МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение высшего профессионального образования
“ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Лабораторная работа №1

Разработка прикладного программного обеспечения для микропроцессорных систем на основе микроконтроллера (Быстрый старт)

Учебное пособие к выполнению лабораторных работ по дисциплине «Микропроцессорные системы» для студентов ФТИ специальности 140306.

Томск – 2010

УДК 681.322

Горюнов А.Г. Ливенцов С.Н. Разработка прикладного программного обеспечения для микропроцессорных систем на основе микроконтроллера (Быстрый старт): Учеб. пособие.

Учебное пособие посвящено быстрому освоению студентом цикла разработки приложений для микроконтроллеров. Пособие состоит из двух основных глав. Первая глава содержит введение в интегрированную среду разработки приложений для микроконтроллеров Keil Software, а вторая - “Быстрый старт” – обычный приём разработчиков современных программных средств. Данное учебное пособие ориентировано на курс лабораторных работ с использованием учебно-лабораторных стендов SDK-1-1.

Пособие подготовлено на кафедре «Электроника и автоматика физических установок» ТПУ и предназначена для студентов очного обучения специальности 140306.

Учебное пособие рассмотрено и рекомендовано
к изданию методическим семинаром кафедры электроники и автоматики
физических установок "___"_____ 2004г.

Зав. кафедрой,
д-р. техн. наук _____ С. Н. Ливенцов

СОДЕРЖАНИЕ

1 Цель работы

Изучение инструментальных средств и интегрированной среды разработки программного обеспечения для микроконтроллеров, а также изучение по методу “Быстрый старт” этапов технологии разработки и отладки программ.

2 Содержание работы

1. Введение в Keil uVision [1].
2. Быстрый старт.

3 Инструментальные средства Keil Software

Инструментальные средства *Keil Software* включают C и EC (Embedded C) компиляторы, ассемблеры, отладчики и симуляторы, интегрированные среды разработки и оценочные платы. *Keil Software* разрабатывает и производит средства разработки для следующих промышленных стандартов:

- *Infineon C16x/XC16x и ST-Micro ST10/Super10*
- *ARM7 и ARM7TDMI*
- *8051 Classic и Extended*
- *Philips LPC*
- *251 Atmel, Intel и Sanyo*

Keil Software поддерживает все разновидности вышеуказанных микроконтроллеров и все стадии разработки приложения: создание исходного файла на C / EC или ассемблере, трансляцию, исправление ошибок, линкование объектных файлов, тестирование приложения.

В данном курсе лабораторных работ мы будем использовать средства разработки для микроконтроллеров 8051.

3.1 Средства разработки микроконтроллерного семейства MCS51 (8051 Development tools)

Keil software поставляет следующие средства разработки:

- A51 Assembler Kit содержит A51 Assembler, 8051 Utilities и µVision IDE
- CA51 Compiler/Assembler Kit разработан для проектов с классическими МК 8051, содержит все компоненты A51 плюс C Compiler Cx51 и Code Banking Linker, который поддерживает до 32 банков кода (code banks), что позволяет выходить за пределы адресного пространства 64 KByte
- DK51 Developers Kit включает все компоненты CA51 плюс µVision Debugger, поддерживающий полную симуляцию устройств и target monitor.
- PK51 Professional Developers Kit содержит все компоненты DK51 плюс PK51 Extentions: поддержка расширенной памяти (до 16MB), variable banking, ОС реального времени RtxTiny2, оптимизированное линкование, внутрисхемный отладчик ISD51, поддержку Philips MX и contiguous mode Dallas 390.

Keil Software поддерживает все стадии разработки приложения: создание исходного файла на C или Ассемблере, трансляцию, исправление ошибок, линкование объектных файлов, тестирование приложения.

Средства разработки (CA51, DK51, PK51) для классических 8051 микроконтроллеров (с поддержкой до 32 банков по 64 Кб памяти программ) содержат:

- **C51 Compiler** – компилятор C;
- **A51 Macro Assembler** – макроассемблер;
- **BL51 Lincer/Locater** – динамический загрузчик/компоновщик.

Средства разработки (CA51, DK51, PK51) для классических и расширенных 8051 микроконтроллеров с поддержкой подкачки кода из расширенной памяти (xdata) до 16 Мб содержат:

- **C51 Compiler (with OMF2 Output)** - компилятор С (с новым выходным файловым форматом);
- **AX51 Macro Assembler** – макроассемблер с поддержкой Extended 8051;
- **LX51 Lincer/Locater** – динамический загрузчик/компоновщик с поддержкой Extended 8051.

Также в Keil Software входят дополнительные средства разработки:

- Конвертер объектных файлов объектных файлов OC51;
- Конвертер объектных и HEX-файлов OH51;
- Менеджер библиотек LIB51;
- Интегрированная среда разработки (IDE) Keil uVision;
- Операционная система реального времени (Real-Time Operating System - RTX).

3.2 Интегрированная среда разработки Keil uVision (uVision IDE)

Простая в использовании интегрированная среда разработки (Integrated Development Environment) uVision IDE фирмы Keil позволяет непосредственно вызывать симулятор или внутрисхемный эмулятор и содержит богатый набор инструментальных опций:

- Device Database - интеллектуальная база данных;
- Project Management - управление проектами;
- Source Code Editor - интегрированный редактор;
- Building Projects - автоматическая генерация проекта;
- Integrated Utilities - средства, облегчающие создание проекта.

База данных устройств (Device Database).

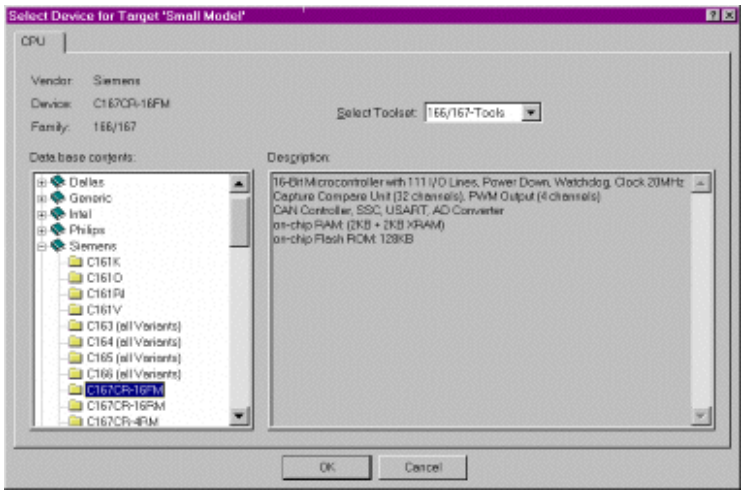

Рис. 1. База данных устройств

База данных содержит детальную информацию о всех устройствах, поддерживаемых инструментальными средствами Keil. База данных поддерживает параметрический поиск МК, удовлетворяющего специфическим требованиям (см. рис. 1). При выборе устройства из базы данных все требуемые опции в проекте под управлением uVision устанавливаются автоматически. Device Database содержит подробное описание конфигурирования и ссылки на другие источники информации (data sheets, оценочные платы, эмуляторы).

Управление проектом (Project Management)

Программный проект состоит из большого числа связанных друг с другом исходных файлов, которые часто обрабатываются индивидуально. Например, часть файлов подлежит С-компиляции, другие следует ассемблировать, а третьи требуют некоторой специальной обработки пользователем. Здесь на помощь приходит Менеджер проекта, который дает разработчику методику создания проекта из исходных файлов, различных опций раз-

работки и директорий. Проект в *uVision* может сгенерировать одну или несколько *target*-программ, каждая из которых компилируется по индивидуальным правилам. На основе исходных файлов создаются *target*-программы, объединенные в группы *Groups*. При этом достигается простая интеграция различных исходных файлов в проект.

Редактор исходного кода (Source Code Editor)

Интегрированный в uVision редактор значительно облегчает подготовку исходного текста за счет многооконности, выделения синтаксиса цветом и исправления ошибок в режиме диалога. Редактор настраивается на конкретный проект и в соответствии со вкусами пользователя. Редактирование остается доступным и во время отладки программы. Это создает все условия для быстрого тестирования и корректировки Вашего приложения.

Сборка проекта (Building Projects)

uVision содержит встроенную утилиту *make*, которая используется для компиляции, ассемблирования и линкования программ. При нажатии на кнопку *Build Target* осуществляется компиляция исходного файла. Ассемблер и компилятор автоматически генерируют зависимости между файлами и добавляют их в проект. Благодаря этой информации вновь обрабатываются только те файлы, которые претерпели изменения или файлы, включающие измененные файлы. Во время компиляции и ассемблирования исходного файла, в окне *Output Window* появляется статусная информация, сообщения об ошибках и предупреждения (см. рис. 2).

Рис. 2. Окно Output Window

При двойном щелчке на сообщение об ошибке или на предупреждение происходит переход к редактированию файла (при этом uVision продолжает обработку исходного файла в фоновом режиме). Номера строк ошибок и предупреждений синхронизированы и соответствуют факту после внесения исправлений в исходный файл. Для получения справки о сообщении об ошибке следует выбрать сообщение и нажать клавишу F1. При глобальной оптимизации uVision неоднократно компилирует исходный файл для достижения оптимального использования ресурсов микроконтроллера.

Все параметры проекта сохраняются в специальном файле, который содержит список исходных файлов, командные строки компилятора, ассемблера, редактора, отладчика, симулятора и утилиты make. При использовании этого файла компиляция и линковка проекта происходят по нажатию одной клавиши.

Встроенные утилиты (Integrated Utilities)

uVision содержит мощные средства, облегчающие создание проекта:

- Source Browser - база данных программных символов для быстрой навигации по исходному файлу;
- Find in Files - полный поиск во всех выделенных файлах;
- Tools Menu - утилиты других фирм из uVision IDE;
- SVCS - контроль версии программного обеспечения;
- PC-Lint - анализ синтаксиса исходного кода;
- Flash tool - утилиты загрузки flash-памяти;
- Easy CASE - генерация кода при задании программы на уровне блок-схемы;

- DAvE - автоматическая генерация программ и драйверов для периферии МК Infineon.

