Изучение интегрированной среды разработки CodeVisionAVR (CVAVR)

2.1. Общее описание среды проектирования

CodeVisionAVR — это интегрированная среда разработки, объединяющая кросс-компилятор языка С, удобные инструменты организации проекта, ручного ввода и автоматической генерации исходных текстов программ на языке С, которая позволяет выполнять полный цикл создания управляющих программ для широкой гаммы моделей микроконтроллеров семейства Atmel AVR.

Встроенный кросс-компилятор языка программирования С реализует практически все элементы стандарта ANSI С, которые допускает архитектура ядра AVR, а также некоторые дополнительные возможности, призванные в наиболее полной мере использовать особенности архитектуры этого семейства МК и интегрированных системных решений на их основе.

Общий вид графической оболочки среды проектирования приведён на рис. 2.1.

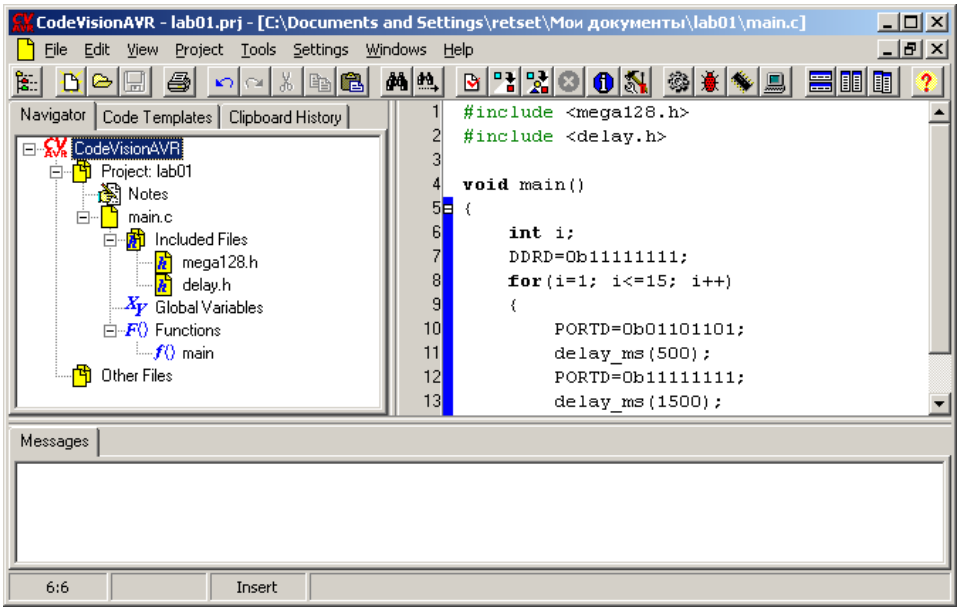


Рис. 2.1 — Интерфейс среды CodeVisionAVR версии 1.25 (Evaluation).

Интерфейс основного окна включает в себя строку меню, панель инструментов, навигатор по проекту (с дополнительными вкладками программных заготовок и истории буфера обмена) область редактора исходного кода и область сообщений компилятора.

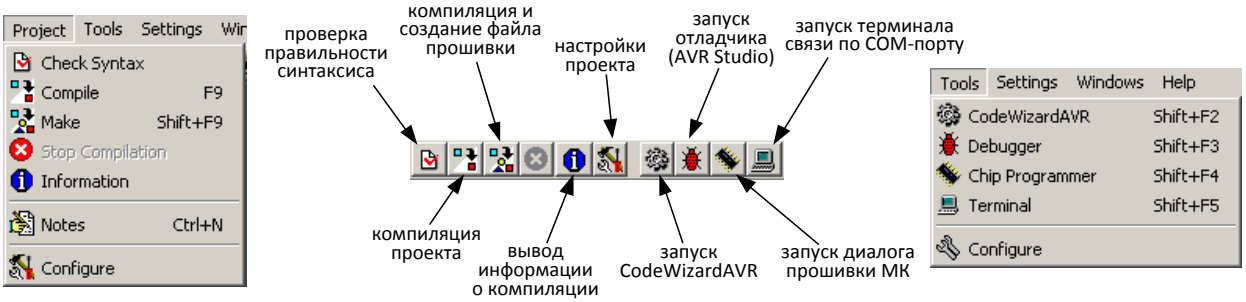


Рис. 2.2 — Основные рабочие меню и панель инструментов среды CodeVisionAVR.

Основные инструменты среды и операции по работе с проектом можно вызвать из меню **Project** и **Tools**, а также с помощью соответствующих кнопок на панели инструментов (рис. 2.2).

2.2. Настройка среды CodeVisionAVR и подготовка к работе с макетом ML-1

Перед началом создания проекта и написанием программы в соответствии с заданием лабораторной работы необходимо произвести настройку среды CodeVisionAVR для работы с макетом ML-1.

Основными этапами настройки являются:

1) настройка редактора исходного кода, шрифта и цветов подсветки синтаксиса (меню **Settings→Editor**);
2) выбор типа программатора, который будет выполнять загрузку файла прошивки в МК отладочного комплекса ML-1 (меню **Settings→Programmer**);
3) настройка отладчика: указание пути к исполняемому файлу среды разработки Atmel AVR Studio (меню **Settings→Debugger**);
4) настройка встроенного терминала по работе с последовательным интерфейсом COM-порта ПК (меню **Settings→Terminal**).

Настройки в пп. 1 и 3 не являются обязательными по обеспечению базовой работоспособности лабораторной установки, поэтому могут быть выполнены самостоятельно, а настройка в п. 4 рассматривается перед выполнением лабораторной работы по ознакомлению с работой последовательного интерфейса USART.

