPROM). Таймеры включают и часы реального времени, и таймеры прерываний. Средства I/O включают последовательные порты связи, параллельные порты (I/O линии), аналого-цифровые преобразователи (A/D), цифроаналоговые преобразователи (D/A), драйверы жидкокристаллического дисплея (LCD) или драйверы вакуумного флуоресцентного дисплея (VFD). Встроенные устройства обладают повышенной надежностью, поскольку они не требуют никаких внешних электрических цепей.

В отличие от микроконтроллера контроллером обычно называют плату, построенную на основе микроконтроллера, но достаточно часто при использовании понятия "микроконтроллер" применяют сокращенное название этого устройства, отбрасывая приставку "микро" для простоты. Также при упоминании микроконтроллеров можно встретить слова "чип" или "микрочип", "кристалл" (большинство микроконтроллеров изготавливают на едином кристалле кремния), сокращения МК или от английского microcontroller - MC.

Микроконтроллеры можно встретить в огромном количестве современных промышленных и бытовых приборов: станках, автомобилях, телефонах, телевизорах, холодильниках, стиральных машинах. и даже кофеварках. Среди производителей микроконтроллеров можно назвать Intel, Motorola, Hitachi, Microchip, Atmel, Philips, Texas Instruments, Infineon Technologies (бывшая Siemens Semiconductor Group) и многих других. Для производства современных микросхем требуются сверхчистые помещения.

Основным классификационным признаком микроконтроллеров является разрядность данных, обрабатываемых арифметико-логическим устройством. По этому признаку они делятся на 4-, 8-, 16-, 32 - и 64-разрядные. Сегодня наибольшая доля мирового рынка микроконтроллеров принадлежит восьмиразрядным устройствам (около 50% в стоимостном выражении). За ними следуют 16-разрядные и DSP-микроконтроллеры (DSP -

Digital Signal Processor - цифровой сигнальный процессор), ориентированные на использование в системах обработки сигналов (каждая из групп занимает примерно по 20% рынка). Внутри каждой группы микроконтроллеры делятся на CISC - и RISC-устройства. Наиболее многочисленной группой являются CISC-микроконтроллеры, но в последние годы среди новых чипов наметилась явная тенденция роста доли RISC-архитектуры.

Тактовая частота, или, более точно, скорость шины, определяет, сколько вычислений может быть выполнено за единицу времени. В основном производительность микроконтроллера и потребляемая им мощность увеличиваются с повышением тактовой частоты. Производительность микроконтроллера измеряют в MIPS (Million Instructions per Second - миллион инструкций в секунду).

Термин контроллер образовался от английского слова to control - управлять. Эти устройства могут основываться на различных принципах работы от механических или оптических устройств до электронных аналоговых или цифровых устройств. Механические устройства управления обладают низкой надежностью и высокой стоимостью по сравнению с электронными блоками управления, поэтому в дальнейшем мы такие устройства рассматривать не будем. Электронные аналоговые устройства требуют постоянной регулировки в процессе эксплуатации, что увеличивает стоимость их эксплуатации. Поэтому такие устройства к настоящему времени почти не используются. Наиболее распространенными на сегодняшний день схемами управления являются схемы, построенные на основе цифровых микросхем.

В зависимости от стоимости и габаритов устройства, которым требуется управлять, определяются и требования к контроллеру. Если объект управления занимает десятки метров по площади, как, например, автоматические телефонные станции, базовые станции сотовых систем связи или радиорелейные линии связи, то в качестве контроллеров можно использовать универсальные компьютеры. Управление при этом можно

осуществлять через встроенные порты компьютера (LPT, COM, USB или Ethernet). В такие компьютеры при включении питания заносится управляющая программа, которая и превращает универсальный компьютер в контроллер.

Использование универсального компьютера в качестве контроллера позволяет в кратчайшие сроки производить разработку новых систем связи, легко их модернизировать (путём простой смены программы) а также использовать готовые массовые (а значит дешёвые) блоки.

Если же к контроллеру предъявляются особенные требования, такие, как работа в условиях тряски, расширенном диапазоне температур, воздействия агрессивных сред, то приходится использовать промышленные варианты универсальных компьютеров. Естественно, что эти компьютеры значительно дороже обычных универсальных компьютеров, но всё равно они позволяют экономить время разработки системы, за счёт того, что не нужно вести разработку аппаратуры контроллера.

Контроллеры требуются не только для больших систем, но и для малогабаритных устройств, таких как радиоприёмники, радиостанции, магнитофоны или сотовые аппараты. В таких устройствах к контроллерам предъявляются жёсткие требования по стоимости, габаритам и температурному диапазону работы. Этим требованиям не могут удовлетворить даже промышленные варианты универсального компьютера. Приходится вести разработку контроллеров на основе однокристальных ЭВМ, которые в свою очередь получили название микроконтроллеры. Любые устройства, в том числе и устройства связи, радиоавтоматики или аудиовизуальной аппаратуры требуют присутствия в своем составе устройства управления (контроллера). Контроллеры требуются практически во всех предметах и устройствах, которые окружают нас. Наиболее распространёнными в настоящее время являются микроконтроллеры семейства MCS-51. Это семейство поддерживается рядом фирм - производителей микросхем. Не менее распространёнными в мире являются

микроконтроллеры фирмы Motorola. Это такие семейства как HC05, HC07, HC11 и многие другие. Пожалуй, не менее популярными микроконтроллерами являются микроконтроллеры семейства AVR фирмы Atmel. Если представить все типы современных микроконтроллеров, то можно поразиться огромным количеством разнообразных приборов этого класса, доступных потребителю. Однако все эти приоры можно разделить на следующие основные типы: встраиваемые (embedded) 8-разрядные микроконтроллеры; 16 - и 32-разрядные микроконтроллеры; цифровые сигнальные процессоры. Промышленностью выпускаются очень широкая номенклатура встраиваемых микроконтроллеров. В них все необходимые ресурсы (память, устройства ввода-вывода и т.д.) располагаются на одном кристалле с процессорным ядром [3]. Если подать питание и тактовые импульсы на соответствующие входы микроконтроллера, то можно сказать, что он как бы "оживет" и с ним можно будет работать. Обычно микроконтроллеры содержат значительное число вспомогательных устройств, благодаря чему обеспечивается их включение в реальную систему с использованием минимального количества дополнительных компонентов. В состав этих микроконтроллеров входят:

- Схема начального запуска процессора (SET);
- Генератор тактовых импульсов;
- Центральный процессор;
- Память программ (EPROM) и программный интерфейс;
- Средства ввода/вывода данных;
- Таймеры, фиксирующие число командных циклов.

Общая структура микроконтроллера показана на (Рис.1) Эта структура дает представление о том, как микроконтроллер связывается с внешним миром. Более сложные встраиваемые микроконтроллеры могут дополнительно реализовывать следующие возможности:

- Встроенный монитор/отладчик программ;
- Внутренние средства программирования памяти программ (ROM);

- Обработка прерываний от различных источников;
- Аналоговый ввод/вывод;
- Последовательный ввод/вывод (синхронный и асинхронный);
- Параллельный ввод/вывод (включая интерфейс с компьютером);
- Подключение внешней памяти (микропроцессорный режим).

Все эти возможности значительно увеличивают гибкость применения микроконтроллера и делают более простым процесс разработки систем на и основе.

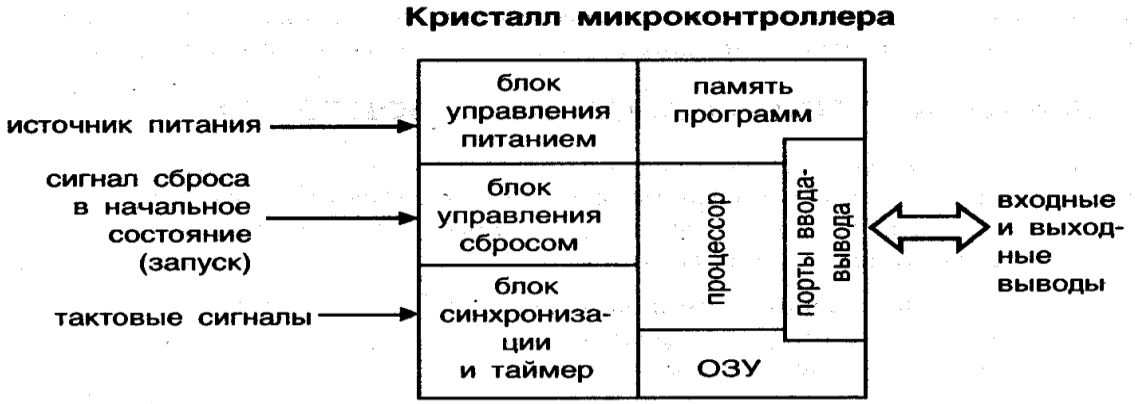


Рис.1. *Структура микроконтроллера.*

Некоторые микроконтроллеры (особенно 16- и 32-разрядные) используют только внешнюю память, которая включает в себя как память программ (ROM), так и некоторый объем памяти данных (RAM), требуемый для данного применения. Они применяются в системах, где требуется большой объем памяти и относительное не большое количество устройств (портов) ввода/вывода. Типичным примером применения такого микроконтроллера с внешней памятью является котроллер жесткого диска (HDD) с буферной кэш-памятью, который обеспечивает промежуточное хранение и распределение больших объемов данных (порядка нескольких мегабайт). Внешняя память дает возможность такому микроконтроллеру работать с более высокой скоростью, чем встраиваемый микроконтроллер.

Цифровые сигнальные процессоры (DSP) - относительно новая категория процессоров. Назначение DSP состоит в том, чтобы получать текущие данные от аналоговой системы, обрабатывать данные и формировать соответствующий отклик в реальном масштабе времени. Они обычно входят в состав систем, используясь в качестве устройств управления внешним оборудованием, и не предназначены для автономного применения.

Система команд

В зависимости от числа используемых кодов операций системы команд микроконтроллеров можно разделить на две группы: CISC и RISC. Термин CISC означает сложную систему команд и является аббревиатурой английского определения Complex Instruction Set Computer. Аналогично термин RISC означает сокращенную систему команд и происходит от английского Reduce Instruction Set Computer. Систему команд микроконтроллера 8051 можно отнести к типу CISC. Однако, не смотря на широкую распространенность этих понятий, необходимо признать, что сами названия не отражают главного различия между системами команд CISC и RISC. Основная идея RISC архитектуры - это тщательный подбор таких комбинаций кодов операций, которые можно было бы выполнить за один такт тактового генератора. Основной выигрыш от такого подхода - резкое упрощение аппаратной реализации ЦП и возможность значительно повысить его производительность.

Очевидно, что в общем случае одной команде CISC соответствует несколько команд RISC. Однако обычно выигрыш от повышения быстродействия в рамках RISC перекрывает потери от менее эффективной системы команд, что приводит к более высокой эффективности RISC систем в целом по сравнению с CISC.

Однако в настоящее время грань между CISC и RISC архитектурой стремительно стирается. Например, микроконтроллер семейства AVR фирмы Atmel имеют систему команд из 120 инструкций, что соответствует типу

CISC. Однако большинство из них выполняется за один такт, что является признаком RISC архитектуры. Сегодня принято считать, что признаком RISC архитектуры является выполнение команд за один такт тактового генератора. Число команд само по себе значения уже не имеет.

Типы памяти микроконтроллеров.

Можно выделить три основных вида памяти, используемой в микроконтроллерах:

а) *память программ*;

б) *память данных*;

в) *регистры микроконтроллера*.

Память программ представляет собой постоянную память, предназначенную для хранения программного кода и констант. Эта память не изменяет содержимого в процессе выполнения программы. *Память данных* предназначена для хранения переменных в ходе выполнения программы. *Регистры микроконтроллера* - этот вид памяти включает внутренние регистры процессора и регистры, которые служат для управления периферийными устройствами.

Для хранения программ обычно служит один из видов постоянной памяти: ROM (масочные ПЗУ), PROM (однократно программируемые ПЗУ), EPROM (электрически программируемые ПЗУ с ультрафиолетовым стиранием) или EEPROM (ПЗУ с электрической записью и стиранием, к этому виду также относятся современные микросхемы Flash-памяти). Все эти виды памяти являются энергонезависимыми - это означает, что содержимое памяти сохраняется после выключения питания микроконтроллера.

Многократно программируемые ПЗУ - EPROM и EEPROM (Electrically Erasable Programmable Memory) подразделяются на ПЗУ со стиранием ультрафиолетовым облучением (выпускаются в корпусах с окном), и микроконтроллер с электрически перепрограммируемой памятью, соответственно.

В настоящее время протоколы программирования современной EEPROM памяти позволяют выполнять программирование микроконтроллера непосредственно в составе системы, где он работает. Такой способ программирования получил название - ISP (In System Programming). И теперь можно периодически обновлять программное обеспечение микроконтроллера без удаления из платы. Это дает огромный выигрыш на начальных этапах разработки систем на базе микроконтроллера или в процессе их изучения, когда масса времени уходит на многократный поиск причин неработоспособности системы и выполнение последующих циклов стирания-программирования памяти программ.

Функционально Flash-память мало отличается от EEPROM. Основное различие состоит в способности стирания записанной информации. В памяти EEPROM стирание производится отдельно для каждой ячейки, а во Flash-памяти стирание осуществляется целыми блоками.

ОЗУ (RAM) - оперативное запоминающее устройство, используется для хранения данных. Эту память называют еще памятью данных. Число циклов чтения и записи в ОЗУ неограниченно, но при отключение питания вся информация теряется.

1.2. Среды программирования. Схемы подключения микроконтроллера

Программная среда "AVR Studio" - это мощный современный программный продукт, позволяющий производить все этапы разработки программ для любых микроконтроллеров серии AVR. Пакет включает в себя специализированный текстовый редактор для написания программ, мощный программный отладчик.

Кроме того, "AVR Studio" позволяет управлять целым рядом подключаемых к компьютеру внешних устройств, позволяющих выполнять

аппаратную отладку, а также программирование ("прошивку") микросхем AVR.

Программная среда "AVR Studio" работает не просто с программами, а с проектами. Для каждого проекта должен быть отведен свой отдельный каталог на жестком диске. В AVR Studio одновременно может быть загружен только один проект.

При загрузке нового проекта предыдущий проект автоматически выгружается. Проект содержит всю информацию о разрабатываемой программе и применяемом микроконтроллере. Он состоит из целого набора файлов.

Главный из них - файл проекта. Он имеет расширение aps. Файл проекта содержит сведения о типе процессора, частоте тактового генератора и т.д. Он также содержит описание всех остальных файлов, входящих в проект. Все эти сведения используются при отладке и трансляции программы.

Кроме файла aps, проект должен содержать хотя бы один файл с текстом программы. Такой файл имеет расширение asm. Недостаточно просто поместить файл asm в директорию проекта. Его нужно еще включить в проект. Проект может содержать несколько файлов asm. При этом один из них является главным. Остальные могут вызываться из главного при помощи оператора. include. На этом заканчивается список файлов проекта, которые создаются при участии программиста.

Algorithm Builder предназначен для производства полного цикла разработки начиная от ввода алгоритма, включая процесс отладки и заканчивая программированием кристалла.

Algorithm Builder довольно нетрадиционная программа в плане языка программирования; ассемблер, завернутый в красивую обертку визуального программирования. Algorithm Builder - визуальный ассемблер или построитель ассемблера с помощью которого на выходе можно получить максимально эффективный код.

Работа с переменными и константами организована гениально просто. Инициализация производиться в отдельном окне в виде таблицы - освобождая алгоритм от лишних записей. В буквальном смысле слова все разложено по полочкам.

Algorithm Builder имеет удобный настройщик периферии (таймеры, UART, ADC, SPI и т.д.) позволяющий, не читая даташитов, просто выбрать необходимые параметры работы устройства в окне настройки. В этом же окне Builder честно покажет набор инструкций, обеспечивающих эти параметры.

Переходы осуществляются в программе очень наглядно - вектором. Если требуется перейти по условию в какую либо точку программы - нужно просто провести вектор в эту точку. Это освобождает программу от бесчисленных имен меток, которые в классическом ассемблере являются неизбежным балластом. Переходы по именованным меткам так же возможны.

Внутрисхемное программирование кристалла. При использовании внутрисхемного программатора микроконтроллер подключается к COM порту компьютера через несложный адаптер (три диода и несколько резисторов). Также есть вариант USB подключения. Программатор ведет подсчет числа перепрограммировании кристалла, сохраняя счетчик непосредственно в кристалле. Процесс программирования кристалла очень прост - в два "хода".

Мониторная отладка на кристалле. Algorithm Builder обеспечивает мониторную отладку на кристалле (On Chip debug) которая позволяет наблюдать содержимое реального кристалла в заданной точке останова. При этом для связи микроконтроллера с компьютером используется только один вывод, причем по выбору пользователя. Мониторная отладка может быть применена к любому типу кристалла, имеющего SRAM. Это софтверный вариант debugWIRE.

Для того, чтобы написанная программа превратилась в результирующий код и заработала в конкретном микропроцессорном устройстве, ее нужно оттранслировать и "зашить" в программную память микроконтроллера.

При написании программ обычно нельзя обойтись без процедуры отладки. Отладка выполняется на компьютере при помощи специальной инструментальной программы - отладчика. Он позволяет пошагово выполнять отлаживаемую программу, а также выполняет ее поэтапно с использованием, так называемых точек останова.

В процессе выполнения программы под управлением отладчика можно на экране компьютера:

1) видеть содержимое любого регистра микроконтроллера;
2) видеть содержимое ОЗУ и EEPROM;
3) наблюдать за последовательностью выполнения команд, контролируя правильность отработки условных и безусловных переходов;
4) наблюдать за работой таймеров, отработкой прерываний.

В процессе отладки также можно наблюдать логические уровни на любом внешнем выходе микроконтроллера. А также имитировать изменение сигналов на любом входе. Процесс отладки позволяет убедиться в том, что разрабатываемая программа работает именно так, как нужно.

Существует три основных вида отладчиков:

- программные;
- аппаратные;
- комбинированные программно-аппаратные.

Программный отладчик

Программный отладчик - это компьютерная программа, которая имитирует работу процессора на экране компьютера. Она не требует наличие реальной микросхемы или дополнительных внешних устройств и позволяет отладить программу чисто виртуально.

Однако программный отладчик позволяет проверить только логику работы программы. При помощи такого отладчика невозможно проверить работу схемы в режиме реального времени или работу всего микропроцессорного устройства в комплексе, т. е невозможно гарантировать правильную работу и всех подключенных к микроконтроллеру дополнительных микросхем и элементов.

Аппаратный отладчик

Основа аппаратного отладчика - специальная плата, подключаемая к компьютеру, работающая под его управлением и имитирующая работу реальной микросхемы микроконтроллера. Плата имеет выводы, соответствующие выводам реальной микросхемы, на которых в процессе отладки появляются реальные сигналы.

При помощи этих выводов отладочная плата может быть включена в реальную схему. Возникающие в процессе отладки электрические сигналы можно наблюдать при помощи осциллографа. Можно нажимать реальные кнопки и наблюдать работу светодиодов и других индикаторов.

Здесь, как и в предыдущем случае, можем видеть всю информации об отлаживаемой программе: наблюдать содержимое регистров, ОЗУ, портов ввода-вывода; контролировать ход выполнения программы.

В аппаратном отладчике так же, как и в программном, можно выполнять программу в пошаговом режиме и применять точки останова. Недостатком аппаратного отладчика является его высокая стоимость.

Имитаторы

Имитаторы - программы, которые позволяют на экране компьютера "собрать" любую электронную схему, включающую в себя самые разные электронные компоненты:

a) транзисторы;

b) резисторы;

c) конденсаторы;

d) операционные усилители;

e) логические и цифровые микросхемы, в том числе и микроконтроллеры.

Такие программы обычно содержат обширные базы электронных компонентов и конструктор электронных схем. Собрав схему, можно виртуально записать в память микроконтроллера вашу программу, а затем "запустить" всю схему в работу.

Для контроля результатов работы схемы имитатор имеет виртуальные вольтметры, амперметры и осциллографы, которые можно "подключать" к любой точке схемы, "измерять" различные напряжения, а также "снимать" временные диаграммы.

Такие программы в настоящее время получают все большее распространение. Они позволяют разработать любую схему с микроконтроллером или без него, без использования паяльника и реальных деталей. На экране компьютера можно полностью отладить свою схему и лишь, потом браться за паяльник.

Недостатком данного отладчика является то, что он требует значительных вычислительных ресурсов. Особенно в том случае, когда отлаживается схема, включающая как микроконтроллер, так и некоторую аналоговую часть. Кроме того, имитатор не всегда верно имитирует работу некоторых устройств. Однако подобные программы имеют очень большие перспективы.

WinAVR

WinAVR представляет собой набор инструментальных средств для работы с микроконтроллерами семейства AVR фирмы ATMEL. В него вошли следующие компоненты:

a) компилятор языка Си avr-gcc,

b) библиотека компилятора avr-libs,

c) ассемблер avr-as,

d) интерфейс программатора avrdude,

e) интерфейс JTAG ICE avarice,

f) Debuger avr-gdb,

g) редактор programmers notepad.

Весь этот набор собран в один инсталляционный пакет и предназначен для установки на платформу Windows.

Главным преимуществом моего выбора именно этой микросхемы является ее широкая доступность и не высокая цена.

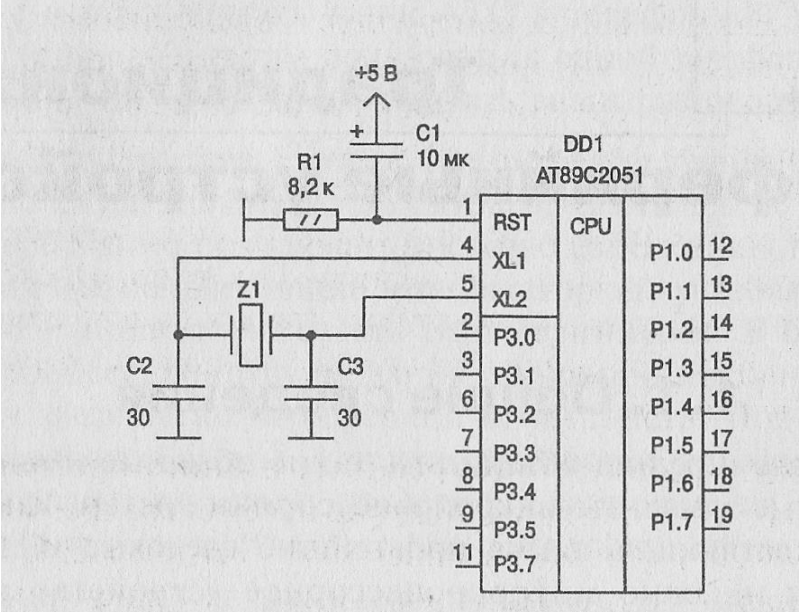


Рис.2. Типовая схема включения микроконтроллера AT89C2051

Элементы R1, C1 составляют цепь начального сброса микроконтроллера. Она служит для перевода в исходное состояние всех внутренних систем микроконтроллера сразу после включения питания. Кварцевый резонатор Z1 определяет частоту встроенного тактового генератора микроконтроллера. Этот генератор предназначен для синхронизации всех внутренних процессов микроконтроллера. Микросхема AT89C2051 допускает выбирать частоту кварцевого резонатора до 24 МГц. Нижний предел частоты не ограничивается. Конденсаторы C2 и C3 - это согласующие элементы для кварца. Микроконтроллер AT89C2051 допускает применение в качестве времязадающей цепи резонансного контура, и даже подключение внешнего тактового генератора. Оставшиеся выводы

микроконтроллера представляют собой два порта ввода/вывода, которые обозначены P1 и P3. Именно к этим двум портам и подключаются периферийные устройства.