3.3 Компилятор Cи Cx51 (C Compiler Keil Cx51)

Новые возможности компилятора Cx51 (Cx51 Highlights):

- Cx51 поддерживает все разновидности 8051 и обеспечивает доступ ко всем программным компонентам;
- Быстрая 32-разрядная IEEE арифметика с плавающей точкой;
- Cx51 поддерживает множественные указатели данных DPTR и дополнительные арифметические устройства;
- Детальные предупреждения и сообщения об ошибках, проверка синтаксиса;
- Cx51 осуществляет полную регистровую оптимизацию New Code Optimizations;
- Доступ на С ко всем регистрам SFR, побитно адресуемым регистрам и отдельным битам Memory and SFRs;
- Очень быстрые прерывания за счет reentrant функций Interrupt Functions;
- Гибкие указатели областей памяти Flexible Pointers;
- Модели и селекторы памяти Memory Models and Memory Selectors;
- Эффективные механизмы memory banking и отладки - расширение адресного пространства за пределы 64 KB;
- Оптимизация при линковании - инструкции AJMP и ACALL;
- Поддержка отладочной информации для всех эмуляторов.

Оптимизация кода (Code Optimizations)

Cx51 поддерживает эффективные механизмы оптимизации, которые генерируют программы минимального размера:

- Регистровая оптимизация *Dynamic Register Allocation* позволяет разместить в регистрах больше переменных, уменьшить размер кода (за счет уменьшения числа команд MOV) и сократить объем оверлейных данных;
- Общая оптимизация кода *Common Tail Optimization* комбинирует идентичные фрагменты кода в специальных блоках и сокращает размер кода.

Память и регистры специальных функций (Memory and SFRs)

Компилятор Cx51 осуществляет прямое управление банками регистров и полное их использование, побитовую адресацию данных:

- Для доступа к регистрам специального назначения и их отдельным битам используются ключевые слова *sfr* и *sbit*;
- В соответствие переменной может быть назначен любой сегмент адресного пространства. С помощью ключевого слова _at_ переменные могут быть размещены по фиксированному адресу памяти.

Функции обработки прерываний (Interrupt Functions)

Cx51 осуществляет эффективное управление прерываниями при написании функций прерывания на С за счет малого времени вызова/возврата в/из прерывания и переключения регистровых банков. Cx51 поддерживает *reentrant* функции и код, не привязанный жестко к регистровым банкам, для генерации процедур прерывания и использования в многозадачных приложениях. Рекурсивные или повторно используемые функции определяются с помощью ключевого слова *reentrant*. Функции, вызываемые многими задачами должны быть определены как *reentrant*.

Гибкие указатели (Flexible Pointers)

Линковщик поддерживает *code banking*, а *uVision Debugger* поддерживает тестирование программ размером до 16MB *code* и *xdata*. Cx51 имеет два типа указателей для различных областей памяти:

- Основные указатели *Generic pointers* позволяют получить доступ ко всем областям памяти 8051, сохраняя информацию о типе памяти и адресе объекта в 3-х байтах;

- Специальные указатели *Memory-specific pointers* объявляются через тип памяти и указывают на определенную область памяти 8051. Поскольку для сохранения информации об объекте требуется всего 2 байта, такие указатели позволяют сгенерировать более компактный код

Модели и селекторы памяти (Memory Models and Memory Selectors)

Модель памяти определяется с помощью *default memory selector* используемого для переменных. Однако всегда есть возможность вполне определенно специфицировать *memory selector* для любой переменной. Область размещения переменных и функций и время доступа к ним определяется моделью памяти. Выбор модели памяти зависит от требуемого размера и физического размещения: *Small* - 128 байт, *Compact* - 256 байт, *Large* - 64 Кбайт. Несколько типов селекторов позволяют осуществить эффективный доступ к различным областям памяти и сгенерировать компактный код (см. табл. 1).

Табл. 1. Модели и селекторы памяти

Селектор	Область памяти
data	128 байт во встроенной RAM – непосредственная адресация
bdata	16 байт во встроенном RAM - непосредственная битовая/байтовая адресация
idata	256 байт во встроенном RAM - косвенная адресация
pdata	256 байт в страничной внешней RAM
xdata	64 Кбайт расширенной RAM
code	64 Кбайт памяти программ
far	16 Мбайт памяти data/const, размер объекта 64 Кбайт
near	64 Кбайт непосредственно адресуемой памяти для 251
huge	16 Мбайт косвенно адресуемой памяти, объект произвольного размера
edata	96 байт расширенной побитно адресуемой памяти для 251

В состав Cx51 входят два компилятора: C51.exe и CX51.exe. Более подробную информацию о компиляторах можно получить в руководстве пользователя [2]. Кроме этого, в техническом описании учебно-лабораторного стенда SDK-1-1 приведён перевод с английского языка для компилятора C51.exe [3].

При работе в Keil uVision выбор компилятора (C51.exe или CX51.exe) происходит автоматически в зависимости от типа микропроцессора.

3.4 Макроассемблер Ax51

Ассемблер фирмы Keil Ax51 специально разработан для семейства микроконтроллеров 8051. Ассемблер в основном применяется при написании фрагментов программ, наиболее критичных к скорости, размеру кода и возможностям аппаратного управления. В ассемблер включен макроязык, использование которого ускоряет разработку и экономит общее время проектирования. Макроязык позволяет также осуществлять доступ ко всем ресурсам микроконтроллеров с использованием символьных обозначений регистров.

В состав Ax51 (аналогичным образом как и в Cx51) входят компиляторы: A51.exe и AX51.exe. При этом компилятор AX51.exe поддерживает расширенные типы микропроцессоров 8051. Более подробную информацию о компиляторе Ax51 можно получить в руководстве пользователя [4].

3.5 Отладчик (uVision Debugger)

uVision Debugger фирмы Keil позволяет вести отладку исходных текстов программ, написанных на C и ассемблере или в смешанном формате, сохраняет историю трассировки и позволяет выбирать между симулятором, монитором и внутрисхемным эмулятором. В состав uVision Debugger входят:

- *CPU & Peripheral Simulator* - симулятор CPU и периферии;
- Performance Analyzer & Code Coverage - анализаторы производительности и эффективности кода;
- Target Monitor - отладочный монитор;
- uVision Cx51 Target Debugger - интерфейс к отлаживаемому устройству через драйверы AGDI;
- uVision C166/ST10 Target Debugger - поддержка начальной загрузки *Bootstrap* и интерфейса *OCDS/JTAG*;
- Breakpoints - точки останова;
- Debug Function Scripts - С-подобный язык для записи функций;
- Variables and Memory - просмотр областей памяти и регистров.

Симулятор центрального процессора и интегрированной периферии (CPU & Peripheral Simulator)

uVision Simulator - чисто программный продукт, который осуществляет отладку в исходных кодах, симуляцию на уровне символов и отладку непосредственно на рабочей плате - *target debugging*. Моделируется вся система команд и все периферийные устройства.

Для просмотра и изменений установок периферии служат специальные диалоговые окна. Симулятор полностью поддерживает периферийные устройства микроконтроллера посредством специальных драйверов *xxx.DLL*.

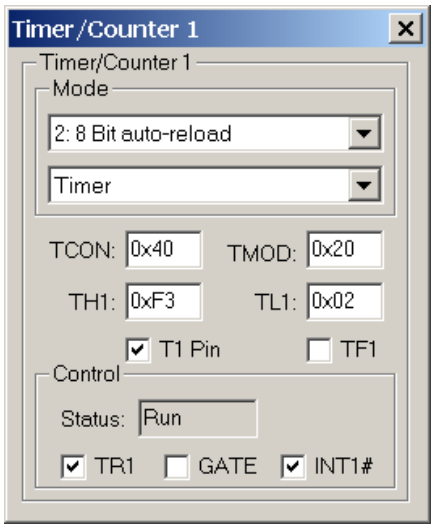

Рис. 3.

Для симуляции аппаратной части МК 8051 Keil предлагает *Advanced Generic Simulation Interface (AGSI)*. *AGSI* является спецификацией *API*, расширяющей возможности симуляции с помощью диалоговых настроек. Кроме *AGSI DLL* от третьих фирм поддерживаются распространенные МК: Philips 51MX, Dallas 390, Analog Devices, Atmel, Mentor M8051EW.

В распоряжение пользователя предоставляется ряд окон, отображающих состояния таймеров (см. рис. 3), портов, прерываний, сторожевого таймера, последовательного порта, аналогово-цифрового преобразователя и т.д. Параметры этих устройств могут быть установлены и изменены в соответствии с контекстом приложения. *uVision Simulator* позволяет проводить пошаговую отладку программы, просматривая ее в окне *Debug*. Трассировщик запоминает команды и позволяет их просматривать в окне *Trace*. Изменение заранее заданных переменных отслеживает окно *Watch*. Последовательность вызова процедур отображается в окне *Call-Stack*.

Анализатор производительности и эффективности кода (Performance Analyzer & Code Coverage)

В *uVision Debugger* встроен анализатор производительности *Performance Analyzer*, который фиксирует время исполнения программных модулей. Задавая список модулей для анализа, пользователь получает диаграмму затрат времени на каждую часть программы (см. рис. 4).

uVision Debugger позволяет также провести анализ эффективности кода *Code Coverage*, локализуя части программы, к которым редко происходит обращение, что позволяет удалить ненужный код (см. рис. 5).

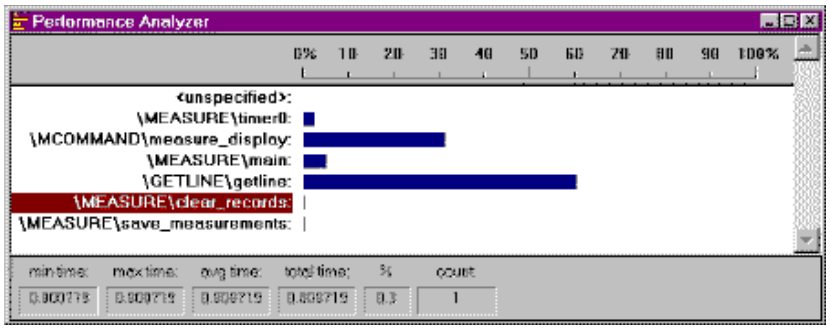

Рис. 4.

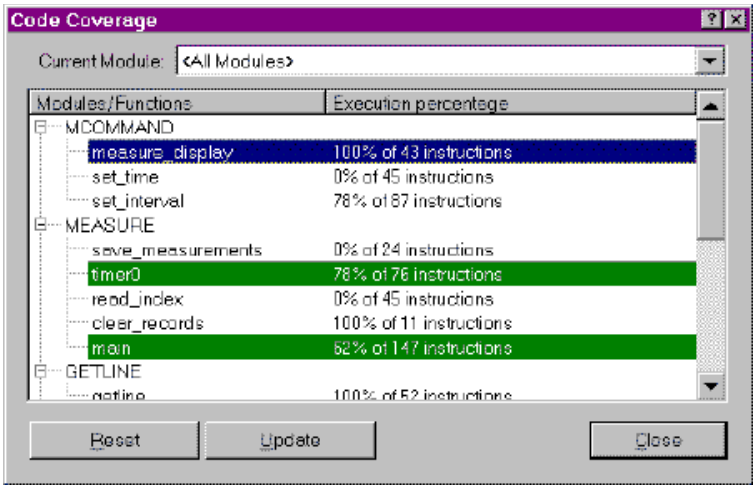

Рис. 5.