Практически ни одно микропроцессорное устройство не обходится без кнопок и простейших датчиков на основе обычных контактов. При помощи этого вида периферийных элементов в микропроцессорное устройство поступает различная информация, которая используется для изменения алгоритма работы программы.

Примером может служить датчик поворота (Рис.3) - механические контакты, связанные с поворачиваемым устройством.

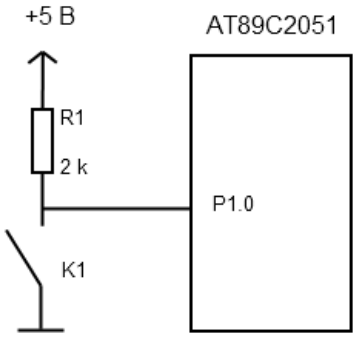


Рис.3. Простая схема подключения датчика на основе геркона

На вход микроконтроллера через резистор R1 подается напряжение от источника питания +5 В. Микросхема воспринимает это напряжение как сигнал логической единицы. При срабатывании датчика контакты замыкаются и соединяют вывод микроконтроллера с общим проводом. В результате напряжение на входе P 1.0 падает до нуля. В следствии микросхема воспринимает входной уровень сигнала как логический ноль. Резистор R1 при этом служит токоограничивающим элементом, предотвращая короткое замыкание между шиной питания и общим проводом.

Рассмотрим программу для обслуживания вышеупомянутого датчика (*Листинг 2.3.),* которая, постоянно опрашивает датчик и в зависимости от состояния запускает одну из двух специальных процедур.

Листинг 2.3.

	; Программа обработки сигнала с датчика
1	m1: mov p1.0,#1 ; Записываем 1 в соответствующий
2	разряд порта
	mov c,p1.0 ; Читаем состояние датчика в битовый
3	аккумулятор
4	
5	jc m2 ; Если контакты датчика разомкнуты, перейти к m2
6	call proc1 ; Вызов процедуры обработки нажатия
7	контакта
	jmp m1 ; Возврат к началу (следующий цикл считывания)
	m2: call proc1 ; Вызов процедуры обработки размыкания контакта
	jmp m1 ; Возврат к началу (следующий цикл считывания)

Здесь явно видно, что программа записывает в линию P1.0 сигнал логической единицы (строка 1). Это необходимо для того, чтобы данная линия могла работать на ввод информации.

Следующая команда считывает бит информации, поступающей от датчика, и помещает ее в регистр признака переноса (строка 2). В микропроцессорной технике принято ячейку признака переноса обозначать как CY. Ячейка CY используется как аккумулятор для битовых операций. Если в момент считывания сигнала контакты датчика были разомкнуты, то в ячейке CY окажется логическая единица. Если контакты замкнуты, то там будет логический ноль.

Оператор условного перехода jc осуществляет оценку содержимого CY (строка 3). Если в CY логический ноль, то управление передается на метку m2**,** и выполняется команда call proc2 (строка 6). В противном случае

передача управления не происходит и выполняется команда call proc1 (строка 4). Оператор call - это вызов подпрограммы. Поэтому, в зависимости от состояния датчика вызывается одна из двух подпрограмм: proc1 или proc2.

1.3. Организация памяти и портов ввода/вывода микроконтроллера.

Микроконтроллеры AVR имеют раздельные пространства адресов памяти программ и данных (гарвардская архитектура). Организация памяти Микроконтроллер ATMega8 показана на рис. 4.

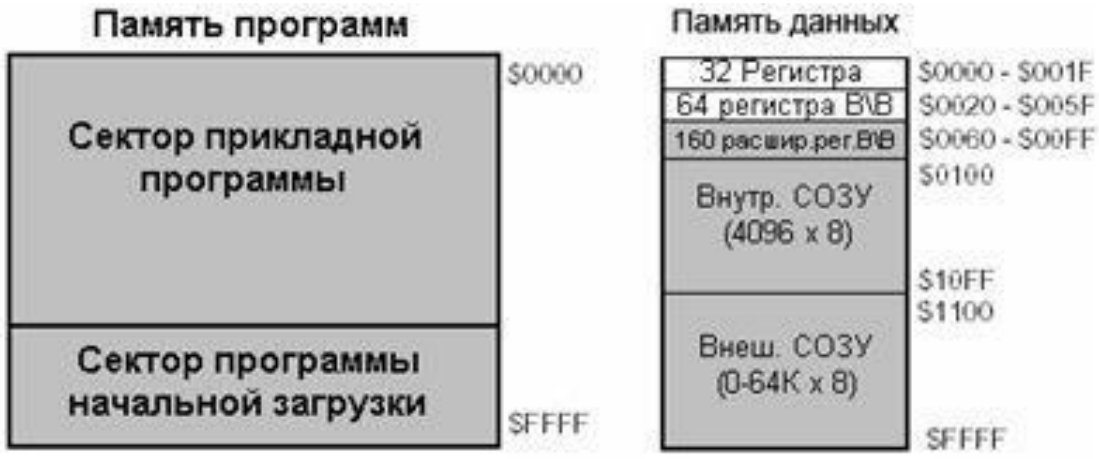


Рис.4. Организация памяти микроконтроллера ATmega8

Высокие характеристики семейства AVR обеспечиваются следующими особенностями архитектуры:

• В качестве памяти программ используется внутренняя флэш-память. Она организована в виде массива 16-разрядных ячеек и может загружаться программатором, либо через порт SPI;

• 16-разрядные память программ и шина команд вместе с одноуровневым конвейером позволяют выполнить большинство инструкций за один такт синхрогенератора (50 нс при частоте FOSC=20 МГц);

• память данных имеет 8-разрядную организацию. Младшие 32 адреса пространства занимают регистры общего назначения, далее следуют 64 адреса регистров ввода-вывода, затем внутреннее ОЗУ данных объемом до

4096 ячеек. Возможно применение внешнего ОЗУ данных объемом до 60 Кбайт;

• внутренняя энергонезависимая память типа EEPROM объемом до 4 Кбайт представляет собой самостоятельную матрицу, обращение к которой осуществляется через специальные регистры ввода-вывода.

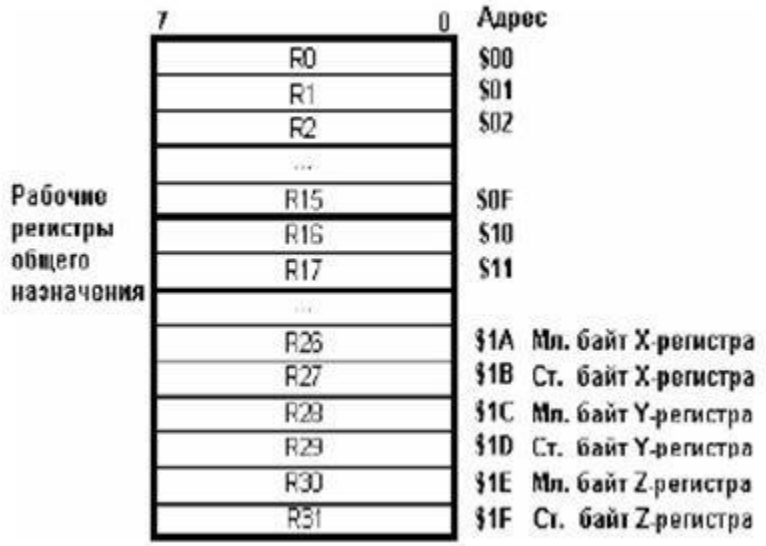


Рис.5. Регистры общего назначения микроконтроллера ATmega8

Как видно из рис. 4 и 5, 32 регистра общего назначения (РОН) включены в сквозное адресное пространство ОЗУ данных и занимают младшие адреса. Хотя физически регистры выделены из памяти данных, такая организация обеспечивает гибкость в работе. Регистры общего назначения прямо связаны с АЛУ. Каждый из регистров способен работать как аккумулятор. Большинство команд выполняются за один такт, при этом из регистров файла могут быть выбраны два операнда, выполнена операция и результат возвращен в регистровый файл. Старшие шесть регистров могут использоваться как три 16-разрядных регистра, и выполнять роль, например, указателей при косвенной адресации.

Следующие 64 адреса за регистрами общего назначения занимают регистры ввода-вывода (регистры управления/состояния и данных). В этой

области сгруппированы все регистры данных, управления и статуса внутренних программируемых блоков ввода-вывода. При использовании команд IN и OUT используются адреса ввода-вывода с $00 по $3F. Но к регистрам ввода-вывода можно обращаться и как к ячейкам внутреннего ОЗУ. При этом к непосредственному адресу ввода-вывода прибавляется $20. Адрес регистра как ячейки ОЗУ приводится далее в круглых скобках. Регистры ввода-вывода с $00 ($20) по $1F ($3F) имеют программно доступные биты. Обращение к ним осуществляется командами SBI и CBI, а проверка состояния – командами SBIS и SBIC [2-ст.9].

1.4. Разработка микропроцессорной системы на основе микроконтроллера

Микропроцессорные системы на основе микроконтроллера используются чаще всего в качестве встроенных систем для решения задач управления некоторым объектом. Важной особенностью данного применения является работа в реальном времени, т.е. обеспечение реакции на внешние события в течение определенного временного интервала. Такие устройства получили название контроллеров.

Перед разработчиком микропроцессорные системы стоит задача реализации полного цикла проектирования, начиная от разработки алгоритма функционирования и заканчивая комплексными испытаниями в составе изделия, а, возможно, и сопровождением при производстве. Сложившаяся к настоящему времени методология проектирования контроллеров может быть представлена так, как показано на рис. 6.

В техническом задании формулируются требования к контроллеру с точки зрения реализации определенной функции управления. Техническое задание включает в себя набор требований, который определяет, что пользователь хочет от контроллера и что разрабатываемый прибор должен делать. Техническое задание может иметь вид текстового описания, не

свободного в общем случае от внутренних противоречий.

На основании требований пользователя составляется функциональная спецификация, которая определяет функции, выполняемые контроллером для пользователя после завершения проектирования, уточняя тем самым, насколько устройство соответствует предъявляемым требованиям. Она включает в себя описания форматов данных, как на входе, так и на выходе, а также внешние условия, управляющие действиями контроллера.

1.5. Основные этапы разработки

Этап разработки алгоритма управления является наиболее ответственным, поскольку ошибки данного этапа обычно обнаруживаются только при испытаниях законченного изделия и приводят к необходимости дорогостоящей переработки всего устройства. Разработка алгоритма обычно сводится к выбору одного из нескольких возможных вариантов алгоритмов, отличающихся соотношением объема программного обеспечения и аппаратных средств.

При этом необходимо исходить из того, что максимальное использование аппаратных средств упрощает разработку и обеспечивает высокое быстродействие контроллера в целом, но сопровождается, как

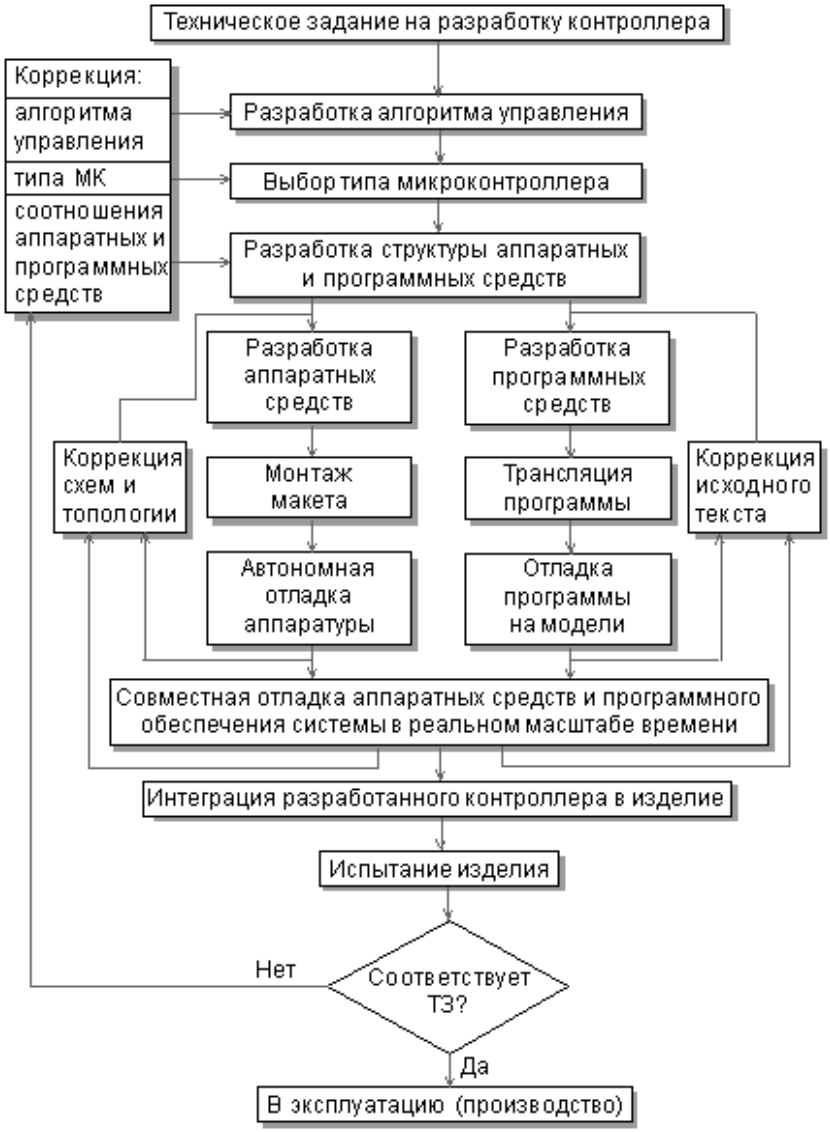


Рисунок 6.- Основные этапы разработки контроллера

правило, увеличением стоимости и потребляемой мощности. При выборе типа микроконтроллера учитываются следующие основные характеристики:

- разрядность;

- быстродействие;

- набор команд и способов адресации;

- требования к источнику питания и потребляемая мощность в различных режимах;

- объем ПЗУ программ и ОЗУ данных;

- возможности расширения памяти программ и данных;

- наличие и возможности периферийных устройств, включая средства поддержки работы в реальном времени (таймеры, процессоры событий и

т.п.);

- возможность перепрограммирования в составе устройства;

- наличие и надежность средств защиты внутренней информации;

- возможность поставки в различных вариантах конструктивного исполнения;

- стоимость в различных вариантах исполнения;

- наличие полной документации;

- наличие и доступность эффективных средств программирования и отладки микроконтроллера;

- количество и доступность каналов поставки, возможность замены изделиями других фирм.

Список этот не является исчерпывающим, поскольку специфика проектируемого устройства может перенести акцент требований на другие параметры микроконтроллера.

Номенклатура выпускаемых в настоящее время микроконтроллеров исчисляется тысячами типов изделий различных фирм. Современная стратегия модульного проектирования обеспечивает потребителя разнообразием моделей микроконтроллера с одним и тем же процессорным ядром. Такое структурное разнообразие открывает перед разработчиком возможность выбора оптимального микроконтроллера, не имеющего функциональной избыточности, что минимизирует стоимость комплектующих элементов.

Однако для реализации на практике возможности выбора оптимального микроконтроллера необходима достаточно глубокая проработка алгоритма управления, оценка объема исполняемой программы и числа линий сопряжения с объектом на этапе выбора микроконтроллера. Допущенные на данном этапе просчеты могут впоследствии привести к необходимости смены модели микроконтроллера и повторной разводки печатной платы макета контроллера. В таких условиях целесообразно выполнять предварительное моделирование основных элементов прикладной

программы с использованием программно-логической модели выбранного микроконтроллера.

На этапе разработки структуры контроллера окончательно определяется состав имеющихся и подлежащих разработке аппаратных модулей, протоколы обмена между модулями, типы разъемов. Выполняется предварительная проработка конструкции контроллера. В части программного обеспечения определяются состав и связи программных модулей, язык программирования. На этом же этапе осуществляется выбор средств проектирования и отладки.

Глава 2. РАЗРАБОТКА ПРОГРАММЫ АВТОМАТИЗАЦИИ РАСЧЕТА ТЕМПЕРАТУРНОГО РЕЖИМА ПОМЕЩЕНИЙ

2.1. Постановка задачи

Микроконтроллер может управлять различными устройствами и принимать от них данные при минимуме дополнительных узлов, так как большое число периферийных схем уже имеется непосредственно на кристалле микроконтроллера. Это позволяет уменьшить размеры конструкции и снизить потребление от источника питания [1- ст. 6].

AVR - это семейство 8-разрядных RISC-микроконтроллеров фирмы Atmel. Эти микроконтроллеры позволяют решить множество задач встроенных систем. Они отличаются от других распространённых в наше время микроконтроллеров большей скоростью работы, большей

универсальностью. Кроме того, они очень легко программируются. Их можно перепрограммировать до 1000 раз, причем непосредственно в собранной схеме [1- ст. 9].

Имеются 3 подсемейства микроконтроллеров AVR:

Tiny AVR- недорогие миниатюрные микроконтроллеры в 8-выводном исполнении;

Classic AVR - основная линия микроконтроллера с производительностью отдельных модификаций до 16 MISP, FLASH-памятью программ 2…8 Кб, памятью данных EEPROM 64...512 байт, оперативной памятью данных SRAM 128…512 байт;

Mega AVR - с производительностью 4…16 MISP для сложных приложений требующих большого объёма памяти, FLASH-памятью программ до 128 Кб, памятью данных EEPROM 64...512 байт, оперативной памятью данных SRAM 2…4 байт, встроенным 10-разрядным 8-канальным АЦП, аппаратным умножителем 8x8.

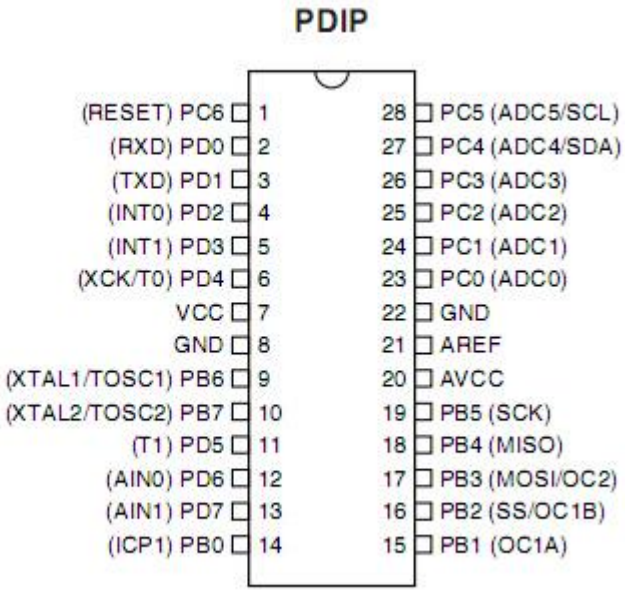
Интересной особеннотью семейства микроконтроллеров является то, что система комманд всего семейства совместима при переносе программы со слабого на более мощный микроконтроллер [1- ст. 11].

Структура микроконтроллера ATMega8

В качестве ядра микроконтроллерной системы для измерения температуры в диапазоне от -10 до +55С согласно техническому заданию был выбран AVR микроконтроллера типа ATMega8.

Назначение выводов

На рис.7 изображен корпус и приведено назначение выводов микроконтроллера. В скобках указана альтернативная функция вывода.



ATmega8, ATmega8L

8-разрядные микроконтроллеры с 8 Кбайтами внутрисистемно программируемой Flash памяти

Отличительные особенности:

8-разрядный высокопроизводительный AVR микроконтроллер с малым потреблением

Прогрессивная RISC архитектура

130 высокопроизводительных команд, большинство команд выполняется за один тактовый цикл

32 8-разрядных рабочих регистра общего назначения Полностью статическая работа

Приближающаяся к 16 MIPS (при тактовой частоте 16 МГц) производительность

Встроенный 2-цикловый перемножитель

Энергонезависимая память программ и данных

8 Кбайт внутрисистемно программируемой Flash памяти (In-System Self-Programmable Flash)

Обеспечивает 1000 циклов стирания/записи

Дополнительный сектор загрузочных кодов с независимыми битами

блокировки

Обеспечен режим одновременного чтения/записи (Read-While-Write)

512 байт EEPROM

Обеспечивает 100000 циклов стирания/записи

1 Кбайт встроенной SRAM

Программируемая блокировка, обеспечивающая защиту программных средств пользователя

Встроенная периферия

Два 8-разрядных таймера/счетчика с отдельным предварительным делителем, один с режимом сравнения

Один 16-разрядный таймер/счетчик с отдельным предварительным делителем и режимами захвата и сравнения

Счетчик реального времени с отдельным генератором

Три канала PWM

8-канальный аналого-цифровой преобразователь (в корпусах TQFP и MLF)

6 каналов с 10-разрядной точностью

2 канала с 8-разрядной точностью

6-канальный аналого-цифровой преобразователь (в корпусе PDIP)

4 канала с 10-разрядной точностью

2 канала с 8-разрядной точностью

Байт-ориентированный 2-проводный последовательный интерфейс

Программируемый последовательный USART

Последовательный интерфейс SPI (ведущий/ведомый)

Программируемый сторожевой таймер с отдельным встроенным генератором

Встроенный аналоговый компаратор

Специальные микроконтроллерные функции

Сброс по подаче питания и программируемый детектор кратковременного снижения напряжения питания

Встроенный калиброванный RC-генератор

Внутренние и внешние источники прерываний

Пять режимов пониженного потребления: Idle, Power-save, Power-down, Standby и снижения шумов ADC

Выводы I/O и корпуса

23 программируемые линии ввода/вывода

28-выводной корпус PDIP, 32-выводной корпус TQFP и 32-выводной корпус MLF

Рабочие напряжения

2,7 - 5,5 В (ATmega8L)

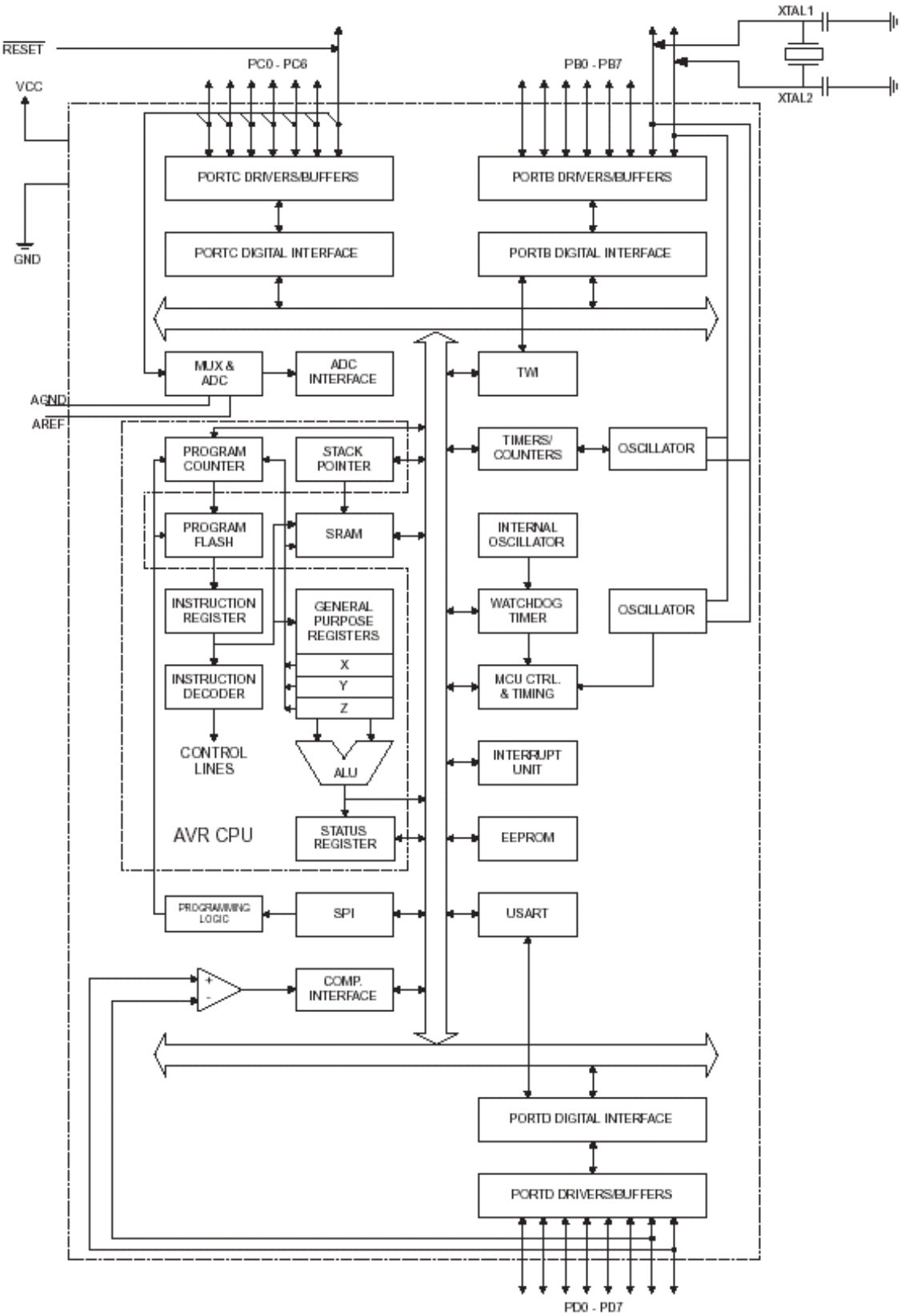
4,5 - 5,5 В (ATmega8)