Отладочный монитор (Target Monitor)

При отладке программ на плате в качестве интерфейса используется специально сконфигурированный отладочный монитор, загружаемый в ОЗУ с помощью встроенного начального загрузчика или прошиваемый в EPROM. Программа монитор обеспечивает прямой интерфейс для отладчика/симулятора и легко настраивается на любой микроконтроллер. При помощи монитора производится комплексная отладка приложения на плате. В остальном же отладка ничем не отличается от режима симуляции. Требования к ресурсам микроконтроллера со стороны монитора минимальны.

Отладочный монитор Cx51 (uVision Cx51 Target Debugger)

В *uVision Debugger* для Cx51 интерфейс к отлаживаемому устройству осуществляется через драйверы Advanced Generic Debugger Interface (AGDI). На сегодняшний день существуют следующие драйверы:

- *Monitor-51* - конфигурируемый монитор для отлаживаемого устройства, который прошивается в ROM устройства (поставляется со многими оценочными платами);
- *Monitor-390* - конфигурируемый монитор для *Dallas contiguous mode*;
- *ISD51* - внутрисхемный отладчик для стандартных МК 8051;
- *EPM900* - эмулятор/программатор для *Philips LPC900*;
- *SmartMX DBox* - эмулятор для *Philips SmartCards*;
- Некоторые из МК подключаются к uVision Debugger также с помощью драйверов *AGDI: ChipCon CC1010, Cygnal 51Fxxx, Cypress USB, Infineon SLE66, SST Flash-Flex51, Triscend E5*;

- Драйверы *uVision AGDI* доступны также для многих эмуляторов.

Точки останова при отладке (Breakpoints)

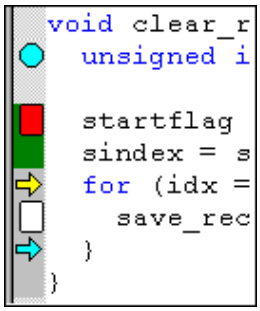

Рис. 6.

uVision предлагает широкие возможности по заданию простых и условных точек останова (см. рис. 6). Условием останова может быть или результат выражения или операция обращения к ячейке памяти/ переменной (чтение, запись, доступ). Для редактирования и просмотра параметров контрольных точек служит окно *Breakpoint*. Точки останова могут остановить исполнение программы или запустить команду или сценарий отладчика.

Язык функций отладки (Debug Function Scripts)

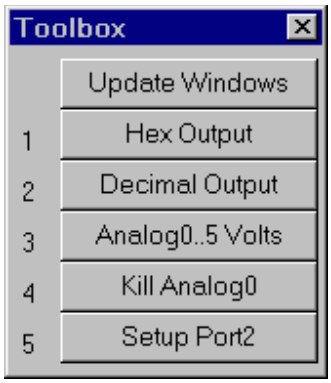

Рис. 7.

Для автоматизации типовых операций, выполняемых при отладке, могут быть созданы специальные командные файлы. Для ввода этих команд служит окно *Command* или окно *Toolbox* (см. рис. 7). Кроме того uVision поддерживает С-подобный функциональный язык, позволяющий генерировать:

- Встроенные функции типа *printf, memset, rand* и другие;
- Сигнальные функции для моделирования аналоговых и цифровых входных/выходных сигналов CPU;
- Функции пользователя для расширения возможностей команд и повторяющихся операций.

Переменные и память (Variables and Memory)

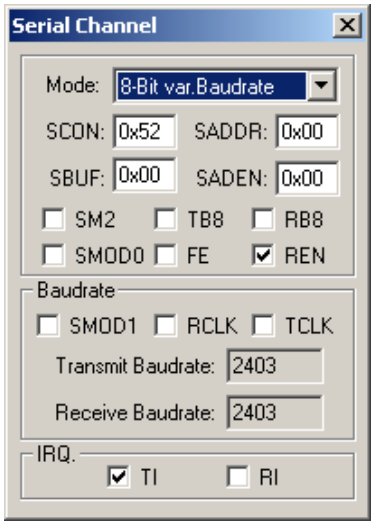

Рис. 8.

В распоряжении пользователя находятся окна для просмотра областей памяти *Memory* и состояний регистров *Register*. Например, с помощью окна *Serial I/O* (см. рис. 8) становиться возможной наглядная симуляция последовательного ввода/вывода. *uVision* предлагает несколько путей для просмотра и изменения переменных и памяти:

- Поместить указатель мышки над переменной, чтобы посмотреть ее значение;
- Использовать окно *Watch* для просмотра и изменения локальных и определенных пользователем переменных;
- Использовать окно Memory для просмотра и редактирования до 4-х массивов памяти.

Более подробную информацию о uVision можно узнать в руководстве пользователя [1].

3.6 Многозадачное ядро реального времени (RtxTiny Real Time Kernel)

Многозадачное ядро реального времени *RtxTiny* предназначено для разработки однопроцессорных многозадачных систем и интегрирована в программный пакет *PK51*. *RtxTiny* - облегченная версия популярной операционной системы *RTX51*. *RtxTiny* имеет следующие свойства:

- Поддержка множественных указателей *DPTR* и арифметических устройств;
- Поддержка режимов одиночного кристалла и *code banking*;
- *Round robin* (циклическое) и совместное переключение задач;
- Управление задачами с функциями инициации и удаления;
- События *Timeout, Signal и Ready*;
- Поддержка прерываний от посылаемых сигналов.

Более подробно о RTX51 можно узнать из руководства пользователя данной операционной системы [5].

3.7 Структура каталогов

Структура каталогов, и в целом методика разработки приложений, в Keil аналогична как в программных продуктах Borland C++, Microsoft Visual C++ и т.д.

На сервере Class FTF (диск L) средства разработки Keil Software установлены следующим образом:

L:\Keil\C51\ASM	Файлы ассемблерных определений SFR-регистров для разных процессоров и файлы исходных шаблонов;
L:\Keil\C51\BIN	Исполнимые файлы средств разработки для 8051;
L:\Keil\C51\EXAMPLE	Примеры приложений для 8051;
L:\Keil\C51\RTX_TINY	Файлы облегчённой версии операционной системы реального времени RTX51;
L:\Keil\C51\INC	Заголовочные файлы компилятора Си;
L:\Keil\C51\LIB	Файлы библиотек компилятора Си;
L:\Keil\C51\UV2	Файлы интегрированной среды разработки uVision.

В каталоге L:\Keil находится файл «Запуск Uv2.lnk» - ярлык для запуска интегрированной среды разработки uVision. Для удобства работы скопируйте этот ярлык на свой рабочий стол.

3.8 Цикл разработки приложения в Keil uVision (Software Development Cycle)

Цикл разработки приложения, при использовании средств разработки Keil Software, приблизительно состоит из следующих этапов:

1. Создание проекта, выбор target-микроконтроллера из базы данных и настройка средств разработки.
2. Создание исходных файлов на языке Си и (или) ассемблере.
3. Сборка приложения с помощью менеджера проектов.
4. Исправление ошибок в исходных файлах.
5. Проверка приложения.

Цикл разработки приложения для 8051 проиллюстрирован на блок-схеме (рис. 9).

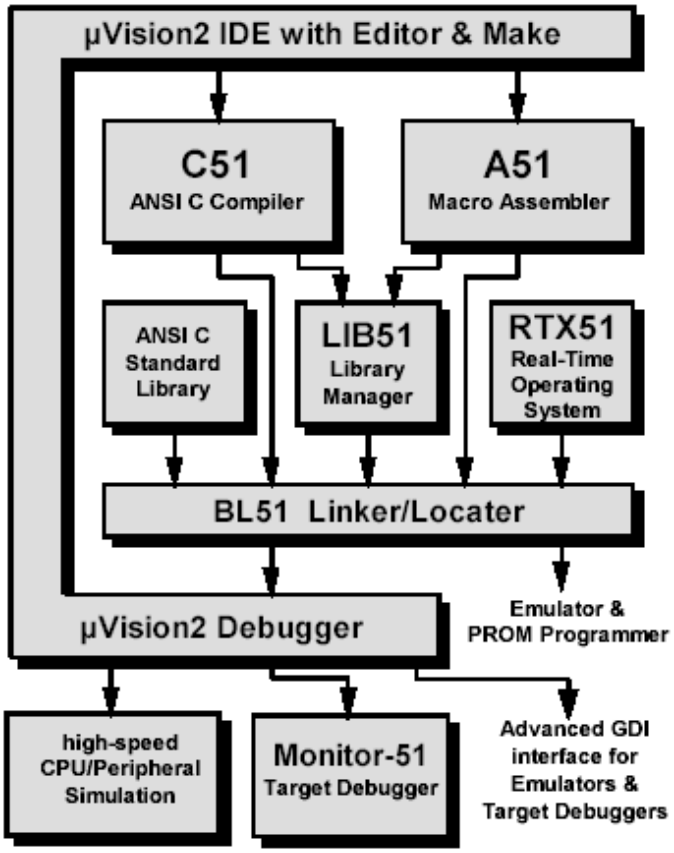

Рис. 9. Цикл разработки приложения для 8051 в Keil Software

Элементы блок-схемы (см. рис. 9) уже рассмотрены в данной главе. Более подробную информацию можно получить из руководства пользователя [1].

4 Быстрый старт

“Быстрый старт” – это обычный приём разработчиков современных программных средств. Цель состоит в том, чтобы, не углубляясь пока в подробности, дать новичку или достаточно опытному пользователю первое представление о программном средстве, дать возможность быстро получить конкретный результат. Полное представление, знания и умения появятся позже в процессе работы и изучения справочных материалов.

В качестве примера возьмём простейшую программу, с которой начинают изучение языков программирования многие поколения студентов. “Hello World” - программа аналогичная примеру из папки \C51\Examples\Hello\, которая выдаёт в последовательный порт (UART) микроконтроллера строку символов “Hello World” (“Привет Мир”). Весь исходный текст программы содержится в файле hello.c:

Пример 1. Исходный текст приложения “hello”

```
/* ---------------------------------------
Ваша первая программа для SDK-1-1 на основе MCS51
--------------------------------------- */
#include <ADuC812.h>
#include <stdio.h>

// Подпрограмма работы с портами ПЛИС
void WriteMax(unsigned char xdata *regnum, unsigned char val)
{
#define MAXBASE 0x8;
unsigned char oldDPP=DPP;

DPP=MAXBASE;
*regnum=val;
DPP=oldDPP;
}

void main(void)
{
unsigned char svet = 1;
unsigned int pause;
// ------- Инициализация UART -----------
TH1 = 0xFD; // Скорость 9600 бит/с
TMOD = 0x20;// Таймер 1 в режиме autoreload
TCON = 0x40;// Запуск таймера 1
SCON = 0x50;// 8 bit UART, разрешение приема
PCON &= 0x7F;// Отключение удвоения скорости
TI = 1; // Требуется для работы с
RI = 1;// stdio.h
EA = 0;// Запрещение прерываний
do
    {
     printf ("Hello World\n");// вывод на терминал "Hello World"
     svet = svet<<1;// сдвиг влево
     if(svet==0) svet=1;
     WriteMax(0x7,svet);//вывод на светодиды
     for(pause=0;pause<=32000;pause++);// задержка
    } while(1);
}
```

4.1 Запуск uVision IDE и создание нового проекта

В каталоге L:\Keil находится файл «Запуск Uv2.lnk» - ярлык для запуска интегрированной среды разработки uVision. Для удобства работы скопируйте этот ярлык на свой рабочий стол, а затем запусти при помощи его среду разработки.

Любая новая работа в uVision IDE, как и во всех современных компиляторах, начинается с создания нового проекта. Файл проекта содержит имена всех исходных файлов, связанных с проектом, а также установки компиляции, трансляции и связывания файлов, чтобы генерировать выполняемую программу.

Для создания нового проекта необходимо выполнить следующие действия:

1. После запуска среды закрыть все существующие проекты (если таковые имеются), иначе могут возникнуть трудности на этапе отладки программы. Для этого в меню Project нажать Close project (см. рис. 10).

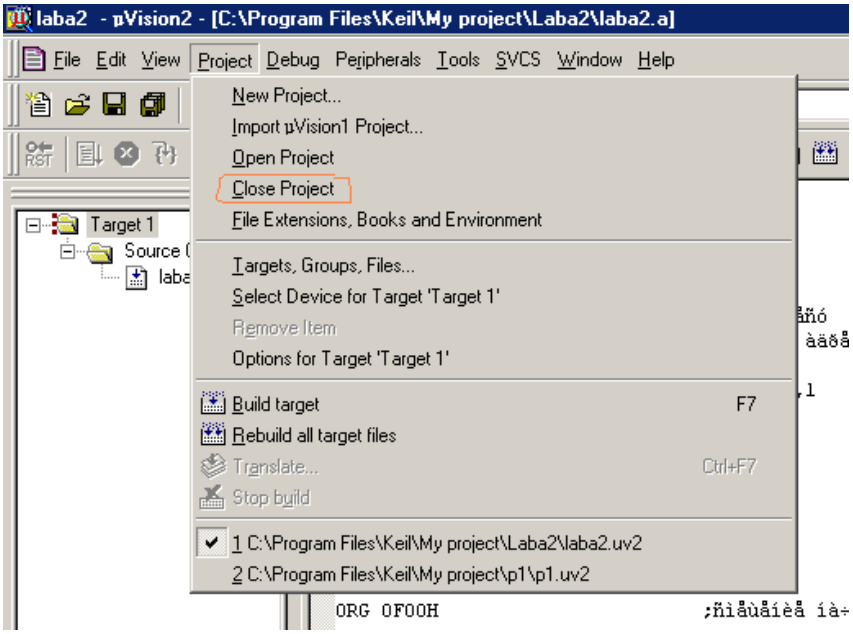

Рис. 10.

2. Создать на своём диске U следующие папки:

U:\MPT
U:\MPT\LAB1
U:\MPT\LAB1\Hello

например, при помощи Far – менеджера.

Желательно чтобы каждый проект находился в отдельной папке.

3. Создать новый проект, для чего в меню Project выбрать New Project (см. рис. 11). Высветиться окно с просьбой сохранить проект (см. рис. 12). Проект сохраняете в заранее подготовленную папку U:\MPT\LAB1\Hello.

Если проект уже существует, то в том же меню (рис. 11) нажать Open Project, после чего в появившемся диалоговом окне выбрать нужный проект.

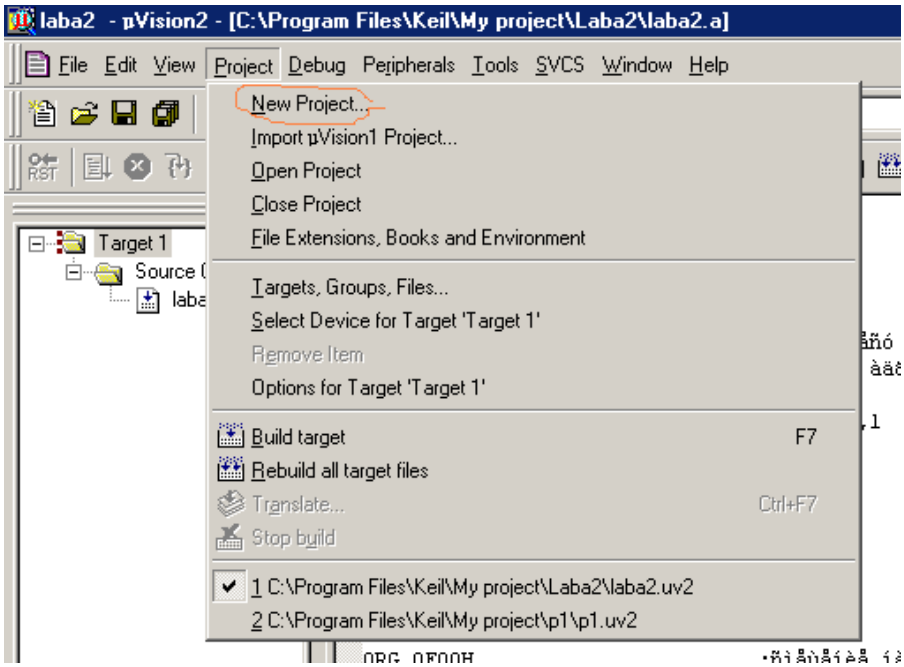
laba2 - µVision2 - [C:\Program Files\Keil\My project\Laba2\laba2.a]
File Edit View Project Debug Peripherals Tools SVCS Window Help
New Project...
Import µVision1 Project...
Open Project
Close Project
File Extensions, Books and Environment
Targets, Groups, Files...
Select Device for Target 'Target 1'
Remove Item
Options for Target 'Target 1'
Build target F7
Rebuild all target files
Translate... Ctrl+F7
Stop build
1 C:\Program Files\Keil\My project\Laba2\laba2.uv2
2 C:\Program Files\Keil\My project\p1\p1.uv2
Target 1

Рис. 11.

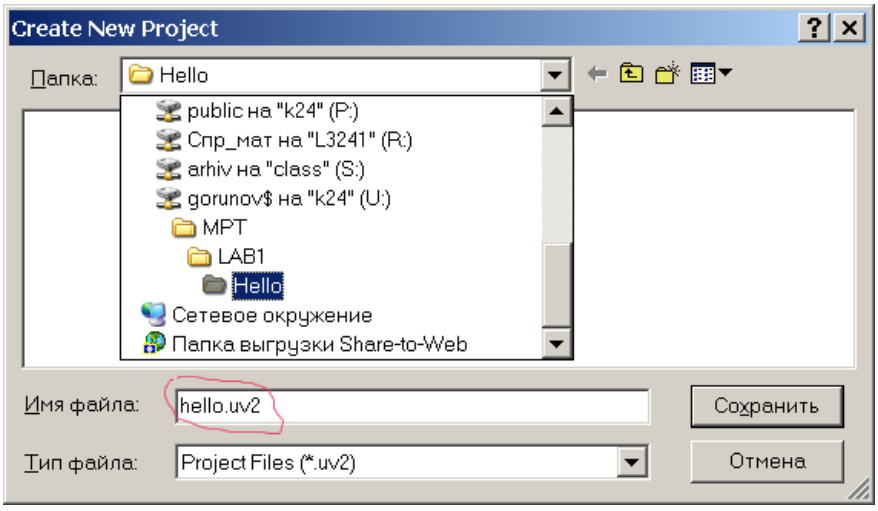
Create New Project
Папка: Hello
public на "k24" (P:)
Спр_мат на "L3241" (R:)
arhiv на "class" (S:)
gorunov$ на "k24" (U:)
MPT
LAB1
Hello
Сетевое окружение
Папка выгрузки Share-to-Web
Имя файла: hello.uv2
Сохранить
Тип файла: Project Files (*.uv2)
Отмена

Рис. 12.

После сохранения проекта высветиться диалоговое окно, в котором необходимо выбрать модификацию микроконтроллера. Необходимо выбрать Analog Device -> ADuC812 (см. рис. 13).

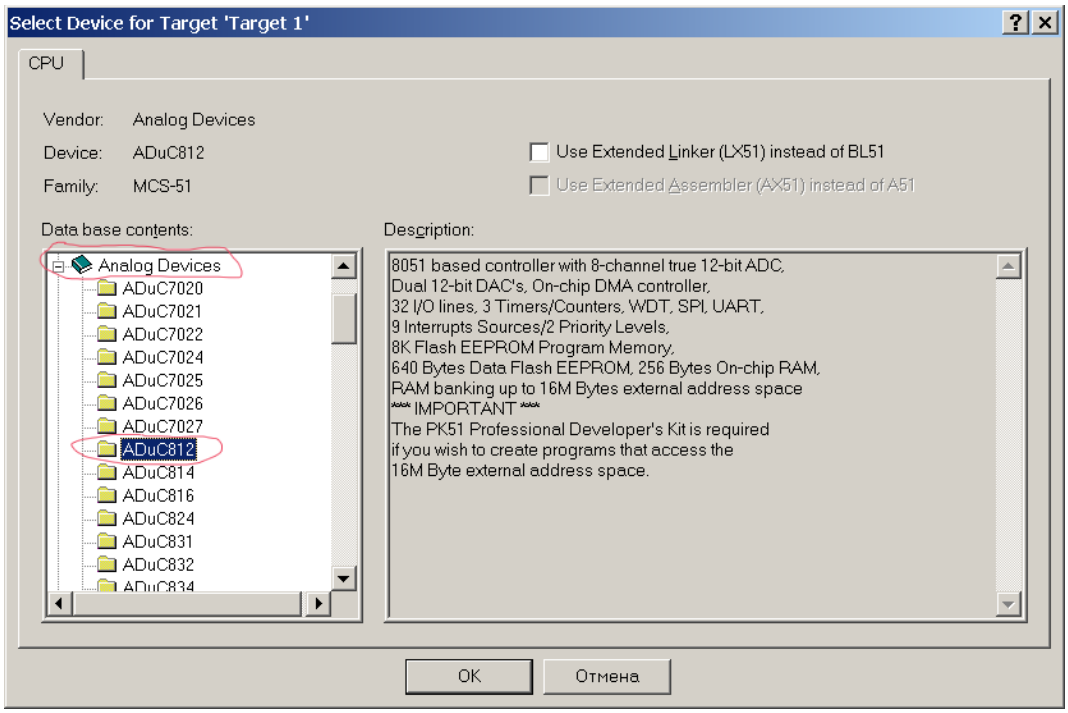

Рис. 13.