Рабочая частота

0 - 8 МГц (ATmega8L)

0 - 16 МГц (ATmega8)

Блок-схема:



Расположение выводов:

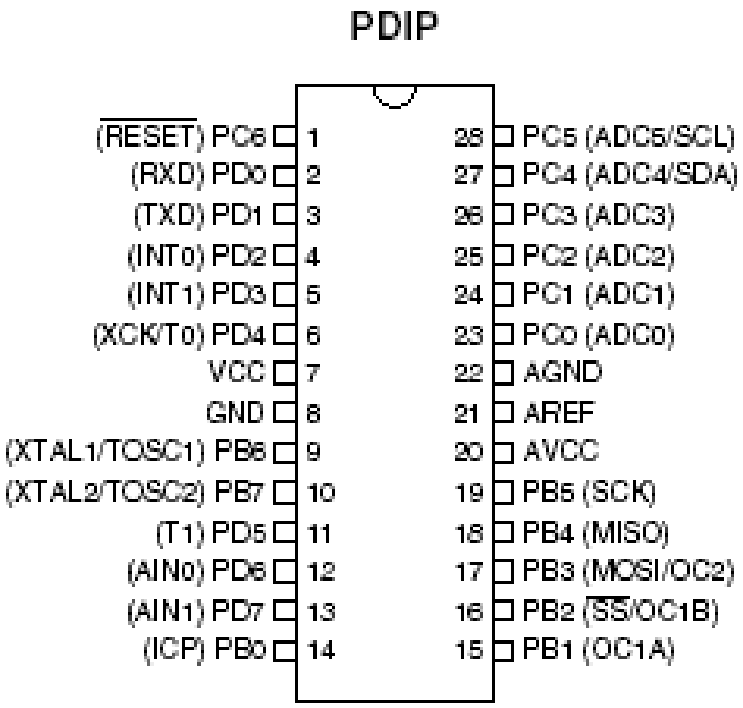


Рис.7. Вид корпуса и обозначение выводов микроконтроллера ATMega8.

Микроконтроллер ATmega8 включает следующие функциональные блоки: - 8-разрядное арифметическо-логическое устройство (АЛУ);

- внутреннюю флэш-память программ объемом 128 Кбайт с возможностью внутрисистемного программирования через последовательный интерфейс;

- 32 регистра общего назначения;

- внутреннюю EEPROM память данных объемом 4 Кбайт;

- внутреннее ОЗУ данных объемом 4 Кбайт;

- 6 параллельных 8-разрядных портов;

- 4 программируемых таймера-счетчика;

- 10-разрядный 8-канальный АЦП и аналоговый компаратор;

- последовательные интерфейсы UART0, UART0, TWI и SPI;

- блоки прерывания и управления (включая сторожевой таймер).

Port A (PA7..PA). 8-разрядный двунаправленный порт. К выводам порта могут быть подключены встроенные нагрузочные резисторы (отдельно к каждому разряду). Выходные буферы обеспечивают ток 20 мА и способность прямо управлять светодиодным индикатором. При использовании выводов порта в качестве входов и установке внешнего

сигнала в низкое состояние, ток будет вытекать только при подключенных встроенных нагрузочных резисторах. Порт A при наличии внешней памяти данных используется для организации мультиплексируемой шины адреса/данных.

Port B (PB7..PB0). 8-разрядный двунаправленный порт со встроенными нагрузочными резисторами. Выходные буферы обеспечивают ток 20 мА. При использовании выводов порта в качестве входов и установке внешнего сигнала в низкое состояние, ток будет вытекать только при подключенных встроенных нагрузочных резисторах. Порт B используется также при реализации специальных функций.

Port C (PC7..PC0). Порт C является 8-разрядным выходным портом. Выходные буферы обеспечивают ток 20 мА. Порт C при наличии внешней памяти данных используется для организации шины адреса.

Port D (PD7..PD0). 8-разрядный двунаправленный порт со встроенными нагрузочными резисторами. Выходные буферы обеспечивают ток 20 мА. При использовании выводов порта в качестве входов и установке внешнего сигнала в низкое состояние, ток будет вытекать только при подключенных встроенных нагрузочных резисторах. Порт D используется также при реализации специальных функций.

Port E (PE7..PE0). 8-разрядный двунаправленный порт со встроенными нагрузочными резисторами. Выходные буферы обеспечивают ток 20 мА. При использовании выводов порта в качестве входов и установке внешнего сигнала в низкое состояние, вытекающий через них ток обеспечивается только при подключенных встроенных нагрузочных резисторах. Порт E используется также при реализации специальных функций.

Port F (PF7..PF0). 8-разрядный входной порт. Входы порта используются также как аналоговые входы аналого-цифрового преобразователя.

#RESET. Вход сброса. Для выполнения сброса необходимо удерживать низкий уровень на входе более 50 нс.

XTAL1, XTAL2. Вход и выход инвертирующего усилителя генератора тактовой частоты.

TOSC1, TOSC2. Вход и выход инвертирующего усилителя генератора таймера/счетчика.

#WR, #RD. Стробы записи и чтения внешней памяти данных.

ALE. Строб разрешения фиксации адреса внешней памяти. Строб ALE используется для фиксации младшего байта адреса с выводов AD0-AD7 в защелке адреса в течение первого цикла обращения. В течение второго цикла обращения выводы AD0-AD7 используются для передачи данных.

AVCC. Напряжение питания аналого-цифрового преобразователя. Вывод подсоединяется к VCC через низкочастотный фильтр.

AREF. Вход опорного напряжения для аналого-цифрового преобразователя. На этот вывод подается напряжение в диапазоне между AGND и AVCC.

AGND. Это вывод должен быть подсоединен к отдельной аналоговой земле, если она есть на плате. В ином случае вывод подсоединяется к общей земле.

#PEN. Вывод разрешения программирования через последовательный интерфейс. При удержании сигнала на этом выводе на низком уровне после включения питания, прибор переходит в режим программирования по последовательному каналу.

VCC, GND. Напряжение питания и земля.

2.2 Разработка структурной схемы устройства

Структурная схема для цифрового термометра приведена на рисунке 8.

Рисунок 8. – структурная схема цифрового термометра

На рисунке 8. показано:

ЖКИ – жидко-кристаллический индикатор;

МК – микроконтроллер;

Д– цифровой датчик температуры;

DS18S20– последовательный интерфейс.

Микроконтроллер выполняет две основные функции:

производит опрос датчика температуры и сохраняет в ОЗУ значения температуры, полученные от датчика в каждом цикле опроса температуры;

по требованию компьютера, микроконтроллер отсылает в компьютер значения температуры из ОЗУ от датчика температуры.

Измеренная температура не только сохраняется, но и выводится на жидкокристаллический экран (ЖКИ). Графические возможности экрана позволяют отображать не только цифровые значения температуры, но и отображать изменения температуры во времени в виде графиков. Также микроконтроллер может сохранять не одно значение температуры, а несколько (до 20 значений).

Если значение температуры выходит за диапазон 30-400С, то микроконтроллер формирует предупреждающий сигнал с помощью светодиодов.

Кнопка производит сброс, если измеренные значения температуры выходят за значения от -10С до +55С.

При необходимости измеренные значения температуры можно ввести на экран компьютер с помощью последовательного интерфейса DS18S20.

2.3 Разработка функциональной схемы

В данной выпускной квалификационной работе разработан цифровой термометр с использованием микроконтроллера AVR ATMega8. Схемы устройства дано на рисунке 9.

Прибор осуществляет измерения путём заряда конденсатора до уровня примерно равного VCC, последующего разряда его через опорный резистор с одновременным подсчётом внутренних тактов до того момента, пока на входе CIN не появится лог. «0». Далее конденсатор снова заряжается до значения, близкого к VCC и разряжается через термистор, при этом также подсчитываются тактовые импульсы. Неизвестное сопротивление резистора вычисляется как отношение числа тактов при разряде конденсатора термистором к числу тактов при разряде через опорный резистор и последующим домножением на известное значение сопротивления опорного резистора. Программа вычисляет сопротивление термистора, переводит это значение в температуру, переводит её в градусы Цельсия и отображает значение на ЖКИ.

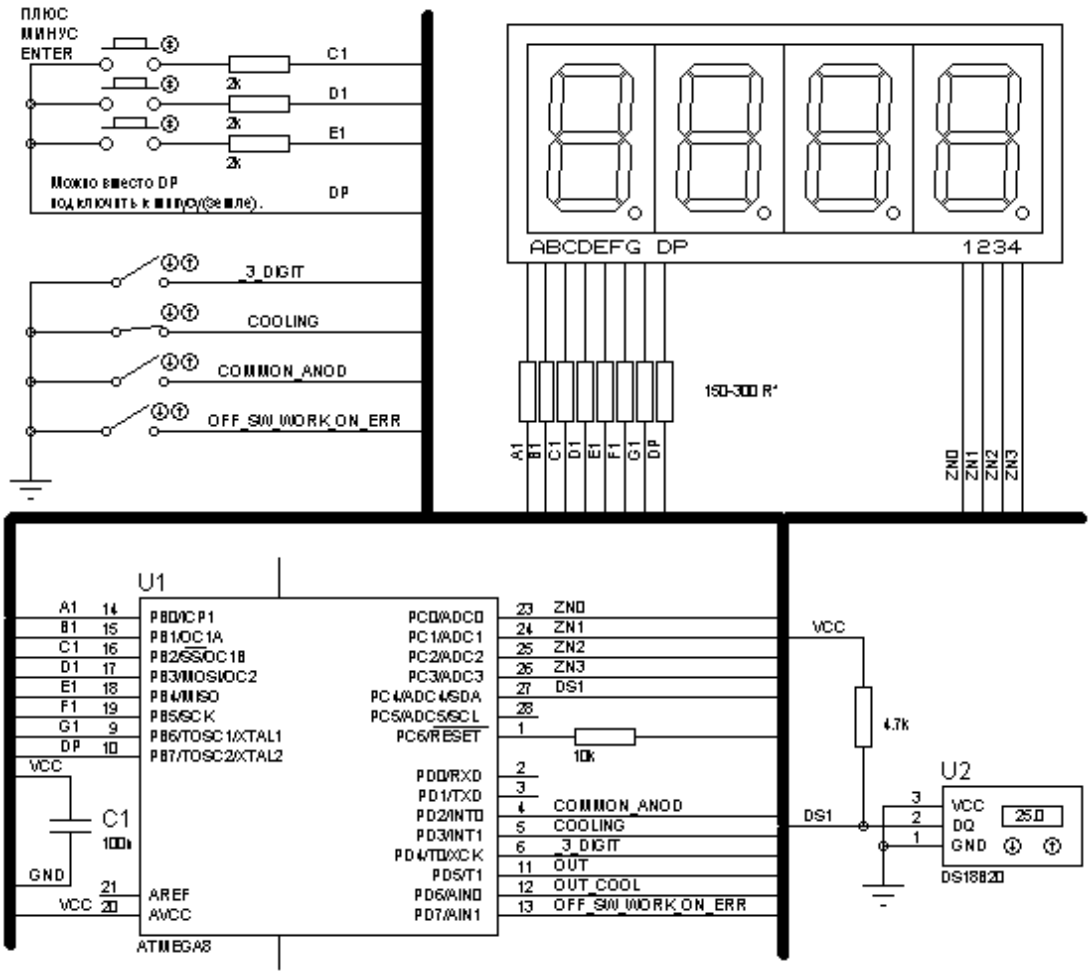
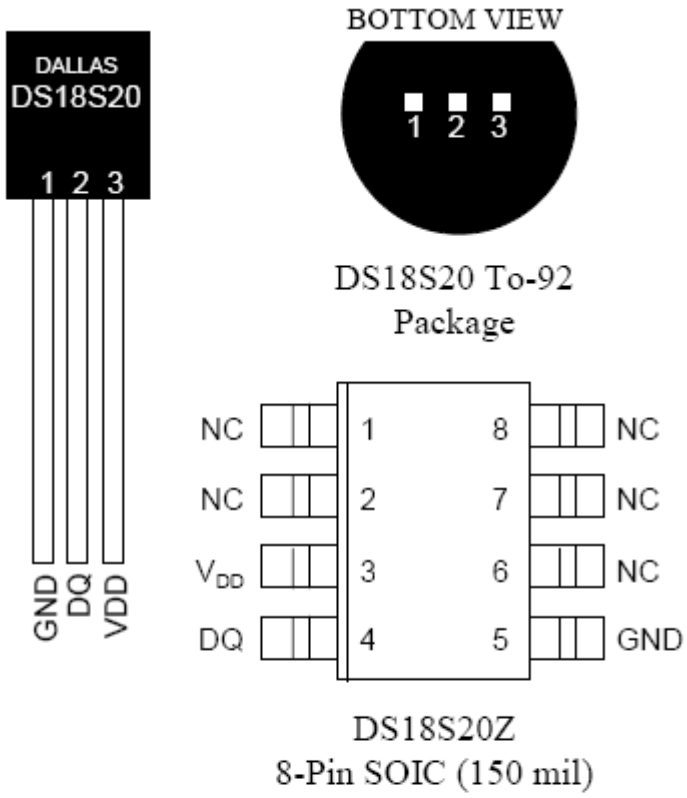


Рис.9. Схема устройства микроконтроллера.

К выводам 50,51 микроконтроллера подключен датчик температуры *DS18S20*, представленная на рисунке 10.



PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

Рисунок 10.- Датчик температуры DS18S20.

DS18S20+ - Калиброванный цифровой термометр(термодатчик) фирмы DALLAS с однопроводным 1-Wire-интерфейсом и разрядностью 9 бит.

Основные характеристики DS18S20+:

Уникальный интерфейс 1-WIRE

Нет внешних элементов

Питание 3,0..5,5V

Диапазон температур -55°C..+125°C

Точность ±0.5°C (в диапазоне -10°C..+85°C)

Разрешение 9 бит

Корпус TO-92

Быстродействие 750mS (9 бит)

Диапазон измеряемых DS18S20 температур от -55°C до +125°C. Считываемый с микросхемы цифровой код является результатом непосредственного прямого измерения температуры и не нуждается в дополнительных преобразованиях. Разрешающая способность встроенного АЦП - 9 разрядов выходного кода. Абсолютная погрешность преобразования меньше 0,5°C в диапазоне контролируемых температур -10°C до +85°C. Максимальное время полного 9-ти разрядного преобразования ~750 мс. Энергонезависимая память температурных уставок микросхемы обеспечивает запись произвольных значений верхнего и нижнего контрольных порогов. Кроме того, термометр содержит встроенный логический механизм приоритетной сигнализации в 1-Wire-линию о факте выхода контролируемой им температуры за один из выбранных порогов. Узел 1-Wire-интерфейса компонента организован таким образом, что существует теоретическая возможность адресации неограниченного количества подобных устройств на одной 1-Wire-линии. Термометр имеет индивидуальный 64-разрядный регистрационный номер (групповой код 010H) и обеспечивает возможность работы без внешнего источника энергии, только за счет паразитного питания 1-Wire-линии. Питание микросхемы через отдельный внешний вывод производится напряжением от 3,0 В до 5,5 В. Термометр размещается в транзисторном корпусе TO-92, или в 8-контактном корпусе SO для поверхностного монтажа (DS18S20Z).

Для формирования правильного импульса сброса в момент включения питания к выводу (SET) микроконтроллера подключена RC-цепочка (R11,C9). Эта цепь используется для задержки запуска микроконтроллера при включении питания, что нужно для его правильного запуска, а также для ручного перезапуска микроконтроллера нажатием на кнопку SB1 . Цепь сброса по включению питания обеспечивает запрет включения процессора до тех пор, пока напряжение питания не достигнет безопасного уровня. После того, как напряжение питания достигнет уровня включения, процессор не включается до тех пор, пока встроенный таймер не обработает несколько

периодов сторожевого таймера. Внешний сброс обрабатывается по низкому уровню на выходе SET. Вывод должен удерживаться в низком состоянии, по крайней мере, два периода тактовой частоты. После снятия сигнала 0 с вывода SET через некоторое время микроконтроллер запускается. Кроме того, для информирования пользователя о рабочем режиме подключается светодиод VD3. Этот светодиод мигает зеленым цветом, когда производится чтение значений температуры из датчика. В остальное время светодиод не горит. Так как чтение значений температуры происходит непрерывно, кроме случаев прерывания вызванных для связи с компьютером, то светодиод мигает с периодом 0,8 мс. И перестает мигать в момент обмена информацией с компьютером.

2.4. Разработка алгоритма управления

Программа работы микроконтроллера заключается в следующем:

при нажатии кнопки SET (C1) производится непрерывный (циклический) опрос датчика и сохранение полученных значений температуры в ОЗУ.

Непрерывный вывод полученных значений температуры на ЖКИ индикатор (цифровое отображение информации на экране).

Формирование сигнала предупреждения с помощью блока светодиодов в случае выхода значения температуры за пределы -10C<T<55C (согласно техническому заданию).

Алгоритм программы приведен на рисунке 13.

Первым действием в программе производятся начальные установки микроконтроллера. В них устанавливается указатель стека на последнюю ячейку ОЗУ, исходное состояние каналов связи с датчиками температуры и UART, скорость обмена по UART, разрешаются прерывания от таймера/счетчика и от UART, переписывается количество и индивидуальные

адреса датчиков температуры из EEPROM в ОЗУ, в регистры записываются необходимые константы.

Когда начальные установки завершены, начинается часть программы, которая производит опрос датчика температуры. Она будет циклически повторятся, пока подводится питание к микроконтроллеру или пока не возникнет запрос на прерывание. Опрос датчика температуры начинается с сигнала сброса на линии (блок 2 рисунок 13.) Затем следует команда игнорирования адреса датчика температуры SKIP ROM [CCh].

Команда начала измерения температуры CONVERT T [44h] (блок 4, рисунок 13.) разрешает преобразование значений температуры в цифровой вид для датчика.

Аналого-цифровое преобразование значений температуры занимает время от 750 мс до 800 мс. Поэтому, чтобы получить правильное значение температуры, необходимо выждать паузу 800 мс (блок 5, рисунок 13.). Пауза выдерживается с помощью таймера/счетчика 0. Во время паузы можно совершать другие действия (например, произвести обмен данными с компьютером или вывести результаты на ЖКИ).

После паузы производится опрос датчика. Опрос датчика начинается с сигнала сброса на линии связи с датчиком (блок 6, рисунок 13.). После сигнала сброса и ответного сигнала от датчика следует команда MATCH ROM [55h]. Эта команда сообщает датчику, что после неё на линии связи будет выставлен индивидуальный 64-х битовый адрес датчика. После того, как адрес выставлен на линии, датчик температуры сравнивает выставленный адрес со своим собственным адресом, и, после этого к работе с микроконтроллером датчик готов.

В блоке 7 производится чтение значения температуры и запись его в соответствующие ячейки ОЗУ.

В блоке 8 производится ветвление программы: если измеренное значение температуры не выходит за пределы -10C<T<55C , то результат

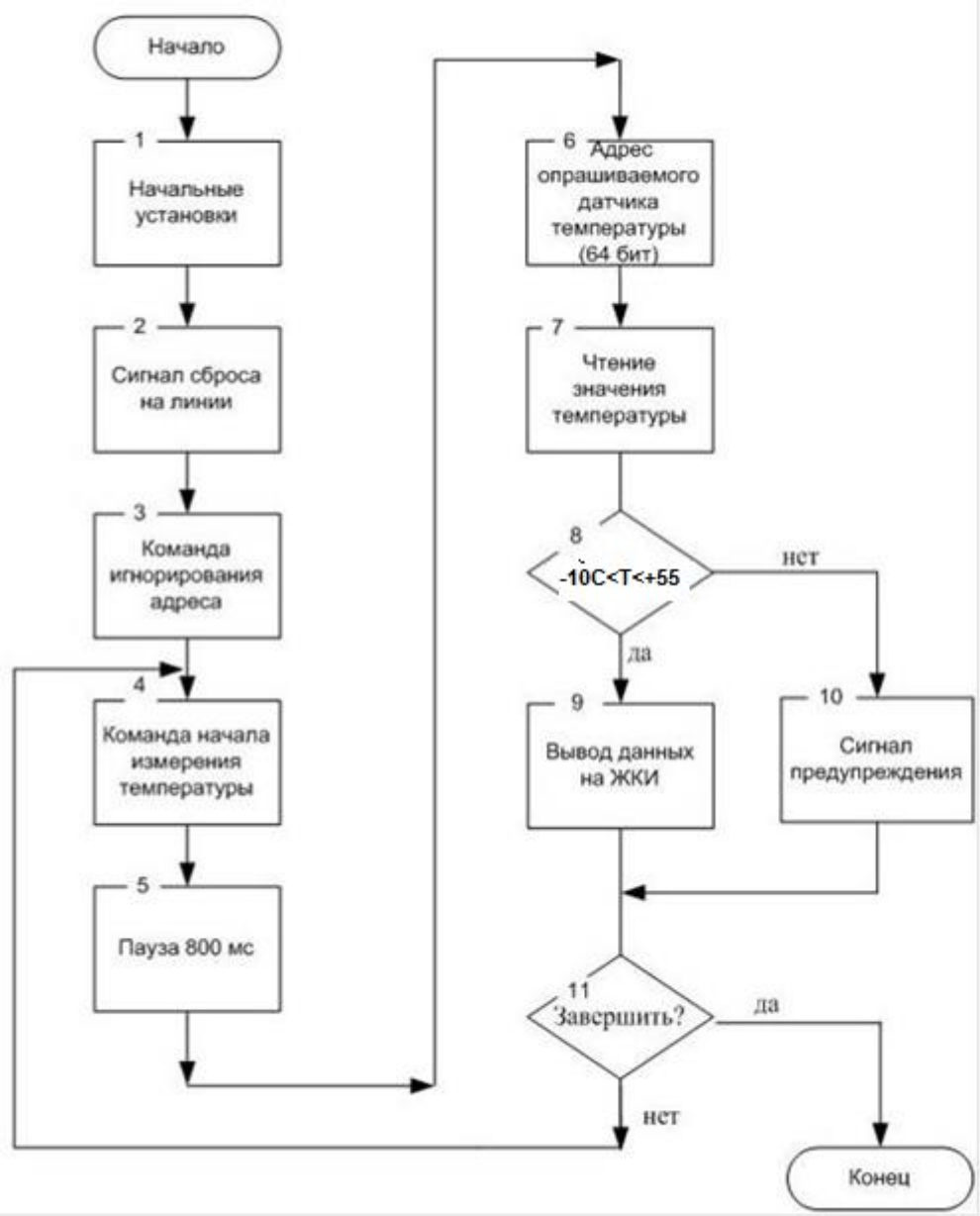


Рисунок 13. – Алгоритм работы микроконтроллера

выводится на экран ЖКИ (блок 9). Если же измеренное значение температуры выходит за пределы диапазона, то загорятся световые индикаторы: VD1- если температура меньше -10С или VD2 –если больше 55С (блок 10).