4. Настроить опции проекта, для чего нажать пункт меню Project -> Option for target и выставить параметры как показано на рис. 14.

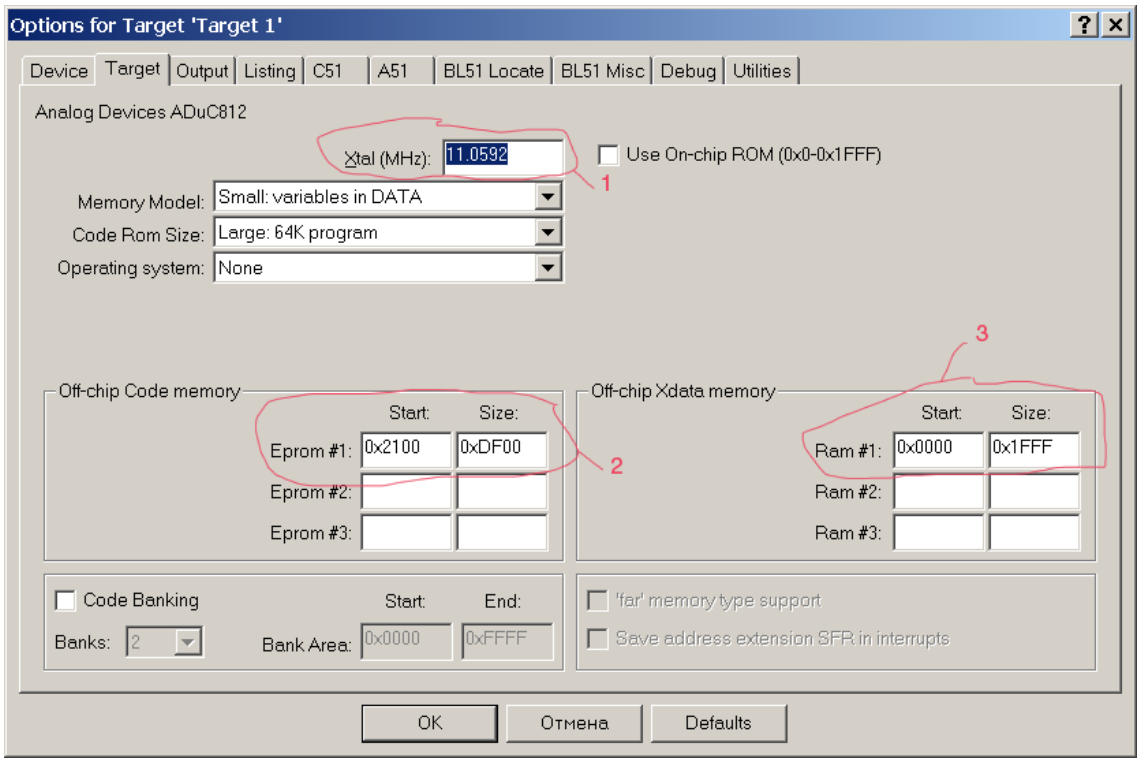

Рис. 14.

На рис. 14 выделено следующее:
1 – Частота резонатора. В SDK-1-1 установлен кварцевый резонатор на 11.0592 МГц.
2 – Область внешней памяти программ. В SDK-1-1 доступно 56 Кб памяти программ/данных, если не используется подкачка кода (Code Banking).
3 – Область внешней памяти данных. В SDK-1-1 младшие 8 Кб внешней ОЗУ доступны только для размещения данных. Будем в ней располагать переменные с типом xRAM.

Далее выбираем закладку Output (см. рис. 15), в которой устанавливаем флажок *Create HEX File*.

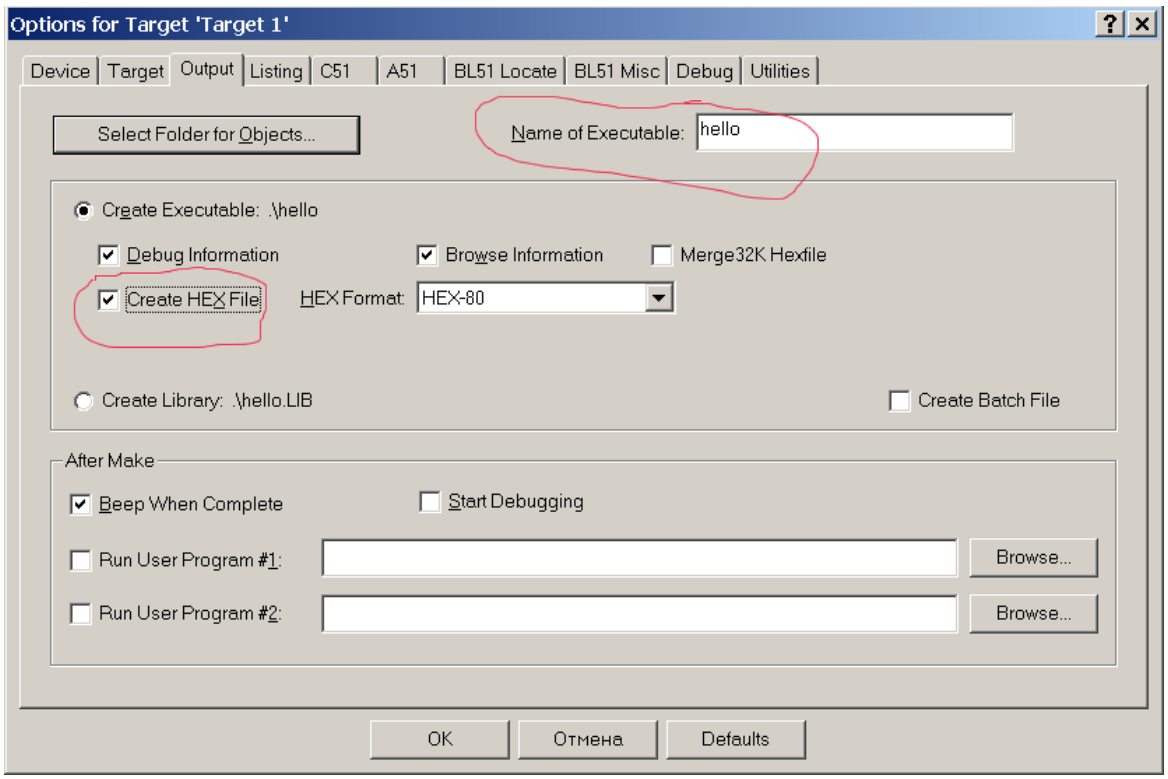

Рис. 15.

В графе *Name of Executable* указывается имя создаваемого hex-файла, как правило, оно совпадает с именем проекта.

Информацию об установки остальных параметров проекта можно получить из руководства пользователя [1]. В данном случае этих настроек достаточно.

4.2 Создание и добавление файла с исходным текстом, и его редактирование

Теперь необходимо создать исходный файл hello.c. Для этого в меню *File* необходимо нажать *New* (см. рис. 16), после чего сохранить его как hello.c в папку проекта используя меню *File -> Save AS*.

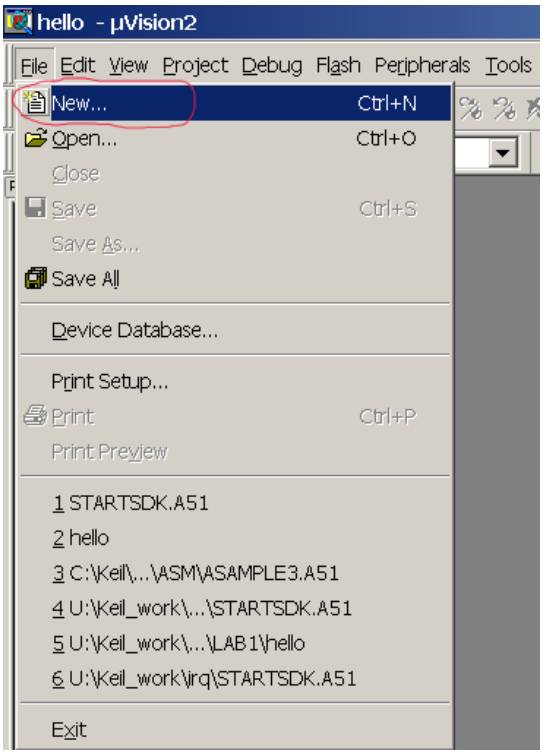

Рис. 16.

Далее необходимо файл hello.c добавить в проект следующим образом:

- Выделить курсором *Source Group 1* в *Project Window* (см. рис. 17).
- Правой кнопкой мыши вызвать меню и добавить файл в проект (см. рис. 18)

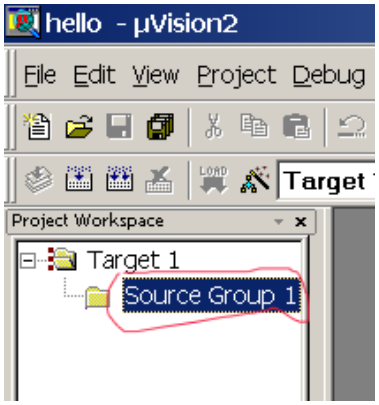

Рис. 17.

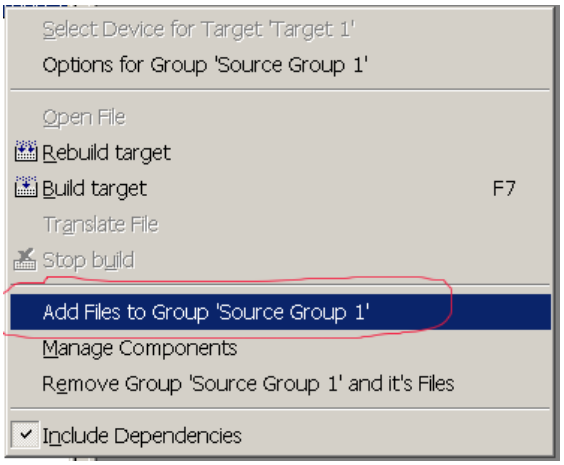

Рис. 18.

Для редактирования файла hello.c в окне *Project Window* левой кнопкой мыши щёлкните по соответственному файлу (см. рис. 19).

В файле hello.c необходимо набрать исходный текст примера 1.

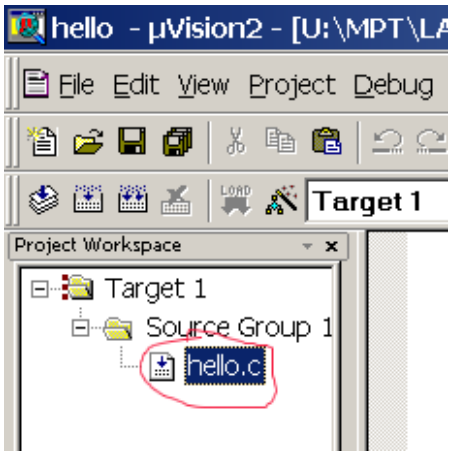

Рис. 19.

Кроме файла исходного текста в проект необходимо добавить ассемблерный файл *STARTSDK.A51*, который находится в папке L:\Study\МПС\SDK_1_1\EXAMPLE, это доработанный вариант стандартного *STARTUP.A51* файла инициализации Си приложений специально для учебно-лабораторного стенда SDK-1-1.

4.3 Сборка проекта и отладка

Откомпилировать проект, используя иконку *Build Target* или меню *Project -> Build Target* (см. рис. 20).

Для отладки программы использовать меню *Debug* (см. рис. 21).

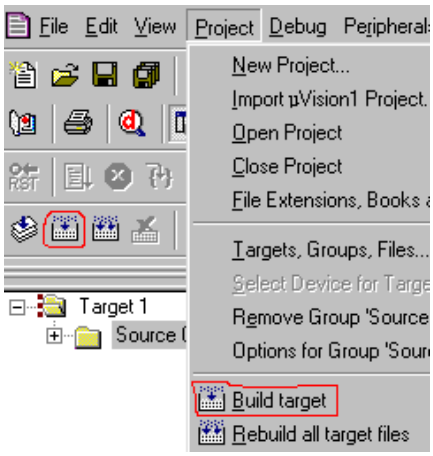

Рис. 20.

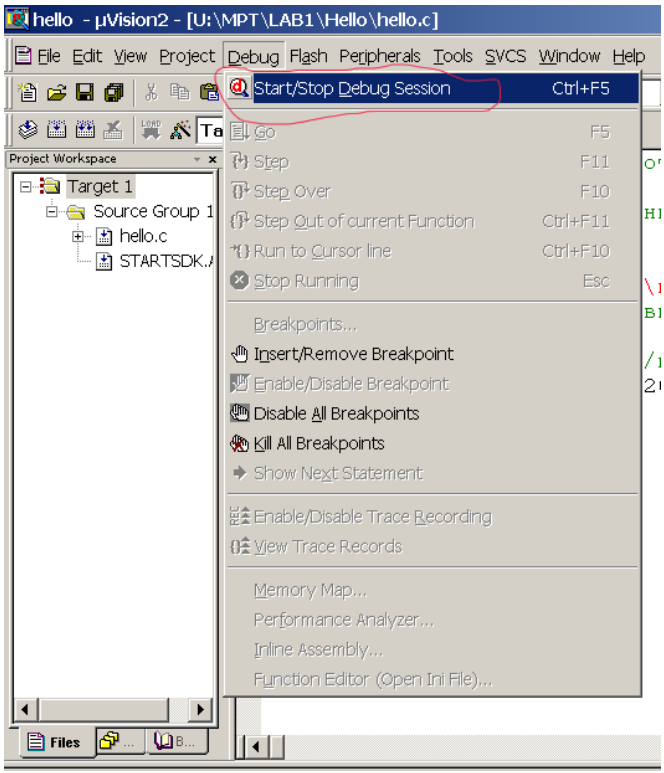

Рис. 21.

Отладчик uVision Debug позволяет выполнять следующие инструкции:
Step – выполнение текущей инструкции и переход на следующую;
Go – выполнение программы с текущей инструкции;
Break points – меню точек останова.

Команды *Step Over* позволяют “шагать” по каждой строке исходного текста. Текущая команда высвечивается на каждом шаге. *Step* позволяет войти в вызываемую функцию, *Step Over* – перешагнуть через неё, не входя во внутрь (см. рис. 22). Перечисленные команды находятся в меню 1 (выделено на рис. 22), а указатель 2 – показывает какую следующую инструкцию будет выполнять отладчик.

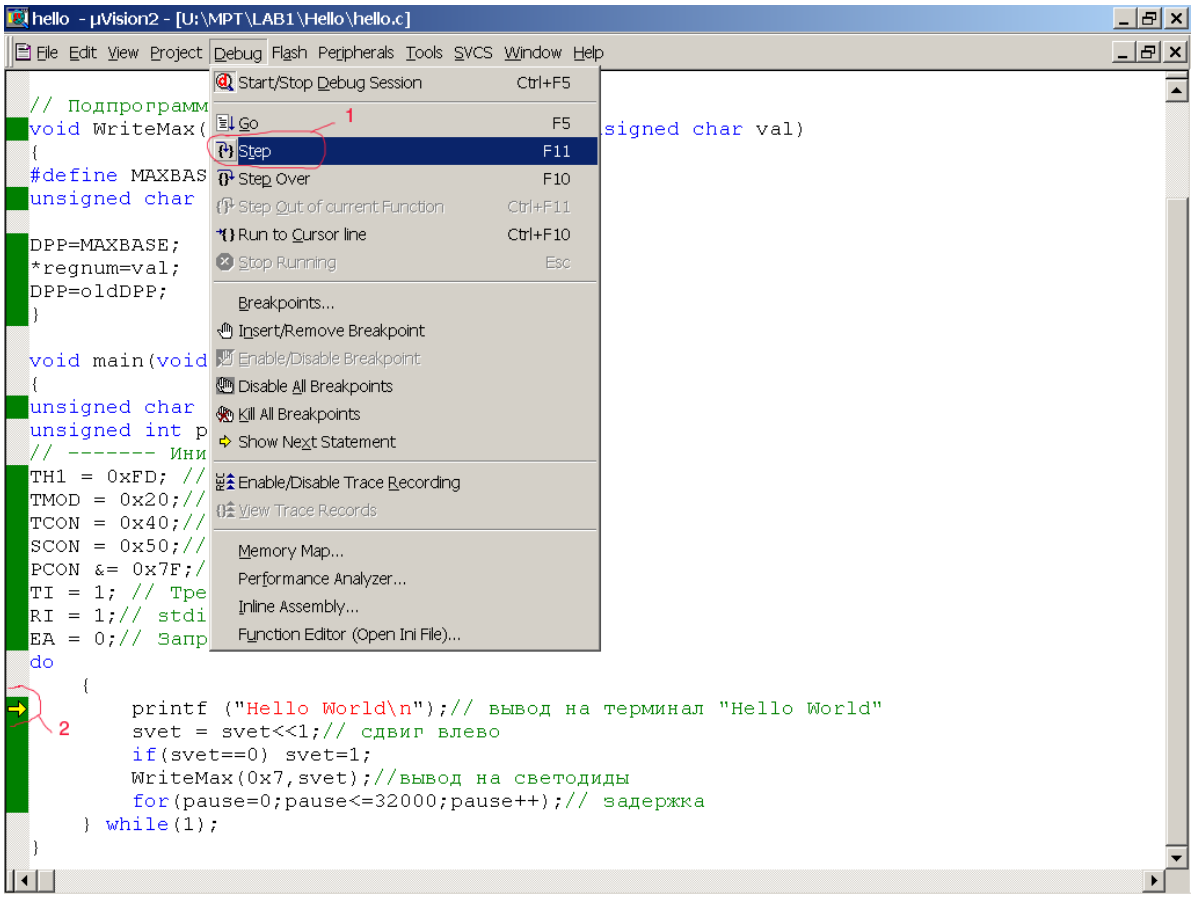

Рис. 22.

Отладчик uVision Debug позволят просматривать трассировку исходного текста на языке ассемблера. Для того чтобы включить данный режим необходимо использовать иконку *Disassembly Window* в поле *Debug* или меню *View -> Disassembly Window* (см. рис. 23). Кроме этого, при помощи отладчика можно просмотреть содержимое регистров, слово-состояние микропроцессора и т.д. (окно 2 – окно проекта), а также просматривать переменные (окно 3 – окно переменных).