Если необходимо продолжать измерять температуру (блок 11), то переходят к блоку 4, если нет, то тогда происходит завершение программы.

Программа работы микроконтроллера для измерения температуры приведена в приложении II.

Глава 3. ОПИСАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ПОРЯДОК ЕГО ИСПОЛЬЗОВАНИЯ

3.1. Разработка программного обеспечения микроконтроллера

Для показа программы прошитой на микроконтроллер пользуемся системаой Electronics Labcenter (Proteus 7.6) предназначенный для проектирования многослойных печатных плат (ПП) аналоговых, цифровых и аналого-цифровых устройств. Она состоит из двух основных модулей:

- ISIS – графический редактор принципиальных схем со встроенным менеджером библиотек ;
- ARES - графический редактор печатных плат со встроенным менеджером библиотек и авто трассировщиком;

Рассмотрим основные этапы проектирования печатных плат (ПП):

1. Создание и редактирование символов элементов электрических принципиальных схем в ISIS, корпусов элементов ПП в ARES и взаимосвязей между ними;
2. Создание электрических принципиальных схем в ISIS и экспорт списка соединений в ARES;
3. Размещение корпусов элементов на ПП в ARES;
4. Выбор необходимой стратегии трассировки ПП с последующей автотрассировкой в ARES;
5. Вывод чертежей ПП и схем электрических принципиальных на принтер или плоттер.

Программа Proteus представляет собой мощную систему схемотехнического моделирования, сделанной на основе виртуальных моделей электронных элементов. Специфической чертой данного программного пакета (Proteus) — есть отличная возможность моделирования различной работы программируемых устройств: микропроцессоров, контроллеров, DSP и т.д.

Кроме этого в пакет Proteus заложена специальная система проектирования и моделирования печатных плат. Программа Proteus умеет симулировать работу таких контроллеров: ARM7, 8051, PIC, AVR, Motorola, Basic Stamp. Внутренняя библиотека компонентов имеет различные справочные данные. Она поддерживает МК: 8051, PIC, HC11, AVR, ARM7/LPC2000 и другие широко распространенные процессоры. Вдобавок к этому в программе содержатся более 6000 цифровых и аналоговых моделей всевозможных устройств.

Программа Proteus прекрасно работает с большинством компиляторов и ассемблерами. PROTEUS VSM делает довольно достоверно моделирование и отлаживание весьма сложных устройств, в которых может находиться несколько «МК» разных семейств в одном устройстве!

Необходимо учитывать и понимать, что любое моделирование электронных схем не может, абсолютно точно повторяет работу реального устройства. Но для общего отлаживания, какого-либо алгоритма работы «МК», этого вполне будет достаточно. Программа PROTEUS имеет большую библиотеку электронных компонентов, а отсутствующие модели можно сделать самостоятельно. В случае, когда какой-либо компонент не программируемый, то на сайте производителя скачать его SPICE модель, и добавить в подходящий корпус.

Проект "Термостат" развился до очередной стадии миниатюрности. Почти не изменившись по физическим габаритам, он тем не менее стал еще проще по схемотехнике, а число функций стало даже больше!

Развитие проекта целиком и полностью следует поставить в заслугу пользователю **Toledo**, который сподвиг меня на разработку. Кроме этого, **Toledo** приложил максимум усилий по тестированию прошивки в реальном "железе" (не было у меня под руками нужного микроконтроллера),

а так же снял видеоролик и фотографии. Он же разработал несложный вариант печатной платы. В общем, масса благодарностей ему!

Особенность устройства в том, что управление всеми режимами осуществляется **одной-единственной кнопкой**, причем, как показали наши с **Toledo** эксперименты, это весьма удобно. Учитывая неистребимое желание измерять температуру с точностью до десятых долей градуса, я реализовал и эту возможность, совместив диапазон термометра от -55.0°C до +125.0°C с трехразрядным семисегментным индикатором. Это вторая изюминка устройства: десятые доли индицируются только в диапазоне -9.9...+99.9°C, а другие температуры отображаются уже без десятых долей. Думаю, это оптимальное решение. Третья изюминка - уже не нова: это режим двухпорогового термостата (т.е. с гистерезисом) с противофазными выходами, что позволяет использовать устройство для поддержания температуры от -50°C до +99°C как путем управления нагревателем, так и охладителем (вентилятором).

Принципиальная схема термостата

На рисунке показана схема термостата. Она элементарна, собственно говоря, это не полностью завершенное устройство, а лишь его основа: источник питания и выходные каскады можно придумать любые.

Микроконтроллер U1 типа ATMega8, датчик U2 – DS18S20 , его можно и нужно вынести на проводках в нужное место подальше от нагревающихся компонентов, способных исказить показания. Питание 5В можно получить от любого источника. Индикатор - четырехразрядный 7-сегментный «динамический» (с общими анодами или катодами - все равно). Используются кнопки C1, E1, D1 - само собой любые. В прошивке реализована посегментная динамическая индикация, поэтому число токоограничительных резисторов сведено к семи, т.е. к минимуму, их сопротивление должно ограничивать ток через сегмент индикатора на уровне не более 30 мА.

Выходы out1 и out2 способны выдать (или принять) ток до 40 мА, поэтому оконечный каскад может быть любым - от маломощного пятивольтового реле до мощного транзисторного ключа. Эти выходы работают всегда в противофазе.

Программа написана на языке Си, ее текст доступен, компилируется при помощи WinAVR. При компиляции обязательно нужно включать максимальную оптимизацию по размеру кода, иначе в память микроконтроллера не поместится. Кстати, в текущей версии остается свободным около 12% памяти программ - есть шанс дополнить программу еще каким-либо полезным свойством. Кстати, очень скоро это будет сделано: в прошивку будет введен дополнительный режим контроля "предельных" уровней температуры (по просьбе одного из посетителей сайта).

Для тех, кто не готов разбираться с исходными текстами, имеются 2 варианта готовых прошивок - для индикаторов с общими анодами и катодами. Перед прошивкой (или после) необходимо установить фьюзы микроконтроллера CKSEL=0100, т.е. активировать встроенный RC-генератор 8 МГц, остальные фьюзы можно оставить в предустановленном на заводе-изготовителе состоянии. Кроме прошивок, доступен исходный текст программы.

Работа устройства: имеется 5 функциональных режимов:

1. Индикация температуры.
2. Индикация верхнего порога термостата.
3. Индикация нижнего порога термостата.
4. Коррекция верхнего порога.
5. Коррекция нижнего порога.

Собственно термостатирование, т.е. сравнение текущей температуры с пороговыми значениями и формирование соответствующих выходнях сигналов, ведется постоянно в любом из рассмотренных режимов. Переключение уровня при повышении температуры на выходах происходит, когда температура превышает верхний порог, а при понижении температуры

- когда опускается ниже нижнего, т.е. промежуток между значениями порогов есть гистерезис термостата.

В первом режиме на индикаторе просто отображается *текущая* температура.

Во втором и третьем режимах отображаются соответствующие пороги термостата. Для порогов значения задаются только в целых градусах. Чтобы можно было отличить одно значение от другого, в первом разряде индикатора дополнительно подсвечиваются сегменты A или D соответственно для верхнего и нижнего порогов.

Переключение первой тройки режимов осуществляется кратковременным нажатием на кнопку, причем только режим №1 стабильный - остальные автоматически переходят к нему, если кнопка не нажимается более 2,5 секунд.

Из режимов индикации порогов можно перейти к режимам изменения соответствующего порога, если нажать и удерживать кнопку более 2,5 секунд. С этого момента начинается интересное (т.е. та самая изюминка управления одной кнопкой). Как только включается режим изменения значения порога, сразу начинает мерцать соответствующий сегмент A или D на первом индикаторе (признак коррекции порога), и одновременно, пока нажата кнопка, происходит быстрое изменение значения. Дождавшись, когда порог «проскочит» желаемое значение, нужно отпустить кнопку. После этого можно кратковременными нажатиями скорректировать значение в противоположном быстрому изменению направлении. Если при удержании кнопки происходит изменение не в том направлении - надо отпустить ее и снова нажать надолго.

Поясню на примере. Допустим, установлены пороги **-5** и **+15** градусов, нужно сделать их **-2** и **+2**. Включаем режим коррекции верхнего порога, нажав и удерживая кнопку во втором режиме. Спустя 2,5 секунды значение начинает быстро меняться в сторону увеличения. Дождавшись, когда появится на индикаторе **15**, отпускаем кнопку. Если не повезло и на

индикаторе **16** - не беда: нажимаем кнопку кратко и значение уменьшается на 1, т.е. становится **15**, *что и требовалось*. Не трогаем кнопку 2,5 секунды - мерцание сегмента **A** прекращается - снова включен режим 2. Нажимаем кнопку кратко, включая тем самым режим 3. Теперь нажимаем кнопку надолго и ждем, пока включится режим коррекции нижнего порога. Как только замерцал сегмент**D**, значение начинает быстро уменьшаться - ждем, пока оно не достигнет значения **-2** и отпускаем кнопку. Проскочили? - не беда! Кратковременным нажатием кнопки возвращаем по одному проскоченному градусу... Далее - как и ранее: не трогаем кнопку 2,5 секунды, по и после выключения режима коррекции не трогаем кнопку - в момент автоматического включения режима 1 произойдет запоминание новых значений порогов.

Попробую сформулировать алгоритм коррекции одним предложением. В режиме коррекции изменение значения осуществляется с шагом в 1 градус, причем краткое нажатие кнопки просто изменяет значение на один шаг, а длительное нажатие приводит к ускоренному изменению, после которого знак шага меняется на противоположный. Надеюсь, все понятно. Во всяком случае, привыкнуть к этому алгоритму довольно просто, и, я надеюсь, он покажется вам удобным.

3.2. Порядок использования программного обеспечения

Порядок открытие программы программы Code VisionAVR : Пуск\ Все программы\CV AVR.

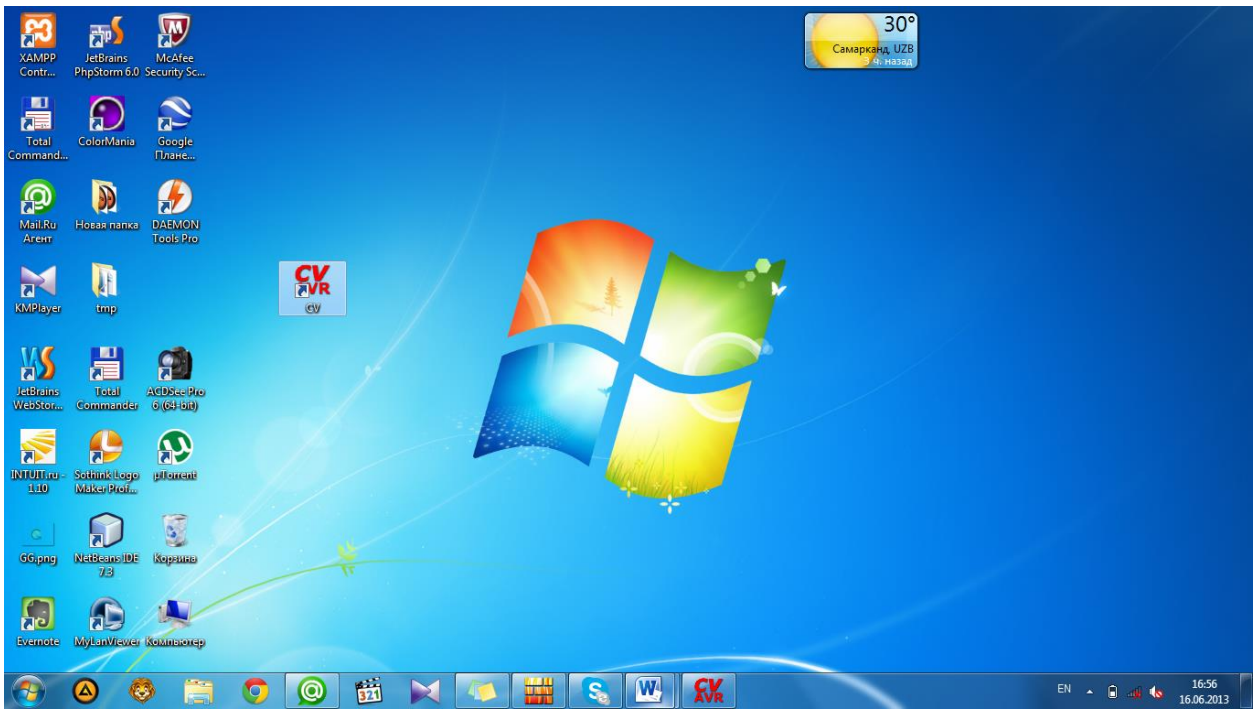


Рис.3.1. Запуск программы Code VisionAVR.

После нажатия на кнопку «CV AVR» появляется окно с парамеирами программы Code VisionAVR

Рис.3.2. Титул программы Code VisionAVR.

попадаем в рабочую область программы:

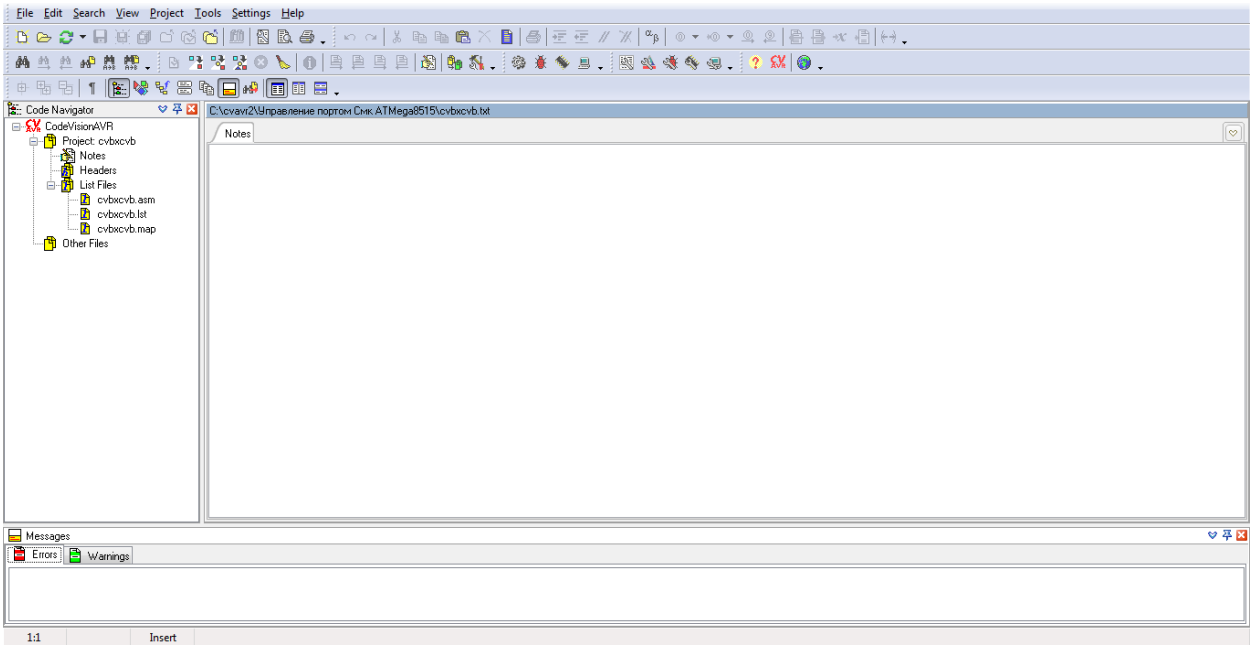


Рис.3.3. Окно IDE Code VisionAVR.

В этой строке выведены названия меню, которые предоставляют к командам

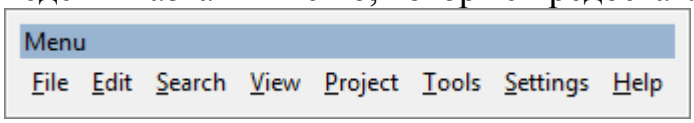


Рис.3.4. Строка меню.

При работе с программой следует использовать команды, которые сообщают Code VisionAVR, что именно требуется сделать. Команды Code VisionAVR можно найти в меню, а многие из них можно вызвать с помощью кнопок панель инструментов.

Эта команда позволяет создать новый исходный файл или новый проект.

Кнопка на панели инструментов  .

После выбора этой команды появится диалоговое окно **Create New File** показанное на Рис.*3.5.*

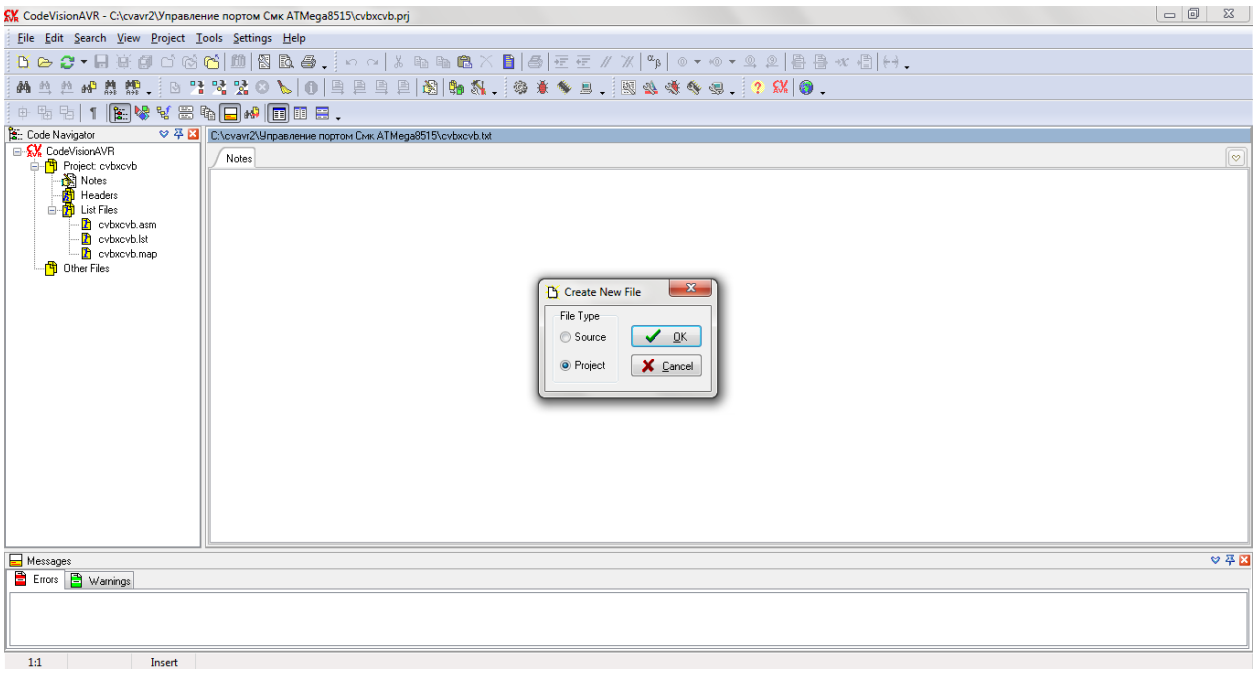


Рис.3.5. Диалоговое окно **Create New File**

Создаем папку для проекта:

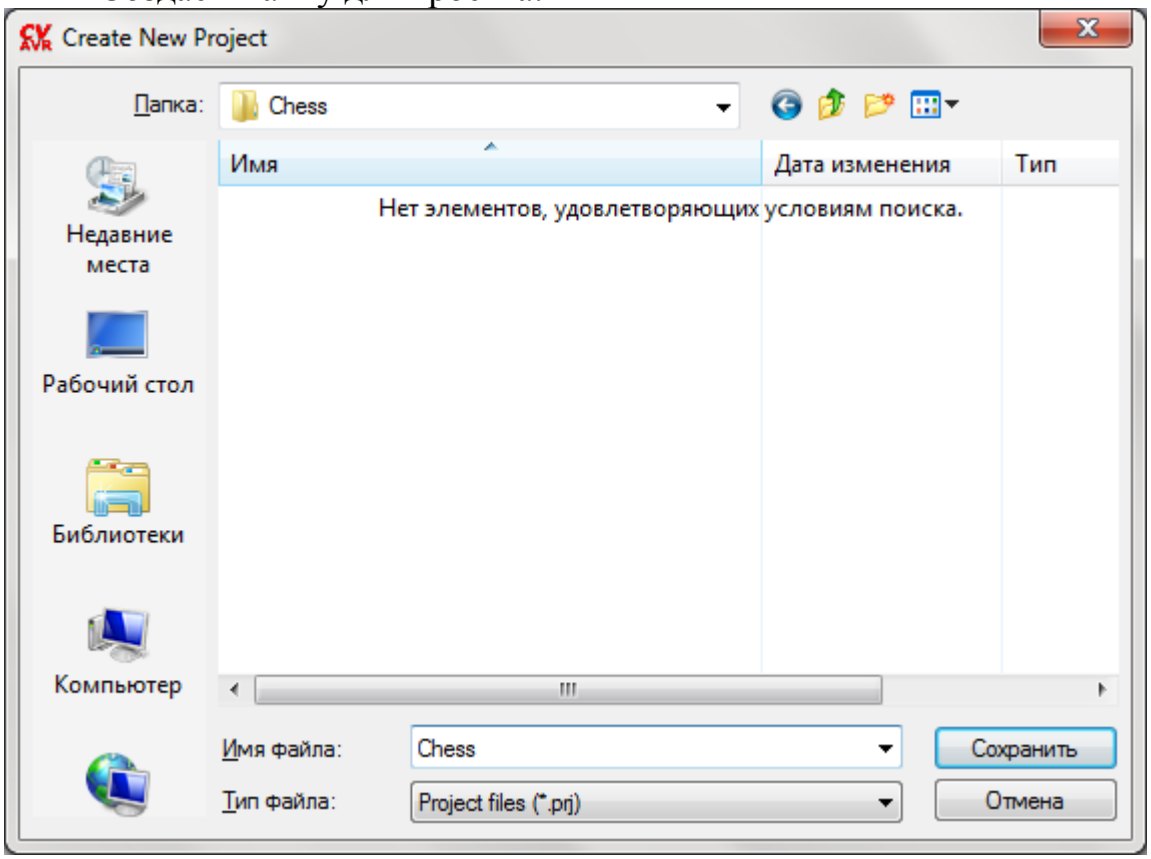


Рис.3.6. Диалоговое окно **Create New Project.**

Файл проект будет иметь расширение **.prj.**

После того как имя файла проекта и его местоположение определены, следует щёлкнуть по кнопке Сохранить. После этого проект будет создан, и будет предложен его сконфигурировать, как показано на *Рис.3.7*