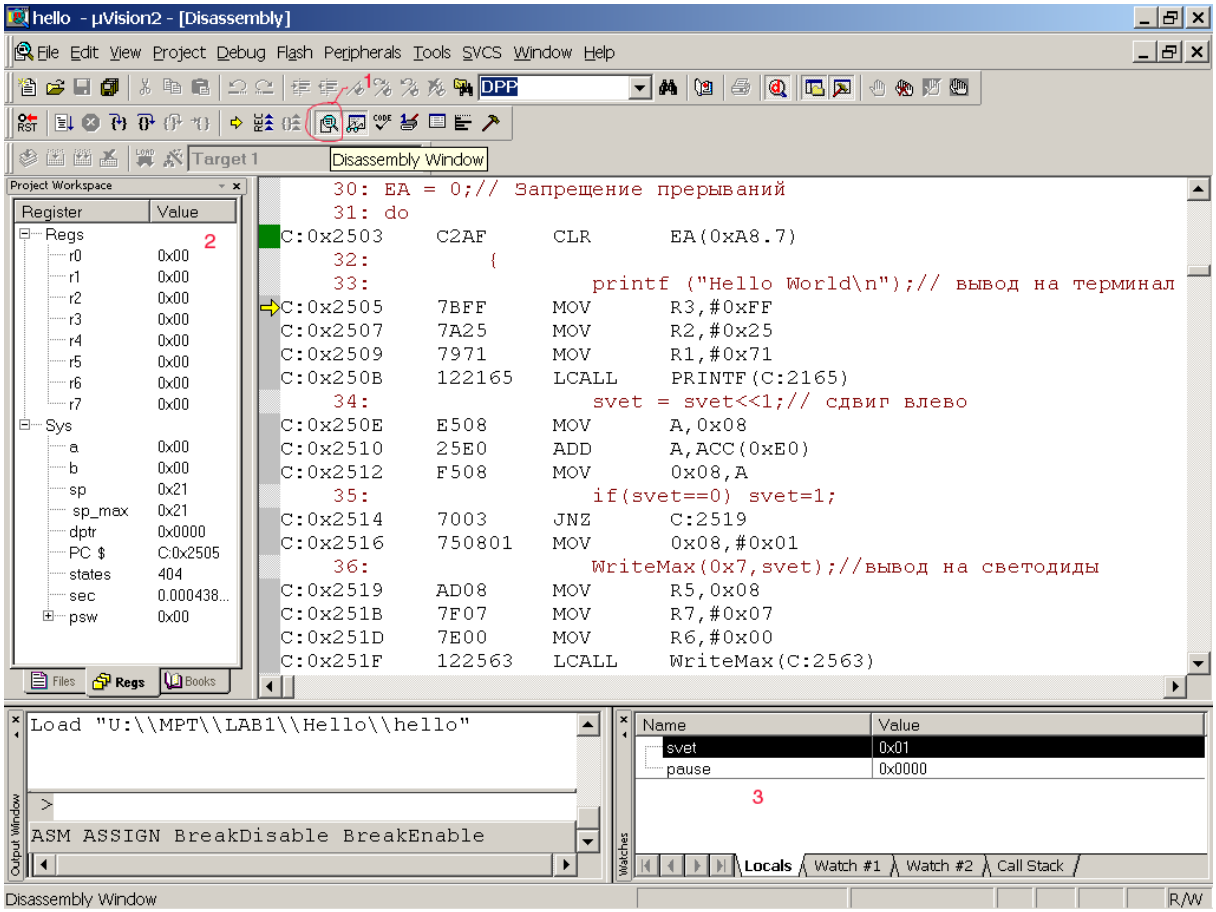

Рис. 23.

При помощи меню *Peripherals* можно просматривать и вносить изменение в состояние интегрированной периферии микроконтроллера (см. рис 24), а использование *Peripherals -> Reset CPU* позволяет в любой момент перезапустить программу.

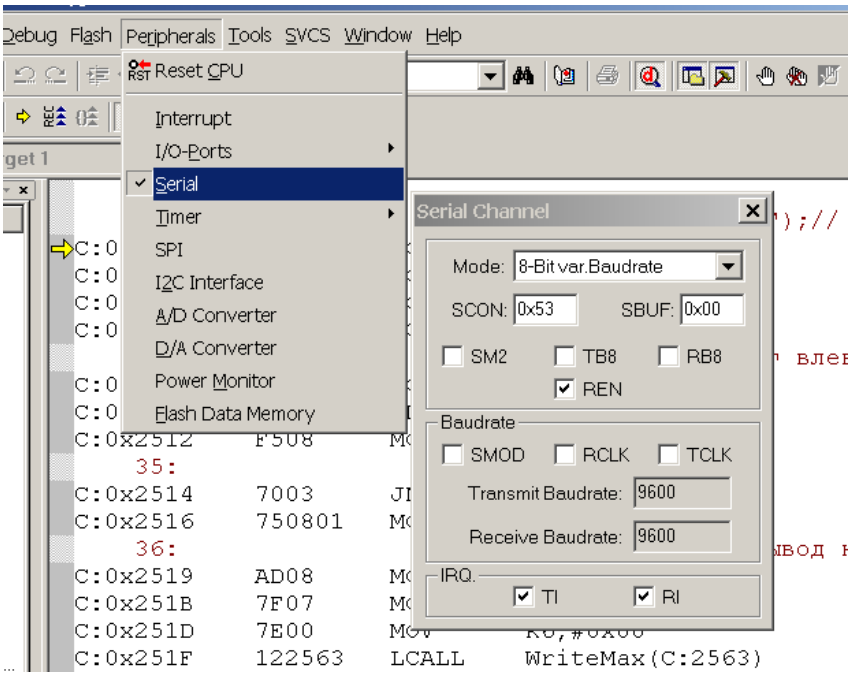

Рис. 24.

В ходе выполнения лабораторной работы необходимо проделать вышеописанные операции.

Более подробное описание возможностей отладчика можно получить из руководства пользователя [1].

4.4 Проверка работоспособности учебно-лабораторного стенда SDK-1-1

1. Изучить инструкцию по эксплуатации учебно-лабораторного стенда SDK-1-1 [3].
2. Ознакомиться с демонстрационной программой учебно-лабораторного стенда SDK-1-1 [6].
3. Запустить стенд в демонстрационном режиме.
4. По результатам тестирования определить работоспособность стенда.

4.5 Загрузка приложения в SDK-1-1 при помощи инструментальной системы T2

Предварительно перед выполнением загрузки приложения необходимо ознакомиться с руководством пользователя по программному обеспечению учебно-лабораторного стенда [3]. Далее необходимо выполнить следующие действия:

1. Скопировать инструментальную систему T2.exe с диска L:\Study\МПС\SDK_1_1\Utilities на свой диск U: в папку МРТ, где она будет располагаться в ходе всего курса лабораторных работ.
2. При помощи любого текстового редактора (например, «блокнот» Windows) создать интерпретационный командный файл load.txt и сохранить его в директории проекта U:\MPT\LAB1\Hello.

Данный файл должен содержать инструкции среды T2, например следующие:

Пример 2. Пример инструкций среды T2

```
0x2100 0x0 addhexstart hello.hex
9600 openchannel com3
0 term
loadhex hello.hex
0 term
bye
```

В соответствующих строках файла load.txt содержатся команды:

1. Добавления адреса запуска приложения в hex- образ приложения (код программы в 16-ном формате с абсолютными адресами).
2. Открытие последовательного интерфейса COM3 на скорости 9600 бит/с.
3. Запуск эмулятора терминала в бинарном формате.
4. Загрузка hex- образа в учебно-лабораторный стенд через открытый последовательный интерфейс. После загрузки приложение автоматически запустится по добавленному адресу.
5. Запуск эмулятора терминала в бинарном формате.
6. Закрытие всех открытых каналов и выход из среды T2.

Для загрузки приложения прямо из Keil uVision необходимо выполнить следующие действия:

1. При помощи меню *Project -> Option for Target* открыть окно параметров проекта и переключиться на закладку *Utilities*. В данном окне необходимо ввести настройки внешней утилиты программирования (в данном случае T2): путь к программе и командную строку (см. рис. 25). При этом параметр *Run independent* должен быть обязательно включен.

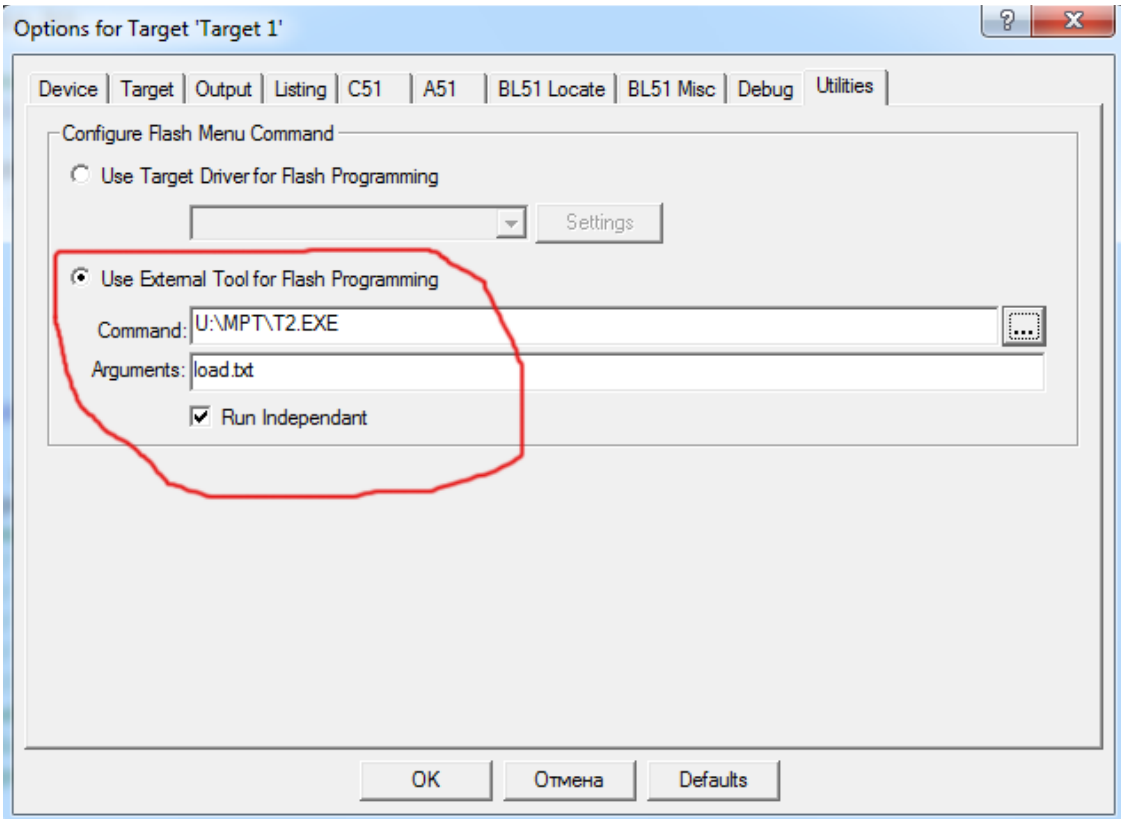

Рис. 25.

2. При помощи меню *Flash -> Download* запустить среду T2 на выполнение командного файла load.txt (см. рис. 26). После чего необходимо перезагрузить стенд (нажав на кнопку сброс).

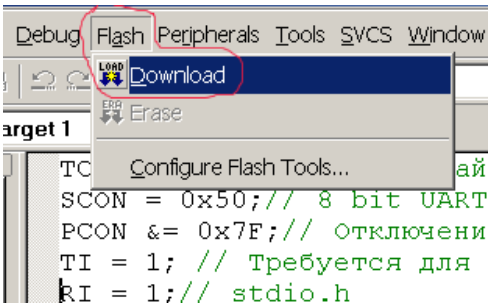

Рис. 26.

В результате чего должно появиться окно с приложением T2 (см. рис. 27).