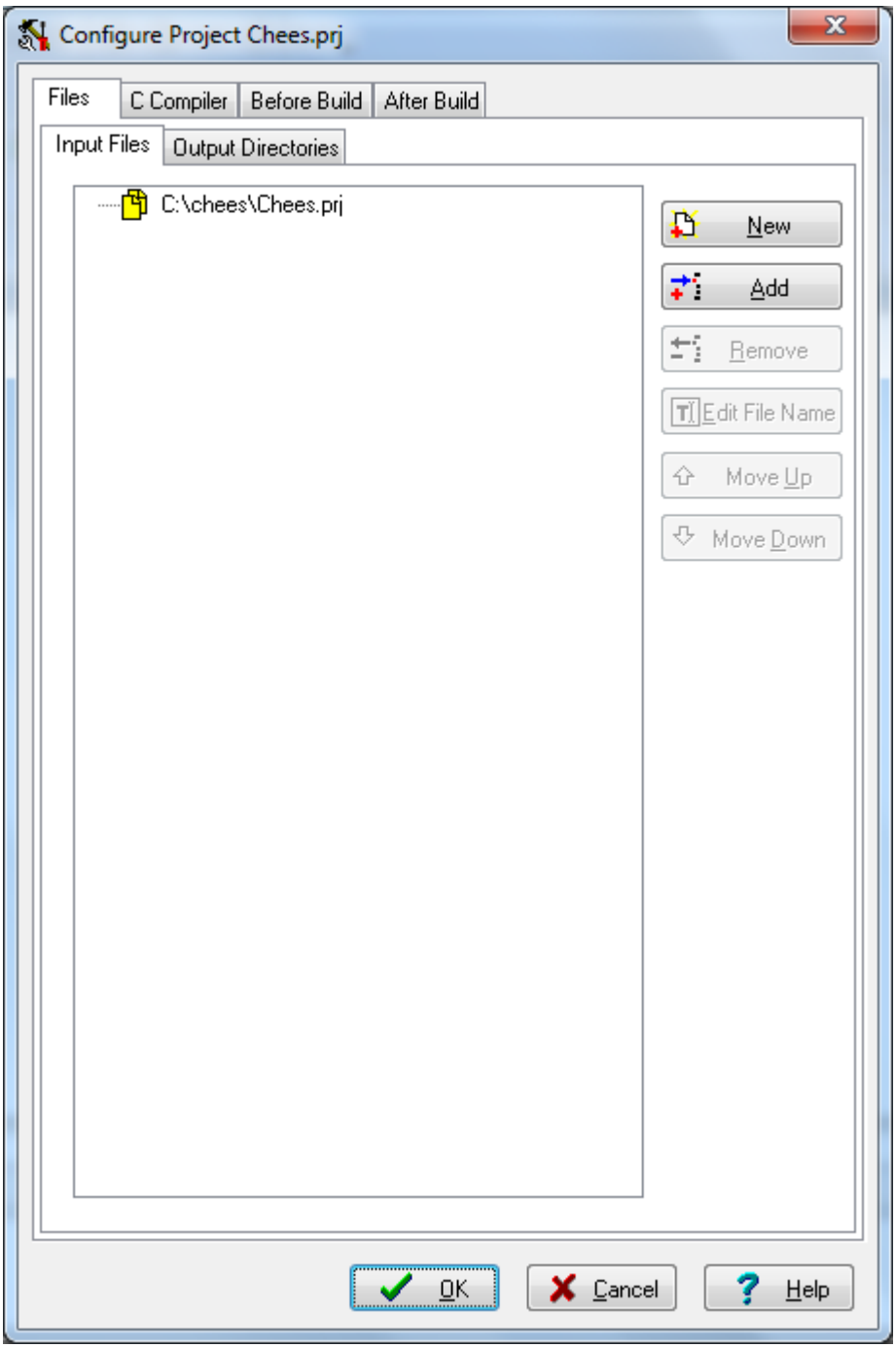


Рис.3.7 Диалоговое окно **Configure Project**

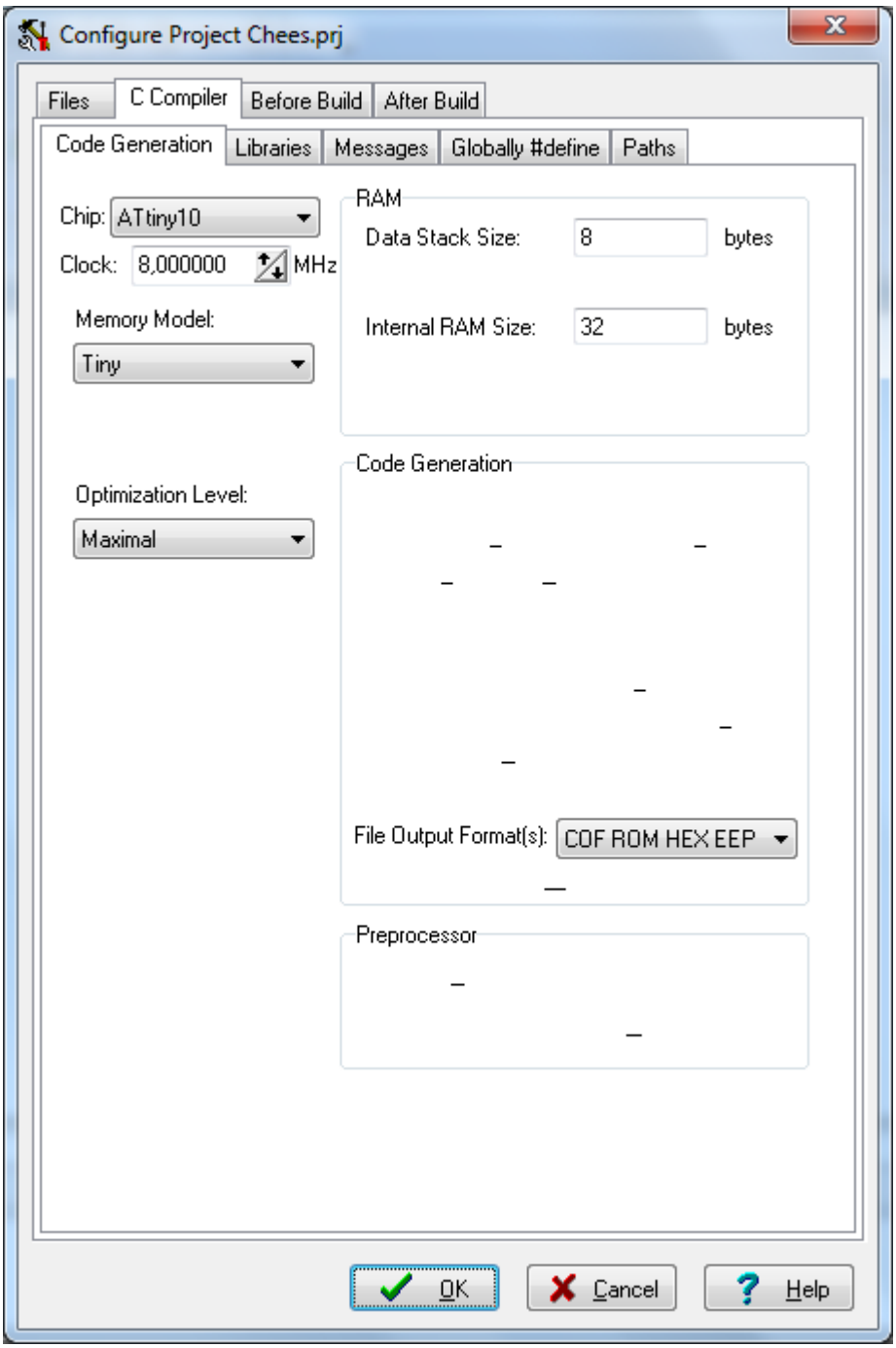


Рис.3.8 Выбор закладки **C Compiler** в диалоговом окне **Configure Project**

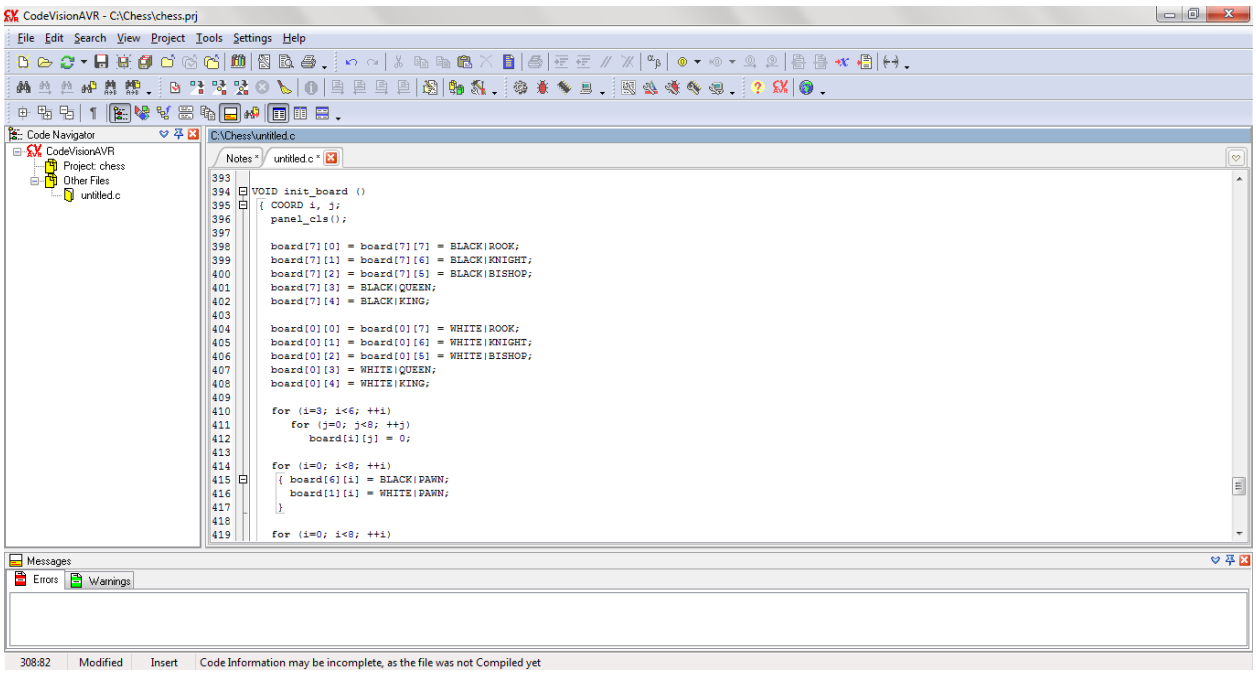


Рис.4.0 Исходный код программы

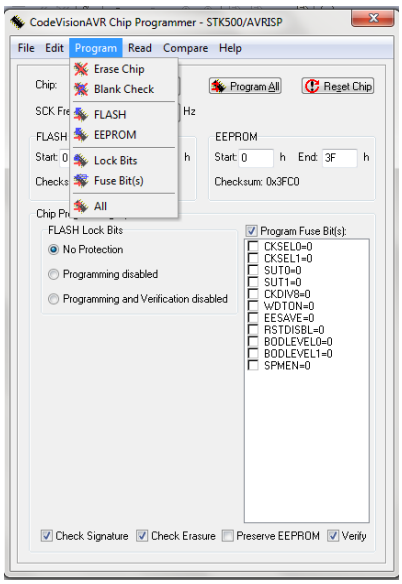


Рис.4.1 Ход вбивания кода программы в микроконтроллер

После компилирования кода программы, чтобы прошить микроконтроллер и показывать его работу мы пользуемся системой ….

Подробный ход открытия этой программы.

Порядок открытие программы программы Proteus 7.7: Пуск\ Все программы\Proteus 7 Professional\ISIS 7 Professional.

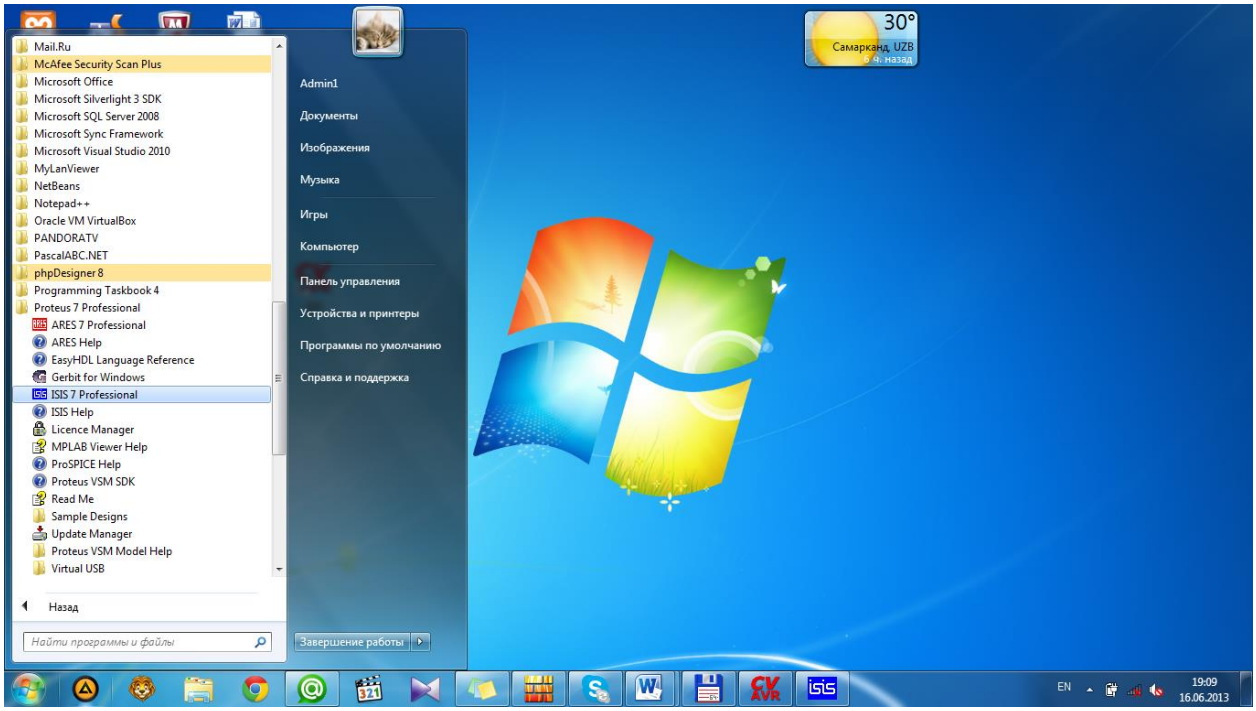


Рис.3.1. Запуск программы Proteus 7.6.

После нажатия на кнопку «ISIS 7 Professional» попадаем в окно системы Electronics Labcenter предназначенной для проектирования многослойных печатных плат (ПП) аналоговых, цифровых и аналого-цифровых устройств.

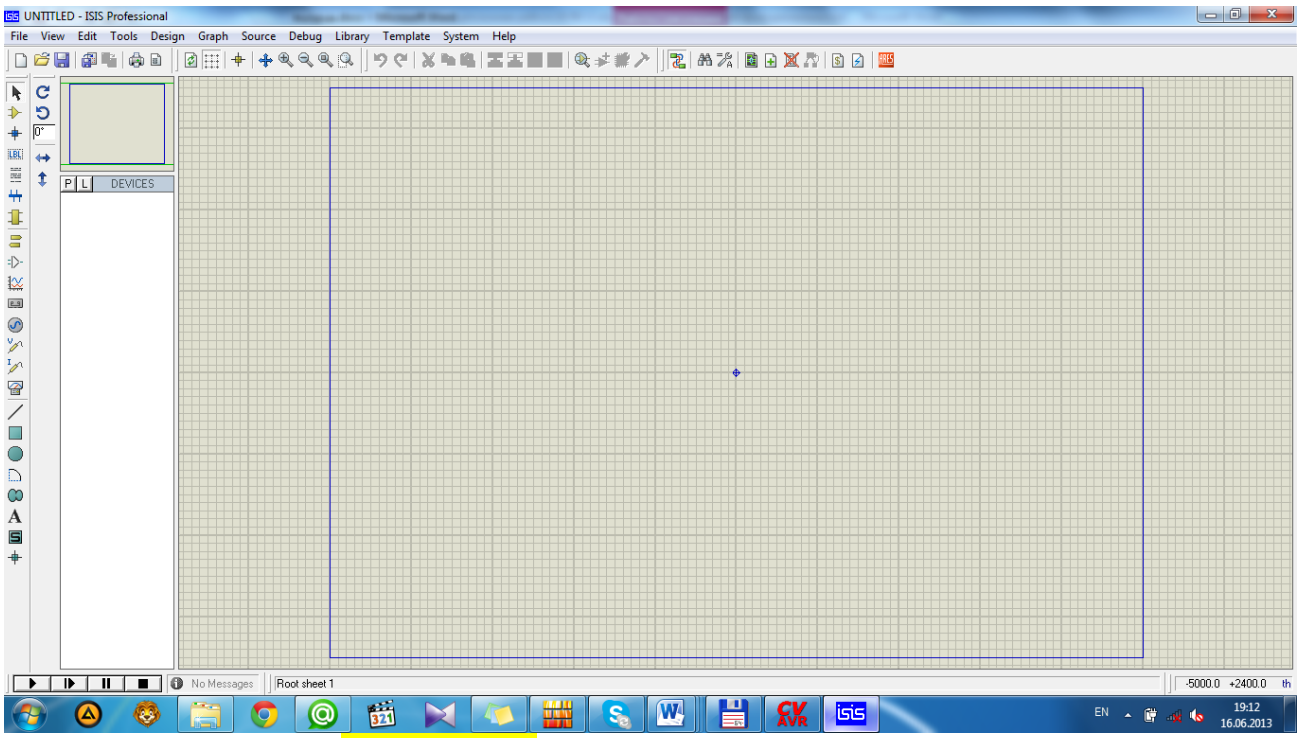


Рис.3.2.Рабочее окно программы Proteus 7.7

Для открытия нашего проекта переходим по ссилке: File\Open Design попадаем в слудущее окно в котором выбираем наш проект

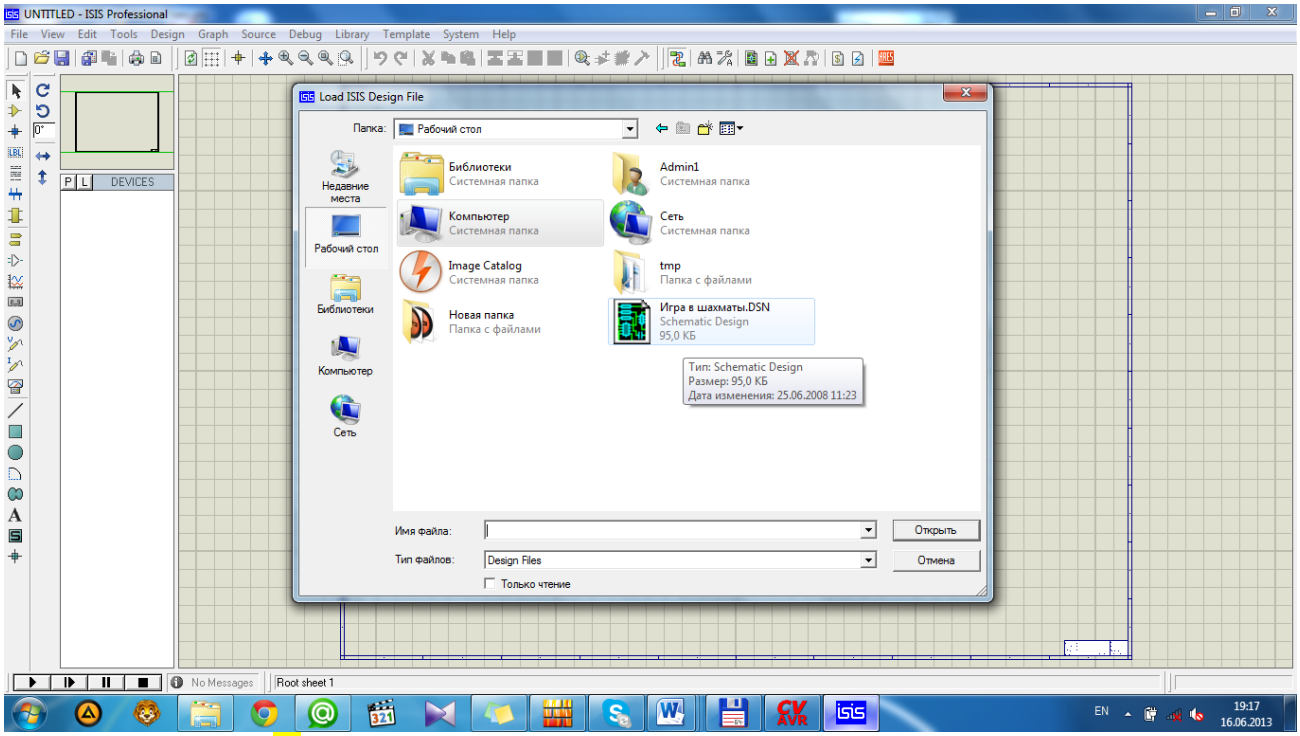


Рис.3.3.Открытие проекта в программе Proteus 7.7.

Вид программы в рабочем режиме.

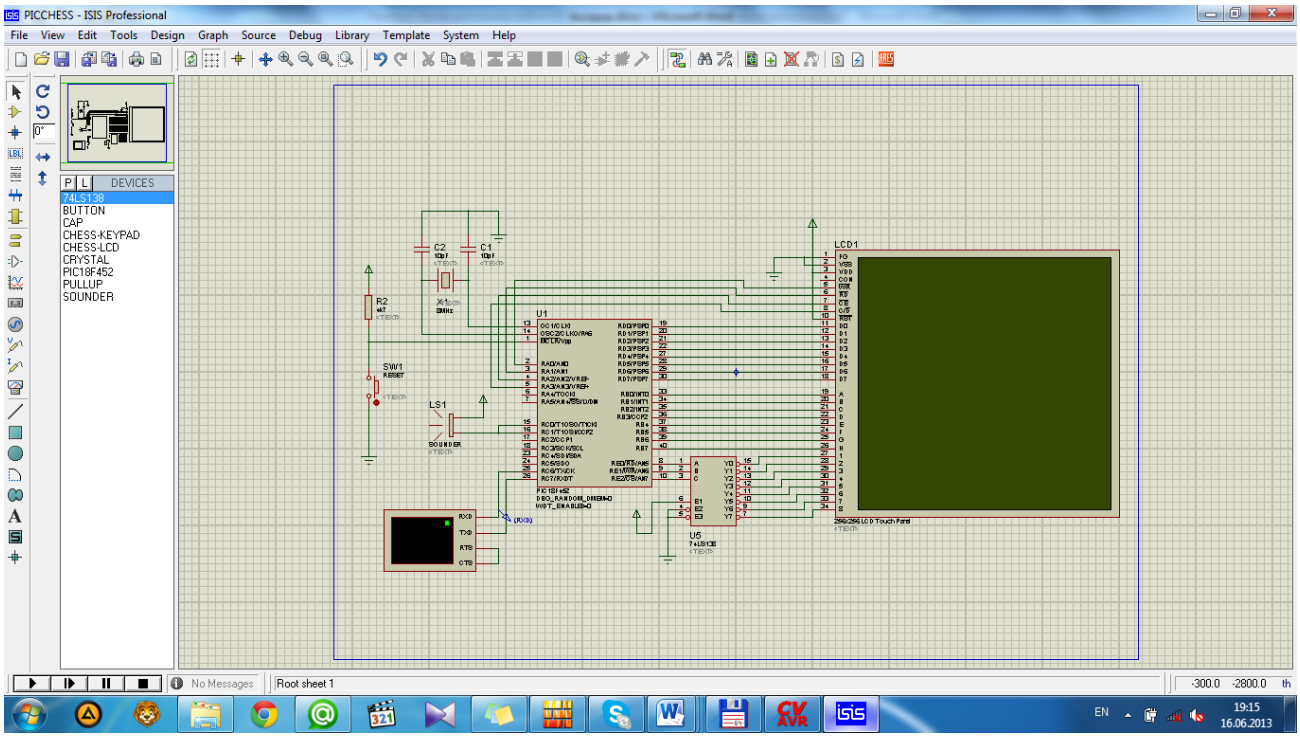


Рис.3.4.Рабочий режим проекта.

Вид проекта после запуска

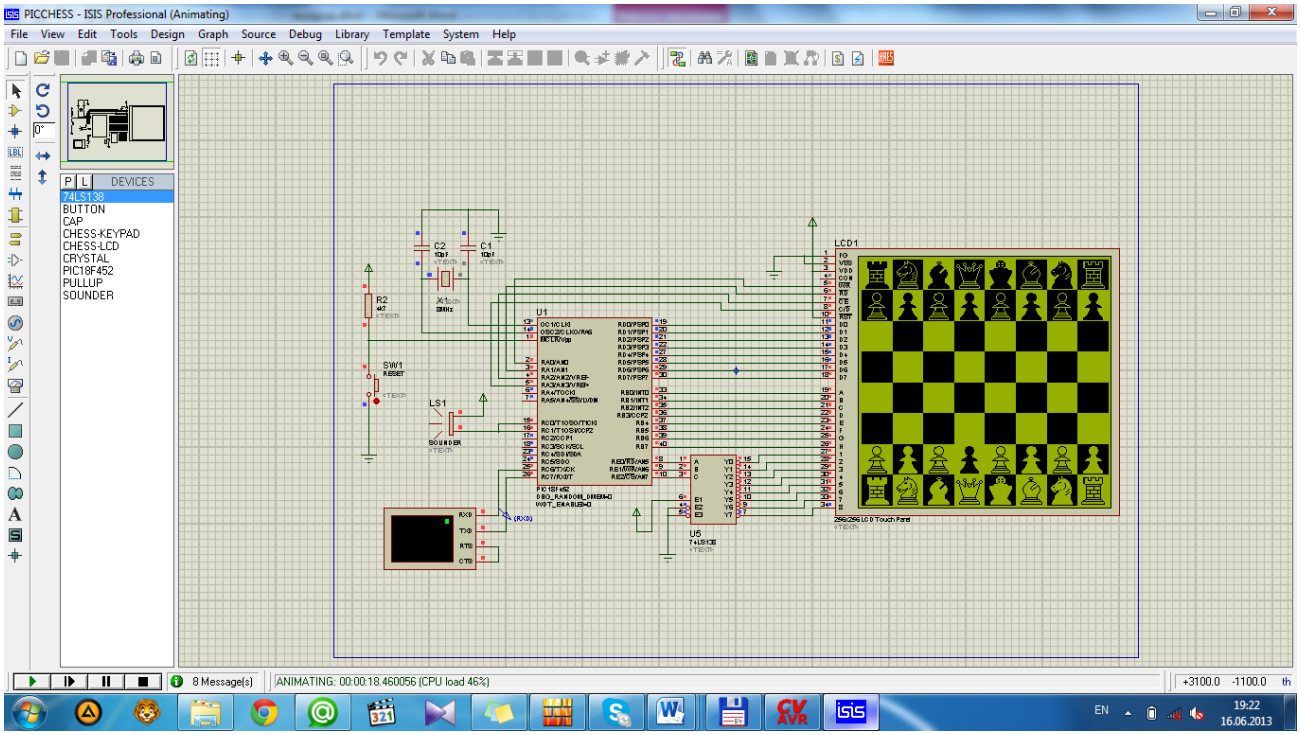


Рис. X Вид при запуске

3.3. Обеспечение безопастности жизнедеятельности при работе на компьюторе

Техника безопасности при работе на ПК.

1. К самостоятельной работе на ПК допускаются лица не моложе 18-ти лет, прошедшие медицинское освидетельствование, специальное обучение, инструктаж по охране труда на рабочем месте, изучившие "Руководство по эксплуатации" и усвоившие безопасные методы и приемы выполнения работы.

Персонал, допущенный к работе на ПК по наладке, эксплуатации PR-нию обязан:

· получить инструктаж по охране труда;

· ознакомиться с общими правилами эксплуатации и указаниями по безопасности труда, которые содержаться в "Руководстве по эксплуатации";

· познакомиться с предупреждающими записями на крышках, стенках, панелях блоков и устройств;

· познакомиться с правилами эксплуатации электрооборудования.

2. ПК должен подключаться к однофазной сети с нормальным напряжением 220 (120) В, частотой 50 (60) Гц и заземленной нейтрально. Заземляющие контакты розеток должны быть надежно соединены с контуром защитного заземления помещения. В помещении должен быть установлен автомат аварийного или рубильник общего отключения питания.

3. Запрещается самостоятельно производить ремонт ПК (его блоков), если это не входит в круг ваших обязанностей.

4. При эксплуатации ПК должны выполняться следующие требования, правила:

· не подключать и не отключать разъемы и кабели электрического питания при поданном напряжении сети;

· не оставлять ПК включенным без наблюдения;

· не оставлять ПК включенным во время грозы;

· по окончании работы отключить ПК от сети;

· устройства должны быть расположены на расстоянии 1 м от нагревательных приборов; рабочие места должны располагаться между собой на расстоянии не менее 1,5 метров;

· устройства не должны подвергаться воздействию прямых солнечных лучей; непрерывная продолжительность работы при вводе данных на ПК не должна превышать 4 часов при 8-часовом рабочем дне, через каждый час работы необходимо делать перерыв 5-10 минут, через 2 часа на 15 минут;в помещении, где расположена компьютерная техника, должен быть оборудован уголок пожаротушения.

Системный блок следует включать как можно реже (обычно включается в начале рабочего дня и выключается, выключается работа на нем — в конце дня). Для того, чтобы не выгорал экран и не расходовалась лишняя энергия, в компьютере предусмотрен специальный режим гашения экрана — через определенное время, если никто не работает на нем, т.е. нет обращения к клавиатуре или мыши, он выключается. Если монитор получает питание от системного блока, включая системный блок, включаем и монитор. Если соединение монитора и системного блока параллельно, то сначала необходимо включить монитор, потом системный блок. Выключать в обратной последовательности.

Экран монитора стеклянный, а потому и хрупкий, и поэтому надо обращаться с ним осторожно. Недопустимо попадание жидкости за заднюю часть экрана может замкнуть проводка, что выведет его из строя и может привести к возникновению пожара. В случае попадания жидкости следует отключить электропитание.

Защита от излучения расположена только на экране, поэтому, находясь прямо перед экраном, пользователь наиболее защищен от вредного воздействия излучения. На заднюю и боковые части монитора в целях экономии защиту не устанавливают. Следовательно, находясь сбоку или сзади монитора, можно получить максимально вредное воздействие.

При работе с клавиатурой стоит придерживаться следующих правил:

1) сильно не ударять по клавишам, это приводит к быстрой изнашиваемости прибора.

2) не распивать напитки над клавиатурой, так как попадание жидкости в нее приводит к короткому замыканию и выводит из строя клавиатуру, в случае попадания необходимо обесточить компьютер.

3) не кушать над клавиатурой бутерброды, семечки, так как крошки, попадающие в клавиатуру, нарушают ее работу.

4) при наличии защитной панели следует закрывать клавиатуру, тем самым, защищая ее от пыли.

Заключение

За время работы над дипломным проектом по теме «Разработка программы автоматизации расчета температурного режима помещений» были изучены теоретические основы разработки микропроцессорной системы.

Результатом дипломного проектирования является разработка прибора, предназначенного для автоматического регулирования температуры. Главная особенностью терморегуляторов - простота схем при существенно более широких, чем у распространенных аналоговых, функциональных возможностях, отсутствие необходимости регулировки и настройки при изготовлении и эксплуатации.

В выпускной квалификационной работе рассмотрены общие подходы к реализации микропроцессорной системы на базе микроконтроллера для терморегулятора помещений. В процессе написания квалификационной работы автором велась разработка реализации микропроцессорной системы на базе микроконтроллера PIC16F84A.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. И.А. Каримов «Мировой финансово-экономический кризис, пути и меры по его преодолению в условиях Узбекистана», Ташкент, Узбекистон, 2009
2. И.Рахманов, Г.Искадария, К.Худойкулов, «Первая медицинская помощь в чрезвычайных ситуациях». Ташкент, «Фан» 2005.
3. Белов А.В. Микроконтроллеры AVR в радиолюбительской практике – СП-б, Наука и техника, 2007 – 352с.
4. Проектирование цифровых устройств на однокристальных микроконтроллерах / В.В. Сташин [и др.]. – М.: Энергоатомиздат, 1990. – 224 с.
5. Евстифеев А.В. Микроконтроллеры Microchip: практическое руководство/А.В.Евстифеев. – М.: Горячая линия – Телеком, 2002. – 296 с.
6. Кравченко А.В. 10 практических устройств на AVR-микроконтроллерах. Книга 1 – М., Додэка –XX1, МК-Пресс, 2008 – 224с.
7. Трамперт В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров: Пер. с нем – К., МК-Пресс, 2006 – 208с.
8. Мортон Дж. Микроконтроллеры AVR. Вводный курс /Пер. с англ. – М., Додэка –XX1, 2006 – 272с.
9. Техническая документация на микроконтроллеры PIC16F84A компании Microchip Technology Incorporated . ООО «Микро -Чип», Москва, 2002.-184 с.
10. А.Кудратов, Т.Ганиев и др. «Безопасность жизнедеятельности». Ташкент, Алокачи, 2005.
11. Белов А.Б. Конструирование устройств на микроконтроллерах / Наука и Техника, 2005. - 255 с.

12. Предко М. Руководство по микроконтроллерам. Том 1. / Пер. с англ. под ред.И. И. Шагурина и С.Б. Лужанского - М.: Постмаркет, 2001. - 416 с.
13. Предко М. Руководство по микроконтроллерам. Том 2. / Пер. с англ. под ред.И. И. Шагурина и С.Б. Лужанского - М.: Постмаркет, 2001. - 488 с.
14. Вуд А. Микропроцессоры в вопросах и ответах. / Пер. с англ. под ред. Д.А. Поспелова. - М.: Энергоатомиздат. 1985. - 184 с.
15. Уильямс Г.Б. Отладка микропроцессорных систем: / Пер. с. англ. - М.: Энергоатомиздат, 1988. - 253с.
16. Угрюмов Е.П. Цифровая схемотехника. - Спб.: БВХ - Санкт-Петербург, 2000. - 528 с.
17. Алексенко А.Г., Шагурин И.И. Микросхемотехника. - М.: Радио и связь, 1990. - 496 с.
18. Бродин Б.В., Шагурин И.И. Микроконтроллеры: Справочник. - М.: ЭКОМ, 1999. - 395 с.
19. Программируемые логические ИМС на КМОП-структурах и их применение. / П.П. Мальцев, Н.И. Гарбузов, А.П. Шарапов, А.А. Кнышев. - М.: Энергоатомиздат, 1998. - 158 с.
20. Соловьев В.В., Васильев А.Г. Программируемые логические интегральные схемы и их применение. - Мн.: Беларуская наука, 1998. - 270 с.
21. Лаптев В. Цифровой измеритель температуры на базе AVR микроконтроллера и RC-цепочки. - Электронные компоненты, 2001. №2, с.46 - 49.
22. Научно- технический журнал «Схемотехника» №2, 2001–2002 гг.

Голубцов М.С., Кириченкова А.В. Микроконтроллеры AVR: от простого к сложному. Изд.2-е, испр. И доп. – М.: СОЛОН- Пресс, 2006. 304с.- (Серия «Биб-лиотека инженера»).

23. Китаев Ю.В. Основы программирования микроконтроллеров AT MEGA128 и 68HC908. Учебное пособие : СПб: СпбГУ ИТМО, 2007, 107с.

24. Температурные измерения. Справочник./ Геращенко О.А. Гордов А.Н., Еремина А.К., и др.; отрв. Ред. Геращенко О.А.; АН УССР Ин-т проблем энергосбережения. – Киев: Наук. думка, 1989г. 704 с.

Ссылки на использованные источники в Интернет

25. http://www.atmel.ru
26. http://www.telesys.users.ru
27. http://www.kulakov.ru
28. http://www.platan.ru
29. http://www.sensorsmag.com
30. http://www.ferrite.ru
31. http://pdfserv.maxim-ic.com/arpdf/DS18S20.pdf.
32. http://www.chip-dip.ru/product0/61922.aspx

Приложение 1

Процесс работы программы

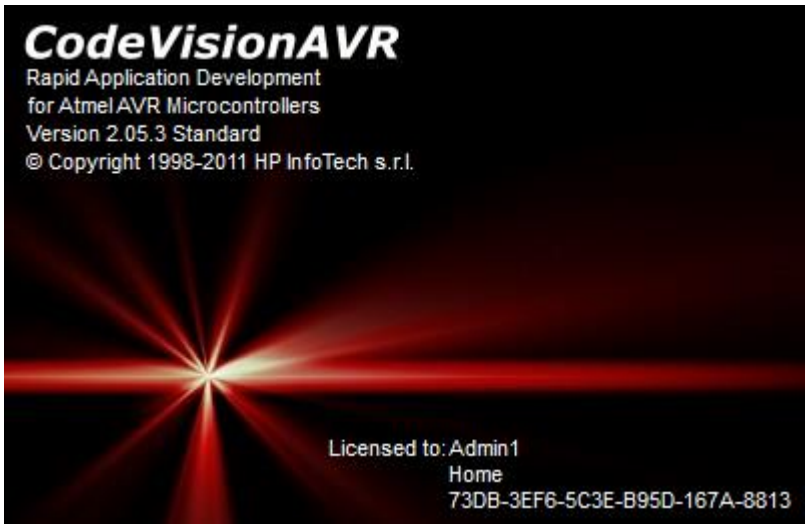


Рис 1.

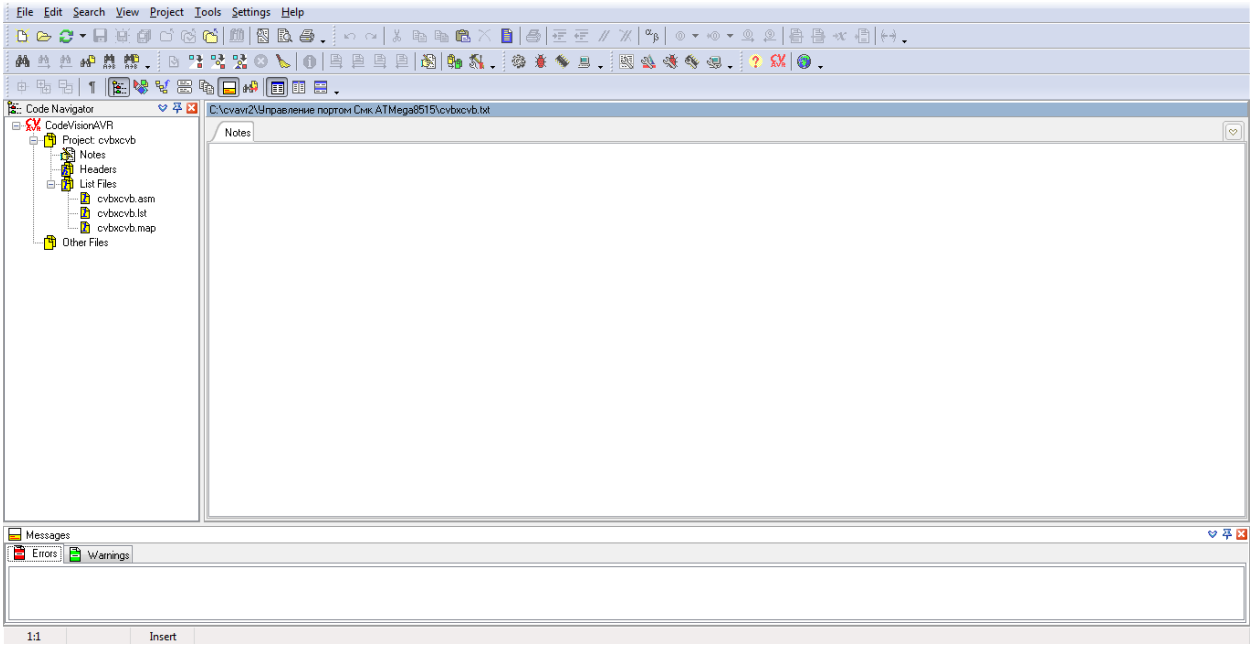
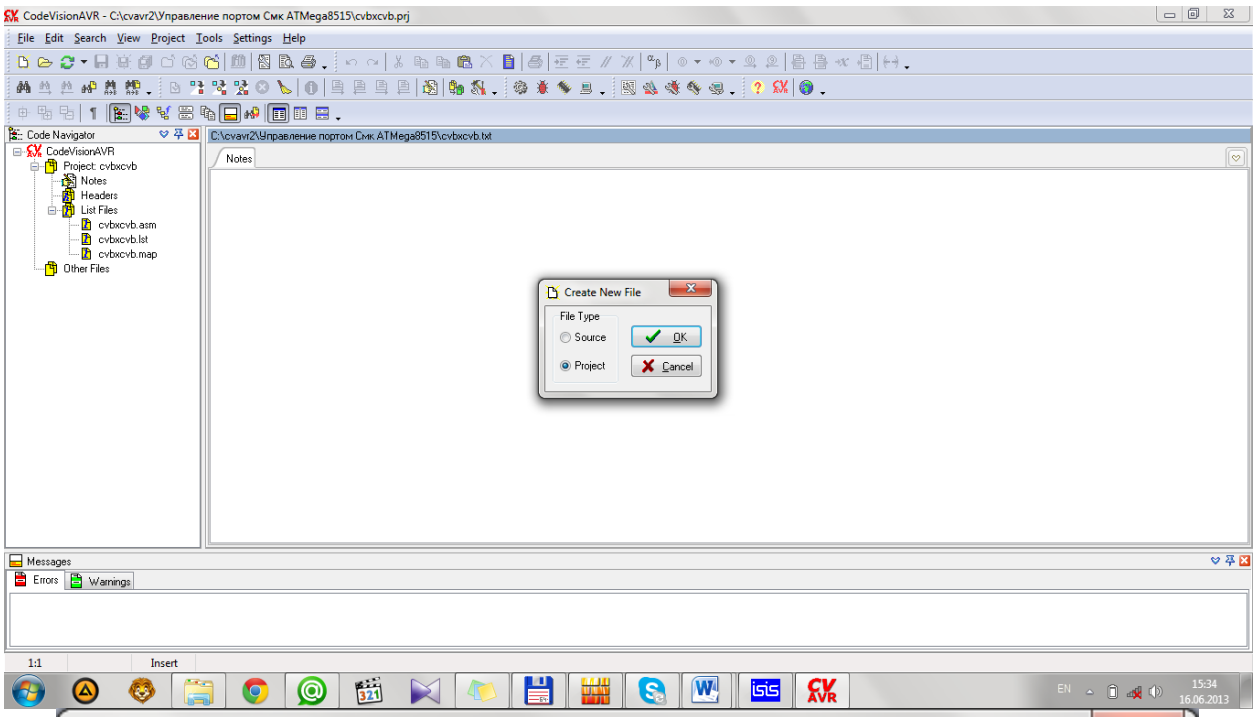
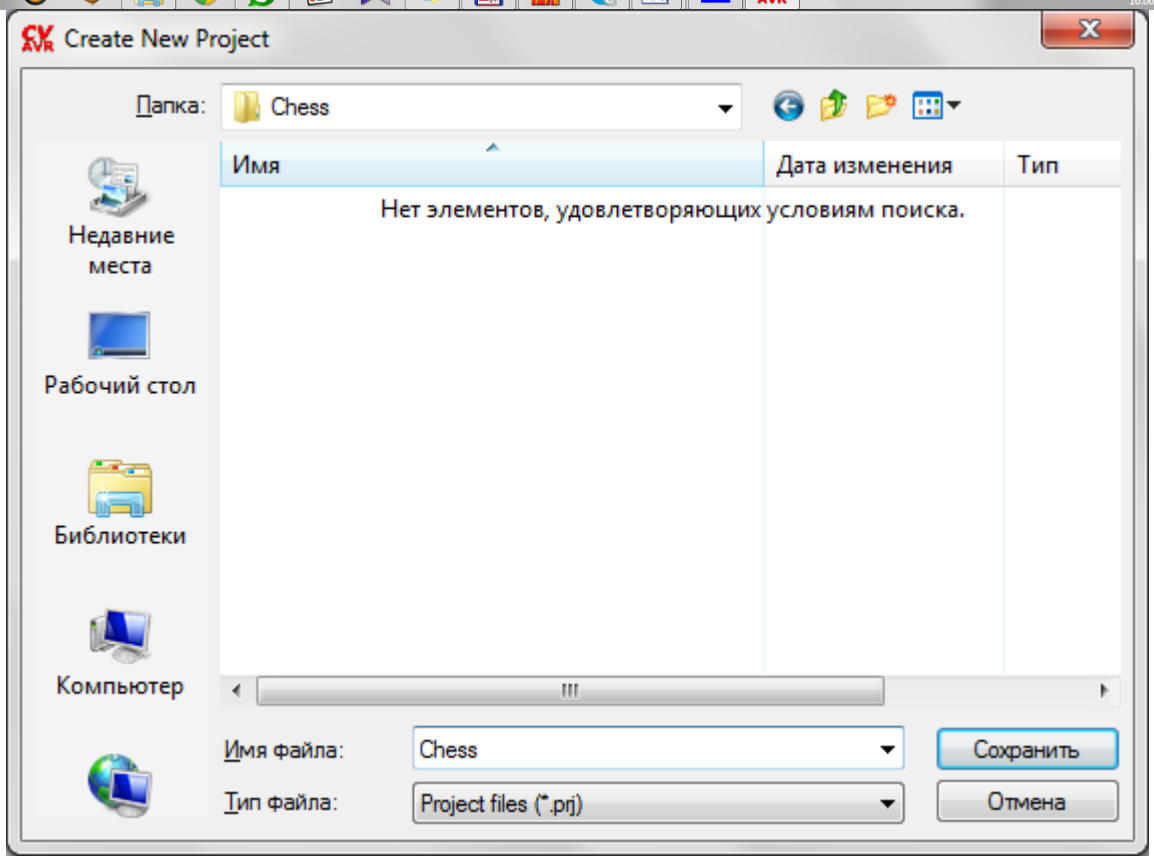
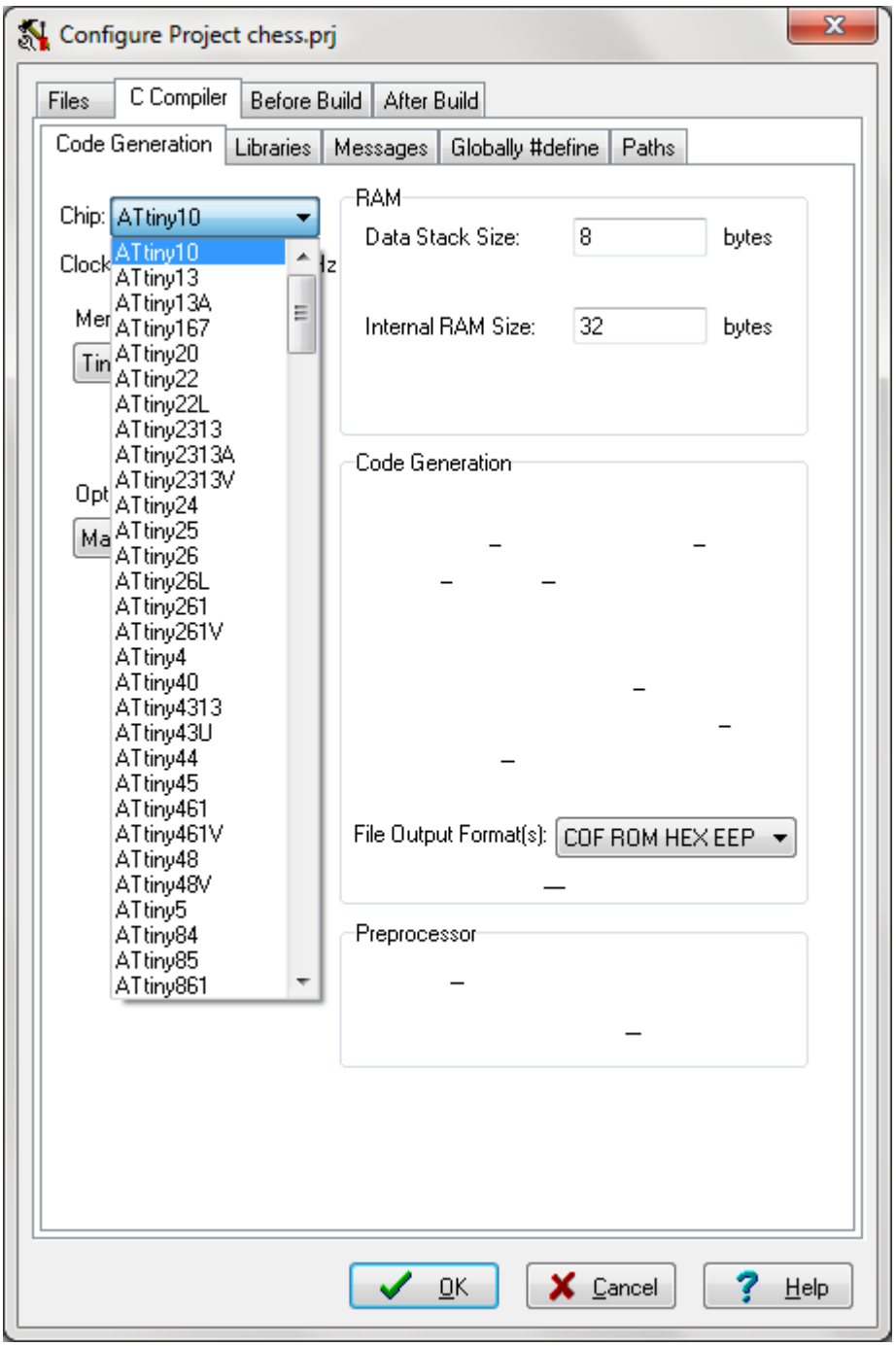
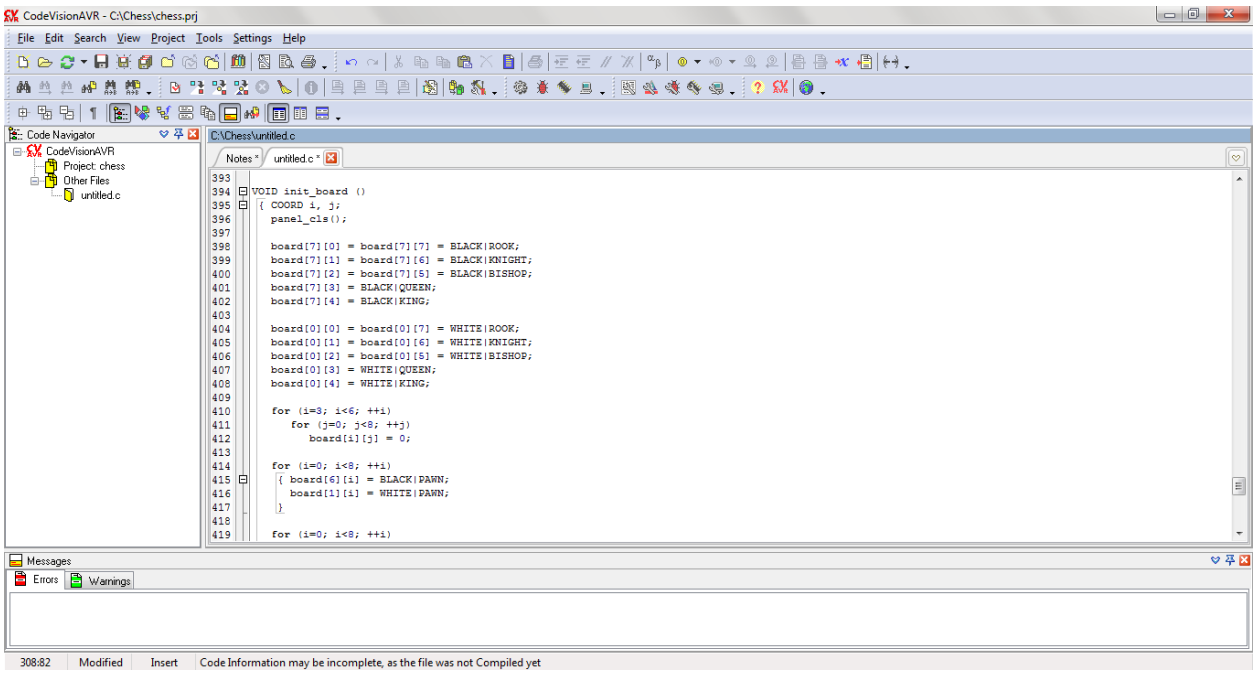