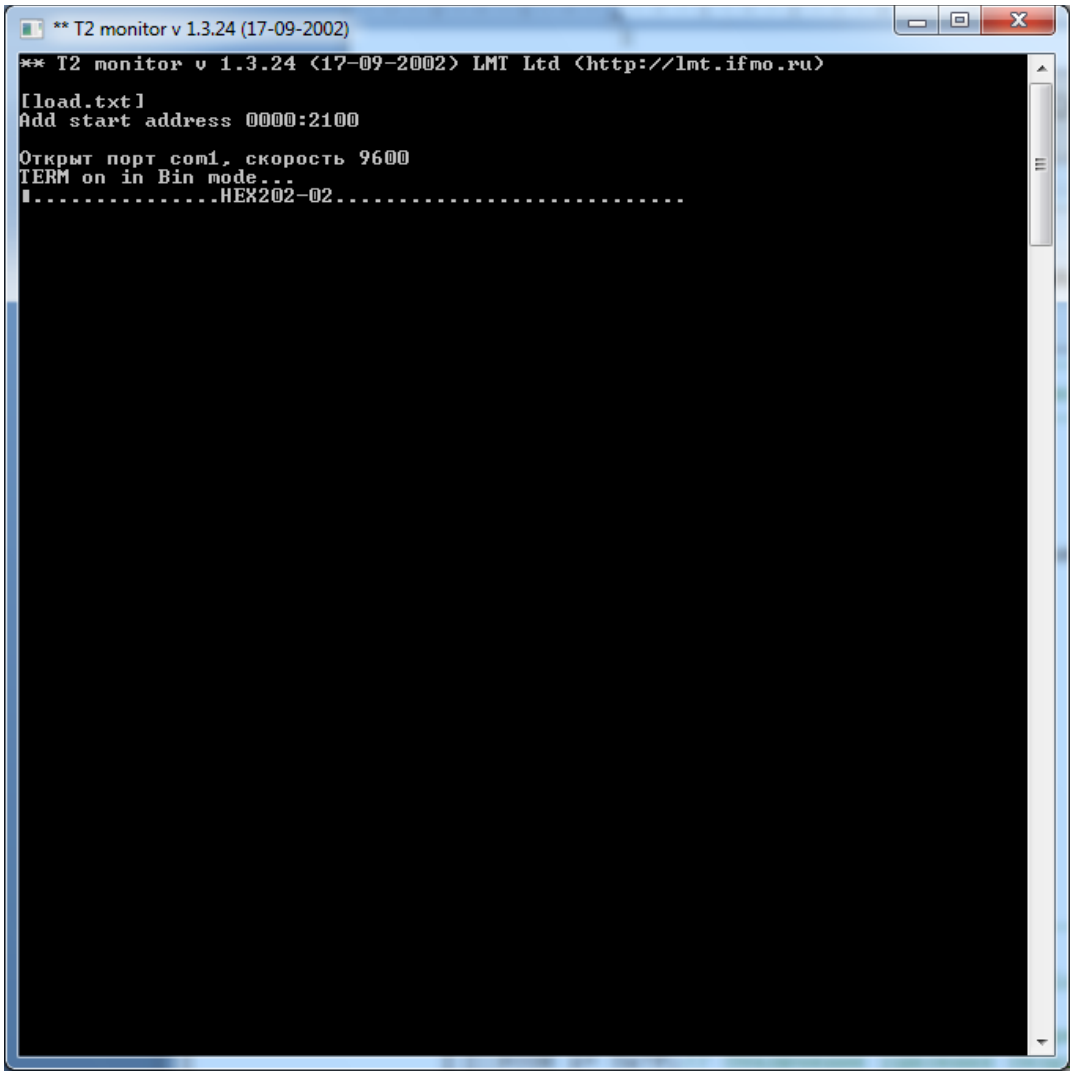

Рис. 27.

При правильной работе должна появиться строка терминала: HEX202……..

3. Далее необходимо нажать клавишу Esc, что должно привести к выполнению следующей инструкции командного файла load.txt, а именно loadhex (см. рис. 28).

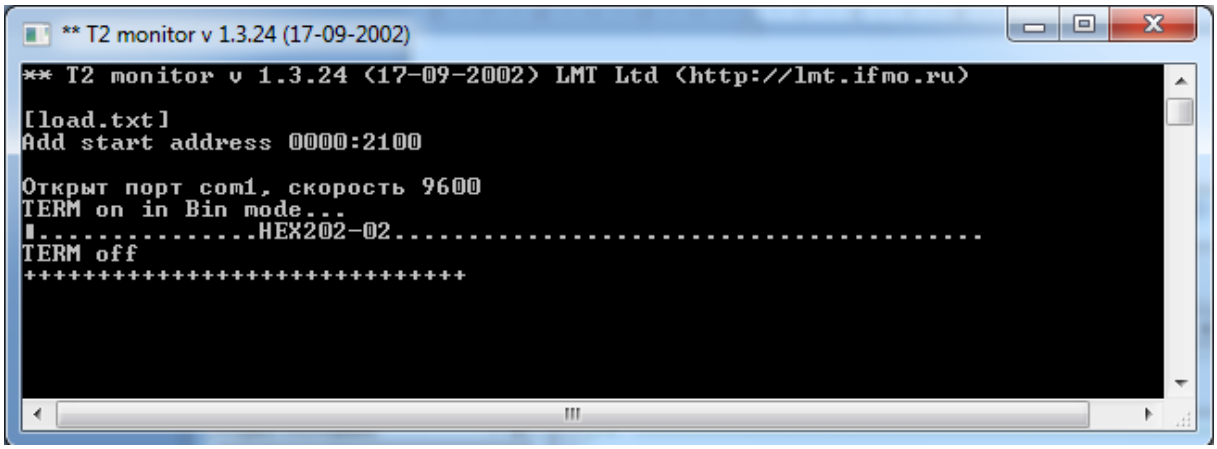

Рис. 28.

После загрузки, приложение должно запуститься (появятся бегущие огни на светодиодах стенда), а T2 – начать выполнять следующую команду, а именно 0 term, что приведёт к отображению строки “Hello Word”, которая передаётся по последовательному интерфейсу (см. рис. 29).

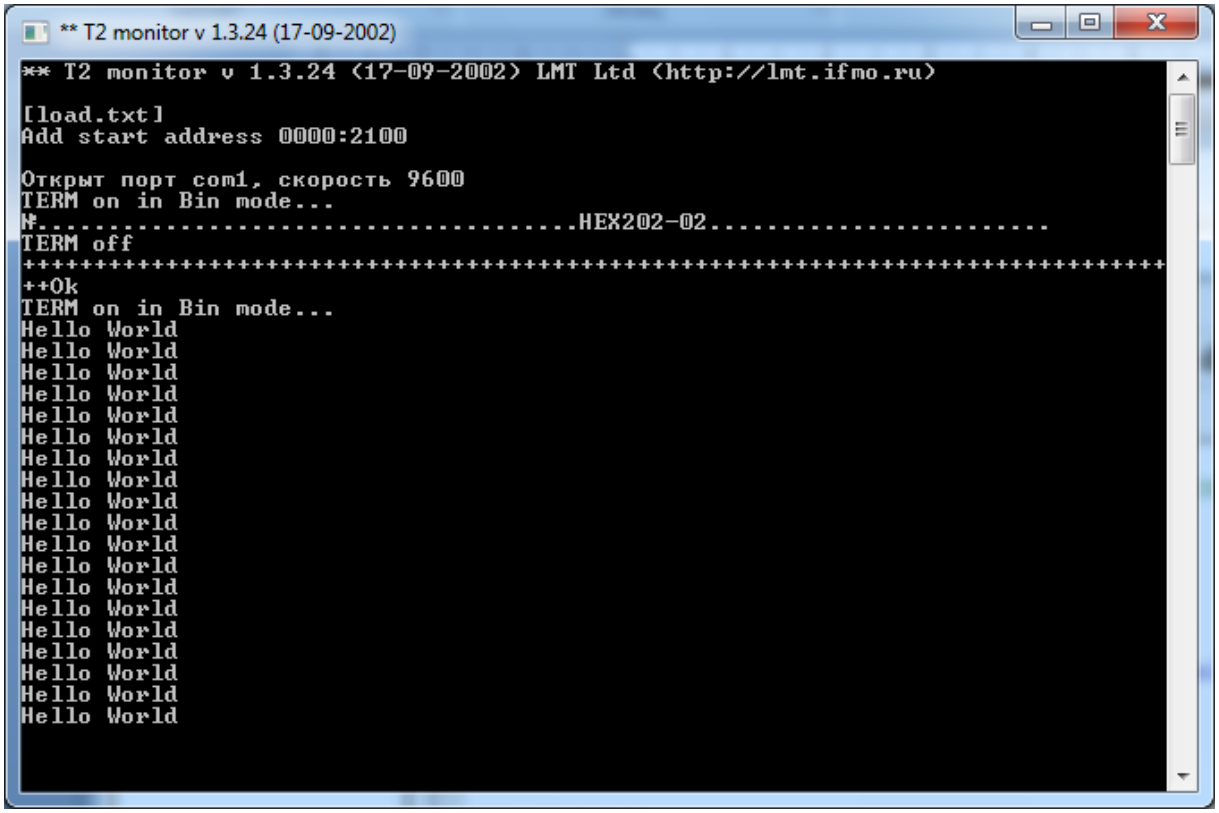

Рис. 29.

При следующем нажатие клавиши Esc окно должно закрыться (приложение T2 выгрузится).

4.6 Возможные трудности при загрузке программы в SDK-1-1

В пункте 2 не появляется строка HEX202……

Возможные причины: учебно-лабораторный стенд подключен к другому порту.

Методы устранения: в командном файле load.txt исправить строку открытия последовательного интерфейса на 9600 openchannel comN, где N-номер последовательного порта ввода-вывода, который можно узнать в диспетчере устройств Windows

В других случаях обратится к преподавателю.

5 Содержание отчёта

Отчёт по лабораторной работе должен быть оформлен в соответствии с требованиями СТП ТПУ. Содержание отчёта:

1. Цель работы.
2. Цикл разработки приложения в Keil uVision (структурная схема с кратким описанием).
3. Основные этапы программирования учебно-лабораторного стенда SDK-1-1.
4. Текст Вашей первой программы для SDK-1-1.
3. Выводы по проделанной лабораторной работе.

ПЕРЕЧЕНЬ ИСТОЧНИКОВ

1 Руководство пользователя интегрированной среды разработки Keil uVision
L:\Keil\C51\HLP\gs51.pdf

2. Руководство пользователя компилятора Cx51
L:\Keil\C51\HLP\C51.pdf

3 Учебный стенд SDK-1-1. Руководство пользователя
L:\Study\МПС\SDK_1_1\DOC\sdk11_userm_v1_0_7.pdf

4 Руководство пользователя компилятора ассемблера Ax51
L:\Keil\C51\HLP\A51.pdf

5 Операционная система реального времени RTX51
L:\Keil\C51\HLP\tr51.pdf

6 SDK-1.1 Demonstration Set. Руководство пользователя.
L:\Study\МПС\SDK_1_1\DemoSet\doc\DemoSetUG.pdf