Рис 2. Создат Project

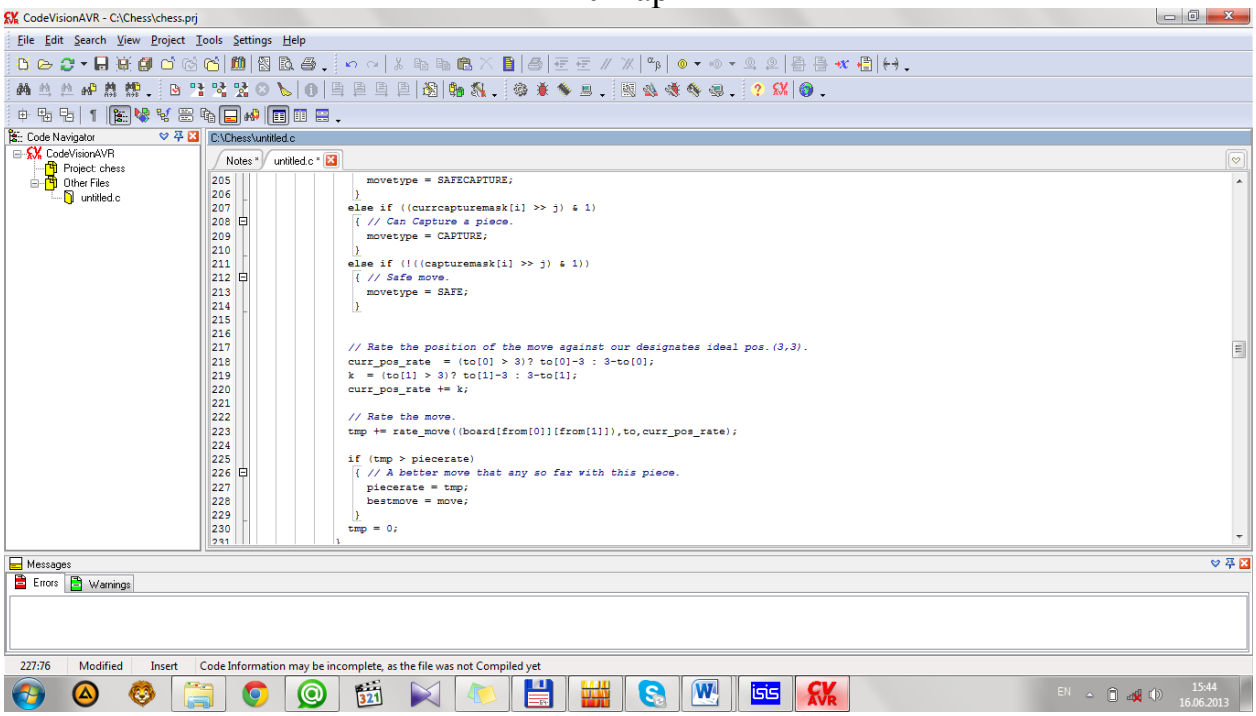

CodeVisionAVR - C:\cvavr2\Управление портом Смк ATMega8515\cvbxcvb.prj
File Edit Search View Project Tools Settings Help
Code Navigator
CodeVisionAVR
Project: cvbxcvb
Notes
Headers
List Files
cvbxcvb.asm
cvbxcvb.lst
cvbxcvb.map
Other Files
C:\cvavr2\Управление портом Смк ATMega8515\cvbxcvb.txt
Notes
Create New File
File Type
Source
Project
OK
Cancel
Messages
Errors
Warnings
1:1
Insert

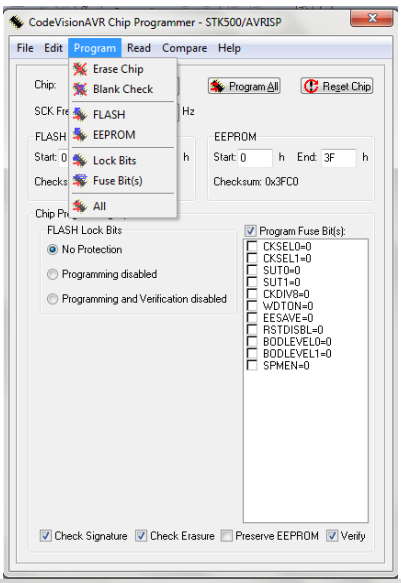

Create New Project
Папка:
Chess
Имя
Дата изменения
Тип
Нет элементов, удовлетворяющих условиям поиска.
Недавние места
Рабочий стол
Библиотеки
Компьютер
Имя файла:
Chess
Тип файла:
Project files (*.prj)
Сохранить
Отмена

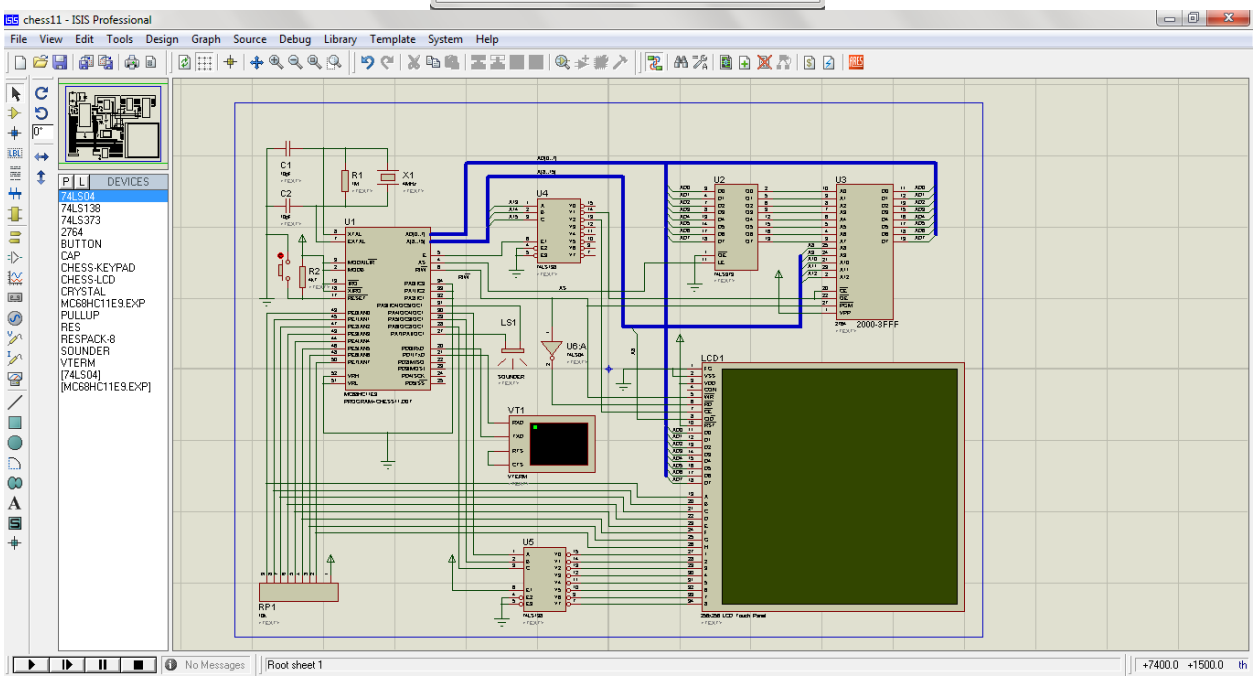

Configure Project chess.prj
Files
C Compiler
Before Build
After Build
Code Generation
Libraries
Messages
Globally #define
Paths
Chip:
ATtiny10
ATtiny10
ATtiny13
ATtiny13A
ATtiny167
ATtiny20
ATtiny22
ATtiny22L
ATtiny2313
ATtiny2313A
ATtiny2313V
ATtiny24
ATtiny25
ATtiny26
ATtiny26L
ATtiny261
ATtiny261V
ATtiny4
ATtiny40
ATtiny4313
ATtiny43U
ATtiny44
ATtiny45
ATtiny461
ATtiny461V
ATtiny48
ATtiny48V
ATtiny5
ATtiny84
ATtiny85
ATtiny861
RAM
Data Stack Size:
8
bytes
Internal RAM Size:
32
bytes
Code Generation
File Output Format(s):
COF ROM HEX EEP
Preprocessor
OK
Cancel
Help

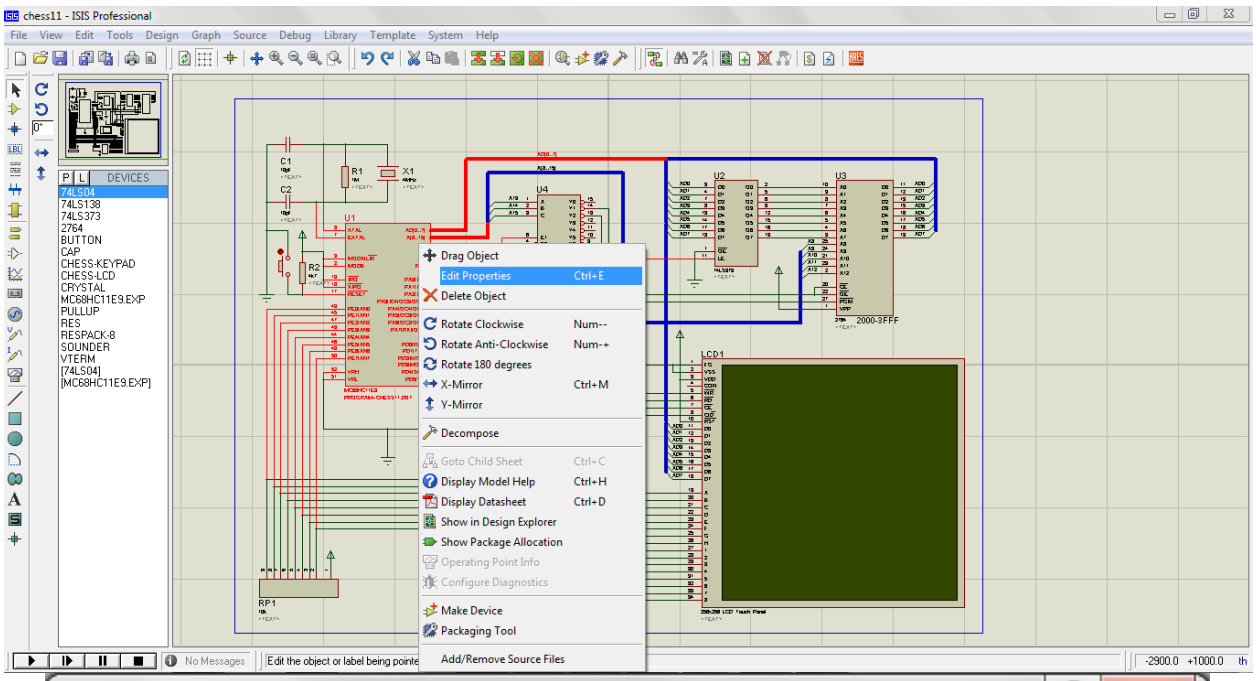

CodeVisionAVR - C:\Chess\chess.prj
File Edit Search View Project Tools Settings Help
Code Navigator
CodeVisionAVR
Project: chess
Other Files
untitled.c
C:\Chess\untitled.c
Notes *
untitled.c *
393
394 VOID init_board ()
395 { COORD i, j;
396 panel_cls();
397
398 board[7][0] = board[7][7] = BLACK|ROOK;
399 board[7][1] = board[7][6] = BLACK|KNIGHT;
400 board[7][2] = board[7][5] = BLACK|BISHOP;
401 board[7][3] = BLACK|QUEEN;
402 board[7][4] = BLACK|KING;
403
404 board[0][0] = board[0][7] = WHITE|ROOK;
405 board[0][1] = board[0][6] = WHITE|KNIGHT;
406 board[0][2] = board[0][5] = WHITE|BISHOP;
407 board[0][3] = WHITE|QUEEN;
408 board[0][4] = WHITE|KING;
409
410 for (i=3; i<6; ++i)
411 for (j=0; j<8; ++j)
412 board[i][j] = 0;
413
414 for (i=0; i<8; ++i)
415 { board[6][i] = BLACK|PAWN;
416 board[1][i] = WHITE|PAWN;
417 }
418
419 for (i=0; i<8; ++i)
Messages
Errors
Warnings
308:82
Modified
Insert
Code Information may be incomplete, as the file was not Compiled yet

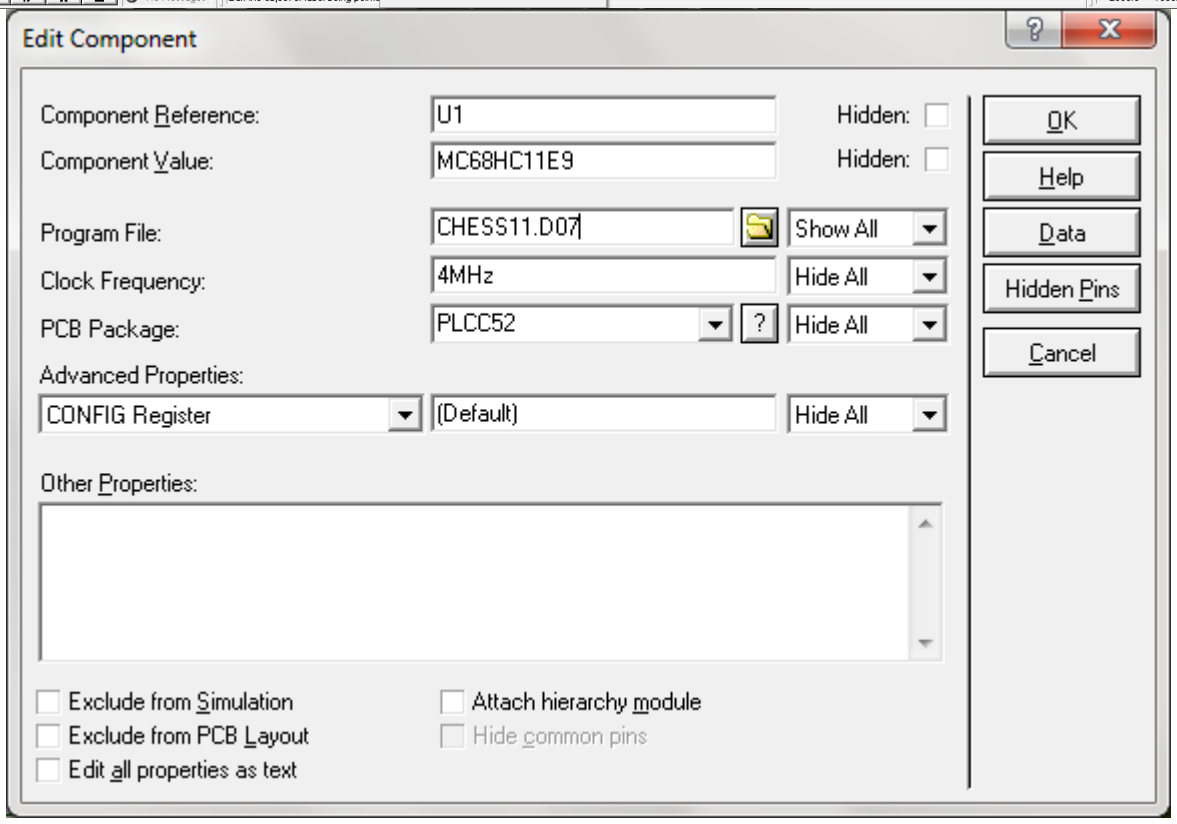
Умар

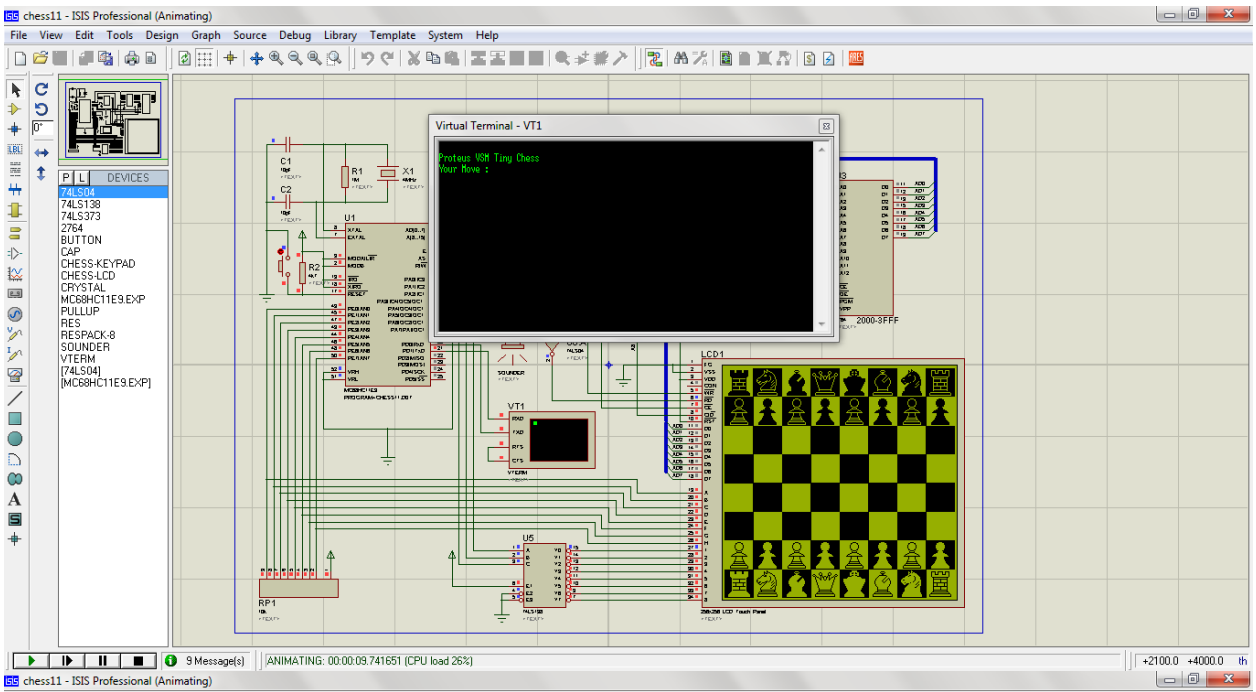

CodeVisionAVR - C:\Chess\chess.prj
File Edit Search View Project Tools Settings Help
Code Navigator
CodeVisionAVR
Project: chess
Other Files
untitled.c
C:\Chess\untitled.c
Notes *
untitled.c *
205 movetype = SAFECAPTURE;
206 }
207 else if ((currcapturemask[i] >> j) & 1)
208 { // Can Capture a piece.
209 movetype = CAPTURE;
210 }
211 else if (!((capturemask[i] >> j) & 1))
212 { // Safe move.
213 movetype = SAFE;
214 }
215
216
217 // Rate the position of the move against our designates ideal pos.(3,3).
218 curr_pos_rate = (to[0] > 3)? to[0]-3 : 3-to[0];
219 k = (to[1] > 3)? to[1]-3 : 3-to[1];
220 curr_pos_rate += k;
221
222 // Rate the move.
223 tmp += rate_move((board[from[0]][from[1]]),to,curr_pos_rate);
224
225 if (tmp > piecerate)
226 { // A better move that any so far with this piece.
227 piecerate = tmp;
228 bestmove = move;
229 }
230 tmp = 0;
Messages
Errors
Warnings
227:76
Modified
Insert
Code Information may be incomplete, as the file was not Compiled yet
EN
15:44
16.06.2013

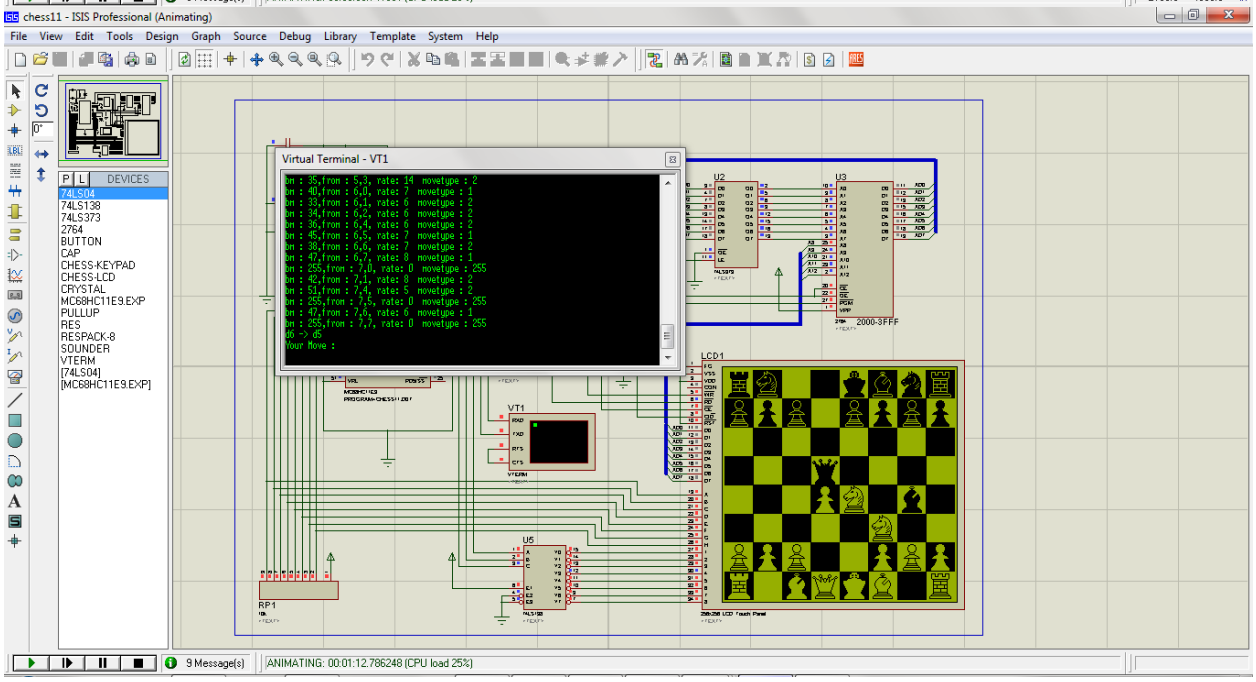

CodeVisionAVR Chip Programmer - STK500/AVRISP
File Edit Program Read Compare Help
Erase Chip
Blank Check
FLASH
EEPROM
Lock Bits
Fuse Bit(s)
All
Program All
Reset Chip
EEPROM
Start: 0 h End: 3F h
Checksum: 0x3FC0
FLASH Lock Bits
No Protection
Programming disabled
Programming and Verification disabled
Program Fuse Bit(s):
CKSEL0=0
CKSEL1=0
SUT0=0
SUT1=0
CKDIV8=0
WDTON=0
EESAVE=0
BODLEVEL0=0
BODLEVEL1=0
SPMEN=0
Check Signature
Check Erasure
Preserve EEPROM
Verify

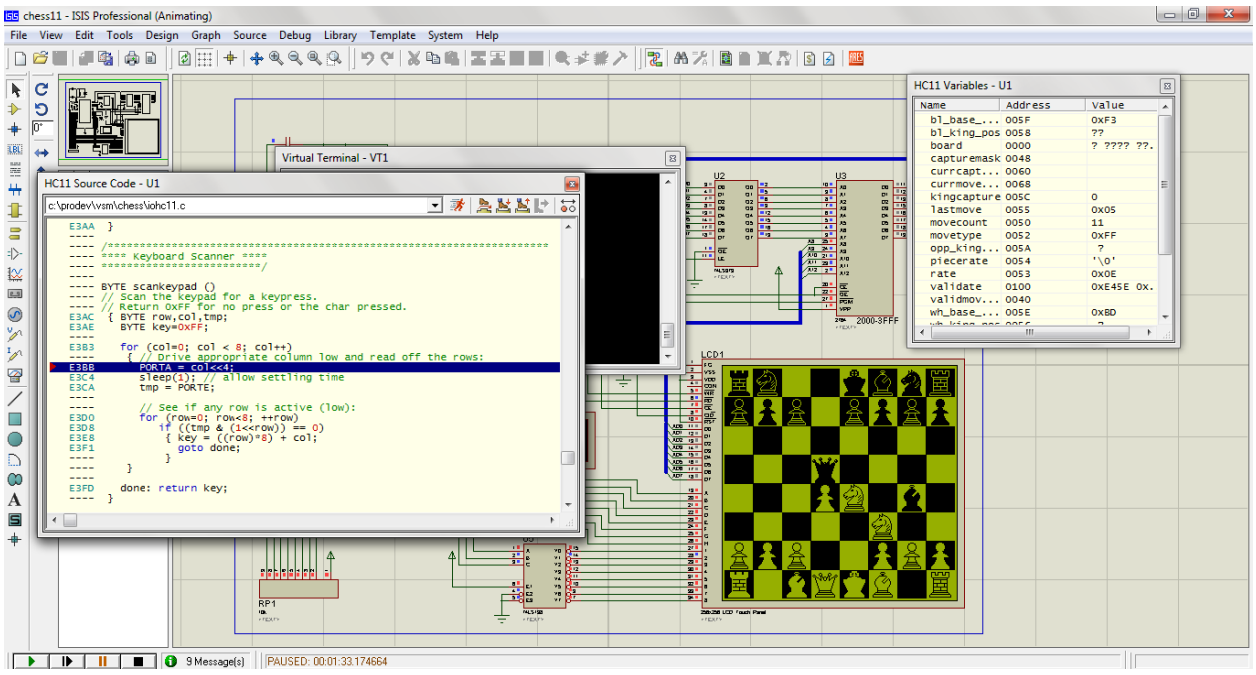

chess11 - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
74LS04
74LS138
74LS373
2764
BUTTON
CAP
CHESS-KEYPAD
CHESS-LCD
CRYSTAL
MC68HC11E9.EXP
PULLUP
RES
RESPACK-8
SOUNDER
VTERM
[74LS04]
[MC68HC11E9.EXP]
No Messages
Root sheet 1


chess11 - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
74LS04
74LS138
74LS373
2764
BUTTON
CAP
CHESS-KEYPAD
CHESS-LCD
CRYSTAL
MC68HC11E9.EXP
PULLUP
RES
RESPACK-8
SOUNDER
VTERM
[74LS04]
[MC68HC11E9.EXP]
Drag Object
Edit Properties Ctrl+E
Delete Object
Rotate Clockwise Num--
Rotate Anti-Clockwise Num-+
Rotate 180 degrees
X-Mirror Ctrl+M
Y-Mirror
Decompose
Goto Child Sheet Ctrl+C
Display Model Help Ctrl+H
Display Datasheet Ctrl+D
Show in Design Explorer
Show Package Allocation
Operating Point Info
Configure Diagnostics
Make Device
Packaging Tool
Add/Remove Source Files
No Messages
Edit the object or label being pointe
-2900.0 +1000.0


Edit Component
Component Reference:
U1
Hidden:
Component Value:
MC68HC11E9
Hidden:
Program File:
CHESS11.D07
Show All
Clock Frequency:
4MHz
Hide All
PCB Package:
PLCC52
?
Hide All
Advanced Properties:
CONFIG Register
(Default)
Hide All
Other Properties:
Exclude from Simulation
Exclude from PCB Layout
Edit all properties as text
Attach hierarchy module
Hide common pins
OK
Help
Data
Hidden Pins
Cancel


chess11 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
Virtual Terminal - VT1
Proteus VSM Tiny Chess
Your Move :
DEVICES
74LS04
74LS138
74LS373
2764
BUTTON
CAP
CHESS-KEYPAD
CHESS-LCD
CRYSTAL
MC68HC11E9.EXP
PULLUP
RES
RESPACK-8
SOUNDER
VTERM
[74LS04]
[MC68HC11E9.EXP]
LCD1
VT1
U5
RP1
9 Message(s)
ANIMATING: 00:00:09.741651 (CPU load 26%)


chess11 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
Virtual Terminal - VT1
Your Move :
DEVICES
74LS04
74LS138
74LS373
2764
BUTTON
CAP
CHESS-KEYPAD
CHESS-LCD
CRYSTAL
MC68HC11E9.EXP
PULLUP
RES
RESPACK-8
SOUNDER
VTERM
[74LS04]
[MC68HC11E9.EXP]
LCD1
VT1
U5
RP1
9 Message(s)
ANIMATING: 00:01:12.786248 (CPU load 25%)


chess11 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
Virtual Terminal - VT1
HC11 Source Code - U1
c:\prodev\vsm\chess\iohc11.c
E3AA }
---- /***
---- **** Keyboard Scanner ****
---- ************************/
---- BYTE scankeypad ()
---- // Scan the keypad for a keypress.
---- // Return 0xFF for no press or the char pressed.
E3AC { BYTE row,col,tmp;
E3AE BYTE key=0xFF;
E3B3 for (col=0; col < 8; col++)
{ // Drive appropriate column low and read off the rows:
E3BB PORTA = col<<4;
E3C4 sleep(1); // allow settling time
E3CA tmp = PORTE;
E3D0 // See if any row is active (low):
E3D8 for (row=0; row<8; ++row)
E3E8 if ((tmp & (1<<row)) == 0)
E3F1 { key = ((row)*8) + col;
goto done;
}
}
E3FD done: return key;
}
HC11 Variables - U1
Name Address Value
bl_base_... 005F 0xF3
bl_king_pos 0058 ??
board 0000 ? ???? ??.
capturemask 0048
currcapt... 0060
currmove... 0068
kingcapture 005C 0
lastmove 0055 0x05
movecount 0050 11
movetype 0052 0xFF
opp_king... 005A ?
piecerate 0054 '\0'
rate 0053 0x0E
validate 0100 0xE45E 0x.
validmov... 0040
wh_base_... 005E 0xBD
U2
U3
LCD1
RP1
9 Message(s)
PAUSED: 00:01:33.174664

Приложение 2

Код программа

```
#include <stdlib.h>
#include <stdio.h>
#include "chess.h"
// General Variables.
ZPAGE BOARD board;
ZPAGE MASK validmovemask;
ZPAGE MASK capturemask;
ZPAGE INT movecount;
ZPAGE BYTE movetype;
ZPAGE BYTE rate;
ZPAGE BYTE piecerate;
ZPAGE BYTE lastmove;
// Check/Checkmate.
ZPAGE LOC wh_king_pos;
ZPAGE LOC bl_king_pos;
ZPAGE LOC opp_king_pos;
ZPAGE BOOL kingcapture;
//Castling.
ZPAGE BYTE wh_base_cont;
ZPAGE BYTE bl_base_cont;
ZPAGE MASK currcapturemask;
ZPAGE MASK currmovemask;
FUNCPTR validate[7] =
{&val_empty,&val_pawn,&val_rook,&val_knight,&val_bishop,&val_queen,&val
_king};
  extern void sleep (int);
main (void )
// Initialise the board and go to the
// designated play control function.
 { movecount   = 1;
   kingcapture = FALSE;
   movetype = NO_MOVE;
   panel_init();   init_board();   cpuplay();
 }
 VOID cpuplay()
// Control Function for a single player game.
 { LOC from, to, cpufrom,cputo;
   PIECE p;
   BYTE i = 0, j = 0;
   BYTE movefrom,bestfrom,moveto,bestto;
```

```
  BYTE illegal = FALSE;
  while (TRUE)
   { // User Move (White).
     if ((movecount+1)%2 == 0)
      { printf("Your Move :\n");
       if (!illegal)
         sound_yourmove();
       else
         illegal = FALSE;
       // Get the move.
       while (!panel_getmove(from, to))  ;
       p = board[from[0]][from[1]];
       //getindex(to[0],to[1]);
       // Get valid moves for given piece.
       validate[piecetype(p)] (from,WHITE);
         // Next, Verify that we can make the move and
       // deal with the King/Check logic.
       if (is_valid_move(to[0],to[1]) &&
test_singlemove(from,to,board[from[0]][from[1]]))
        { draw_move(from,to,p);
          printf("%c%c -> %c%c\n", from[1]+'a', from[0]+'1',to[1]+'a', to[0]+'1');
          movecount++;
        }
       else
        { printf ("\nInvalid move for specified piece\n");
          sound_illegal();     illegal = TRUE;
        }
       // Finally, Clear our masks and reset bools.
       for (i = 0; i < 8;++i)
        {// Clear our mask.    validmovemask[i] = 0;  capturemask[i] = 0;        }
      }
       // CPU move.
     else if ((movecount+1)%2 == 1)
      { bestfrom = bestto = NO_MOVE;
       rate = piecerate = 0;
       printf ("My Move...Thinking:\n");
         // For every black piece...
       for (i = 0; i < 8;++i)
        { for (j = 0;j < 8;++j)
          { if (board[i][j] & 0x08)
            { // Get possible moves.
              cpufrom[0] = i;    cpufrom[1] = j;   p = board[i][j];
              validate[piecetype(p)](cpufrom,BLACK);
              // Get best move for that piece.
              moveto   = getbestpiecemove(cpufrom);
```

```
                movefrom = getindex(cpufrom[0],cpufrom[1]);
                if (((board[cpufrom[0]][cpufrom[1]]&0x07) == lastmove) &&
(movetype < 3))
                 { // Prevent consistent moving of the same piece.
piecerate -= 3;          }
                  if (piecerate > rate)
                  { rate     = piecerate;     bestfrom = movefrom;     bestto   = moveto;
                  }
                 moveto = movefrom = -1;
               }
              movetype = NO_MOVE;
            }
          }
        if (rate != 0)
         { // We can move.
           getcoord (cpufrom[0],cpufrom[1],bestfrom);
           getcoord (cputo[0],cputo[1],bestto);
           p = board[(cpufrom[0])][(cpufrom[1])];
           draw_move(cpufrom,cputo,p);
           lastmove = (((p&0x07) != PAWN) && ((p&0x07) != KING)) ? p&0x07 : 0;
           printf("%c%c -> %c%c\n", cpufrom[1]+'a', cpufrom[0]+'1',cputo[1]+'a',
cputo[0]+'1');
           movecount++;
         }
        else
         { //Checkmate.
           printf("Another Crushing Defeat for the CPU - You Win\n\n");
           movecount++;
         }

       }

      else
       { // Trying to move CPU piece.
         printf("\nNice Try Amigo\n");
       }
    }
 }
INT getbestpiecemove(LOC from)
// This function tests all the valid moves for
// the specified piece and returns the index
// of the best move.
 { BYTE i,j,k;
   BYTE tmp;   LOC to;
   BYTE move,bestmove,curr_pos_rate;
```

```
bestmove = NO_MOVE;   piecerate = tmp = 0;
for (i = 0; i < 8;++i)
 { currmovemask[i]    = validmovemask[i];
   currcapturemask[i] = capturemask[i];
   validmovemask[i] = 0;
   capturemask[i] = 0;
 }
for (i = 0; i < 8; ++i)
 { for (j = 0; j < 8; ++j)
    { if (currmovemask[i] & 0x01)
       { // Test a Valid Move.
         to[0] = i;      to[1] = j;
         if (test_singlemove(from,to,board[from[0]][from[1]]))
          {    move = getindex(to[0],to[1]);
            movetype = NORMAL;
            if ((validmovemask[from[0]] >> from[1]) & 1)
             { // Piece can be taken - add some weight to escape.
               tmp = (board[from[0]][from[1]] & 0x07)/2;
             }
            if((!((capturemask[i] >> j) & 1)) && ((currcapturemask[i] >> j) & 1))
             { // Best Case Scenario - We can Capture a piece and cannot be
               // captured in turn.
               movetype = SAFECAPTURE;
             }
            else if ((currcapturemask[i] >> j) & 1)
             { // Can Capture a piece.
               movetype = CAPTURE;
             }
            else if (!((capturemask[i] >> j) & 1))
             { // Safe move.
               movetype = SAFE;
             }

            // Rate the position of the move against our designates ideal pos.(3,3).
            curr_pos_rate  = (to[0] > 3)? to[0]-3 : 3-to[0];
            k  = (to[1] > 3)? to[1]-3 : 3-to[1];
            curr_pos_rate += k;
            // Rate the move.
            tmp += rate_move((board[from[0]][from[1]]),to,curr_pos_rate);

            if (tmp > piecerate)
             { // A better move that any so far with this piece.
               piecerate = tmp;
               bestmove = move;
             }
```

```
            tmp = 0;
           }
          for (k = 0; k < 8; ++k)
           { // Reset Masks.
            validmovemask[k] = 0;       capturemask[k]   = 0;
           }
         }
        currmovemask[i] = currmovemask[i] >> 1;
       }
    }
  if (bestmove != NO_MOVE) printf("bm : %d,from : %d,%d, rate: %d  movetype
: %d\n",bestmove,from[0],from[1],piecerate,movetype);
  return bestmove;
 }
BOOL test_singlemove (LOC from,LOC to,PIECE p)
{ INT i = 0,j = 0;
   LOC opp_pos;
   PIECE opp_piece,tmpdest;
   if ((p & 0x07) == KING) updatekingpos(to,(p & 0x08));
   tmpdest = board[to[0]][to[1]];
   board[from[0]][from[1]] = EMPTY;
   board[to[0]][to[1]] = p;


   // Test our opponents pieces to see if any can capture our king.
   kingcapture = FALSE;
   i = j = 0;
   while ((i < 8) && (!kingcapture))
    { // scan rows.
      while ((j < 8) && (!kingcapture))
       { // scan columns.
         if (is_opp_piece(i,j,(p&0x08)))
          { // Opponents Piece - Get valid moves.
            opp_pos[0] = i, opp_pos[1] = j,opp_piece = board[i][j];
            validate[piecetype(opp_piece)] (opp_pos,(opp_piece & 0x08));
          }
         j++;
       }
      j = 0;
      i++;
    }
   // Finished Testing - Restore our internals.
   board[to[0]][to[1]] = tmpdest;
   board[from[0]][from[1]] = p;
   if ((p & 0x07) == KING) updatekingpos(from,(p & 0x08));
```

```
  return (kingcapture) ? FALSE : TRUE;
 }
INT rate_move(PIECE piece, LOC to,BYTE posrate)
{ BYTE tmp, value, rating;

  value = ((piece & 0x07)== KING) ? 0 : (piece&0x07);

  switch (movetype)
   { case NORMAL     : // Move low value pieces if possible.
                       rating = (6-value) + posrate/2;
                       break;
     case SAFE       : // Manouver better pieces to better positions.
                       rating = (value)+((8-posrate));
                       break;
     case SAFECAPTURE: // Capture safely if possible.
                       rating = 14;
                       break;
     case CAPTURE    : // Take an opponents piece here if it is better than our own.
                       // Values are arbitrary.
                       tmp = (board[to[0]][to[1]]&0x07)-(value);
                       rating = (tmp < 249) ? 12 : 4;
                       break;
     default         : rating = 0;
                       break;
   }
  return rating;
 }
VOID draw_move (LOC from, LOC to,PIECE p)
// Update the panel with a move.
 { BYTE *basemask,baserow;
  basemask = (p & 0x08) ? &bl_base_cont : &wh_base_cont;
  baserow = (p & 0x08) ? 7 : 0;

  if (piecetype(p) == KING)
   { // King Move.
     updatekingpos(to,(p&0x08));
     panel_draw(from[0],from[1],EMPTY);
     panel_draw(to[0],to[1],p);
     board[from[0]][from[1]] = EMPTY;
     board[to[0]][to[1]] = p;
     if (((to[1] - from[1]) != 1) && ((from[1] - to[1]) != 1))
      { // Castle Move.
        if (((*basemask) & 0x1F)== 0x11)
         { panel_draw(to[0],0,EMPTY);
           panel_draw(to[0],3,((p&0x08)|ROOK));
```

```
          board[to[0]][0] = EMPTY;
          board[to[0]][3] = ((p&0x08)|ROOK);
          printf("Queenside Castle\n");
         }
       else if (((*basemask) & 0xF0) == 0x90)
        { panel_draw(to[0],7,EMPTY);
          panel_draw(to[0],5,((p&0x08)|ROOK));
          board[to[0]][7] = EMPTY;
          board[to[0]][5] = ((p&0x08)|ROOK);
          printf("Kingside Castle\n");
         }
      }
   }
  else if (((p == 0x09) && (to[0] == 0)) || ((p == 1) && (to[0] == 7)))
   {// Queening a pawn.
    panel_draw(from[0],from[1], EMPTY);
    panel_draw(to[0],to[1], QUEEN|(p&0x08));
    board[from[0]][from[1]] = EMPTY;
    board[to[0]][to[1]] = QUEEN |(p&0x08);
   }
  else
   { // Normal move.
     panel_draw(from[0], from[1], EMPTY);
     panel_draw(to[0], to[1], p);

     if (board[to[0]][to[1]] != EMPTY)
        sound_capture();

     board[from[0]][from[1]] = EMPTY;
     board[to[0]][to[1]] = p;
   }

  if ((from[0] == baserow) && ((*basemask) & (1 << from[1])))
   { // Reflect any changes in the base row. After a Castle this
     // becomes superflous.
     (*basemask) &= ~(1 << from[1]);
   }
 }

VOID updatekingpos(LOC to, BOOL piececolour)
 // Update our record of the kings position.
 { if (!piececolour)
   { wh_king_pos[0] = to[0];
     wh_king_pos[1] = to[1];
   }
```

```
  else
   { bl_king_pos[0] = to[0];
     bl_king_pos[1] = to[1];
   }
 }

VOID init_board ()
 { COORD i, j;
   panel_cls();

   board[7][0] = board[7][7] = BLACK|ROOK;
   board[7][1] = board[7][6] = BLACK|KNIGHT;
   board[7][2] = board[7][5] = BLACK|BISHOP;
   board[7][3] = BLACK|QUEEN;
   board[7][4] = BLACK|KING;

   board[0][0] = board[0][7] = WHITE|ROOK;
   board[0][1] = board[0][6] = WHITE|KNIGHT;
   board[0][2] = board[0][5] = WHITE|BISHOP;
   board[0][3] = WHITE|QUEEN;
   board[0][4] = WHITE|KING;

   for (i=3; i<6; ++i)
      for (j=0; j<8; ++j)
         board[i][j] = 0;

   for (i=0; i<8; ++i)
    { board[6][i] = BLACK|PAWN;
      board[1][i] = WHITE|PAWN;
    }
     for (i=0; i<8; ++i)
      for (j=0; j<8; ++j)
         if (board[i][j] != EMPTY)
            panel_draw(i, j, board[i][j]);
wh_king_pos[0] = 0;  wh_king_pos[1] = 4;  bl_king_pos[0] = 7; bl_king_pos[1] =
4;
   wh_base_cont = bl_base_cont = 0xFF;
 }
```