Для настройки соединения между средой разработки CodeVisionAVR и макетом ML-1 в меню **Settings** необходимо выбрать пункт **Programmer** и, после того как появится диалоговое меню с различными видами аппаратных устройств для программирования, необходимо выбрать пункт «**Kanda Systems STK200+/300**» (рис. 2.3).

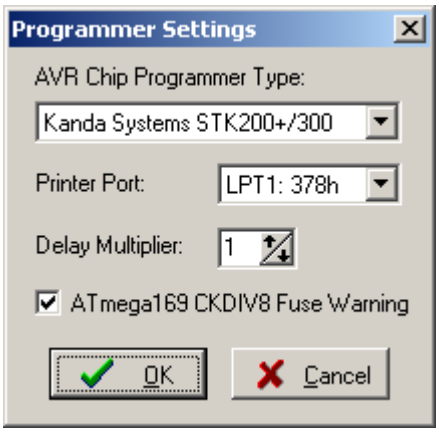


Рис. 2.3 — Диалоговое окно выбора программатора

Это программатор, работающий через параллельный порт LPT, поэтому в графе **Printer Port** необходимо убедиться, что выбран номер порта LPT, к которому подключен программатор отладочного комплекса (LPT1).

Настройку программатора можно выполнить однократно, — в дальнейшем контроль правильности выбранного программатора можно осуществлять по заголовку диалогового окна программирования (рис. 2.4).

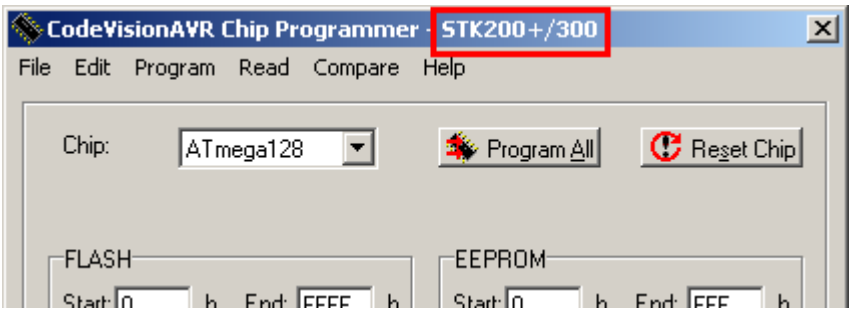


Рис. 2.4 — Отображение текущего выбранного программатора

После выполнения всех необходимых настроек можно приступать к созданию нового проекта в среде CodeVisionAVR и выполнения поставленного лабораторного задания.

2.3. Создание нового проекта в среде CodeVisionAVR

Для создания нового проекта в среде CodeVisionAVR в меню **File** необходимо выбрать пункт **New** (рис. 2.5а) или нажать на соответствующей кнопке панели инструментов, после чего появится диалоговое окно выбора нового объекта: файла исходного текста (**Source**) или проекта (**Project**) (рис. 2.5б).

После выбора пункта **Project** появляется диалоговое окно, в котором среда проектирования предлагает воспользоваться автоматическим генератором исходного кода **CodeWizardAVR** (рис. 2.5в). Это средство является дополнительной функцией среды CodeVisionAVR, которая позволяет при помощи мастера с графическим интерфейсом выполнить настройку любых встроенных ресурсов выбранного МК, а также настроить МК для работы с наиболее часто используемым периферийным оборудованием: графическими и цифробуквенными ЖКИ, термодатчиками, внешним ОЗУ и т.п. Результатом работы CodeWizardAVR является исходный код на языке С, который содержит все выбранные настройки. Он вставляется в основной файл исходного кода проекта и служит основой для проведения дальнейшей разработки.

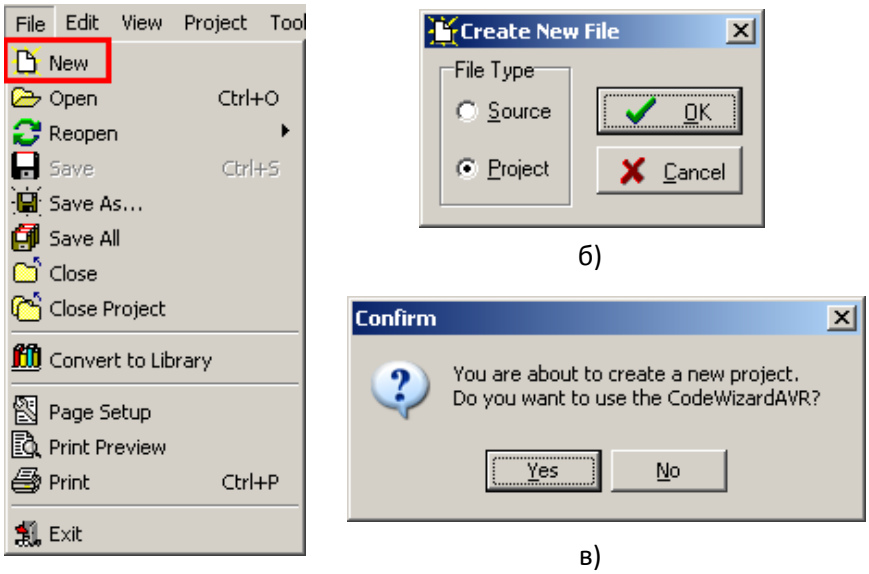


Рис. 2.5 — Процесс создания нового проекта в среде CodeVisionAVR

Кроме автоматического запуска в начале создания проекта мастер CodeWizardAVR доступен в любое время написания исходного кода проекта: его можно вызвать из меню Tools или нажатием соответствующей кнопки на панели инструментов (рис. 2.2).

Далее будут рассмотрены оба варианта создания нового проекта: как с использованием мастера CodeWizardAVR, так и создание пустого проекта без его участия.

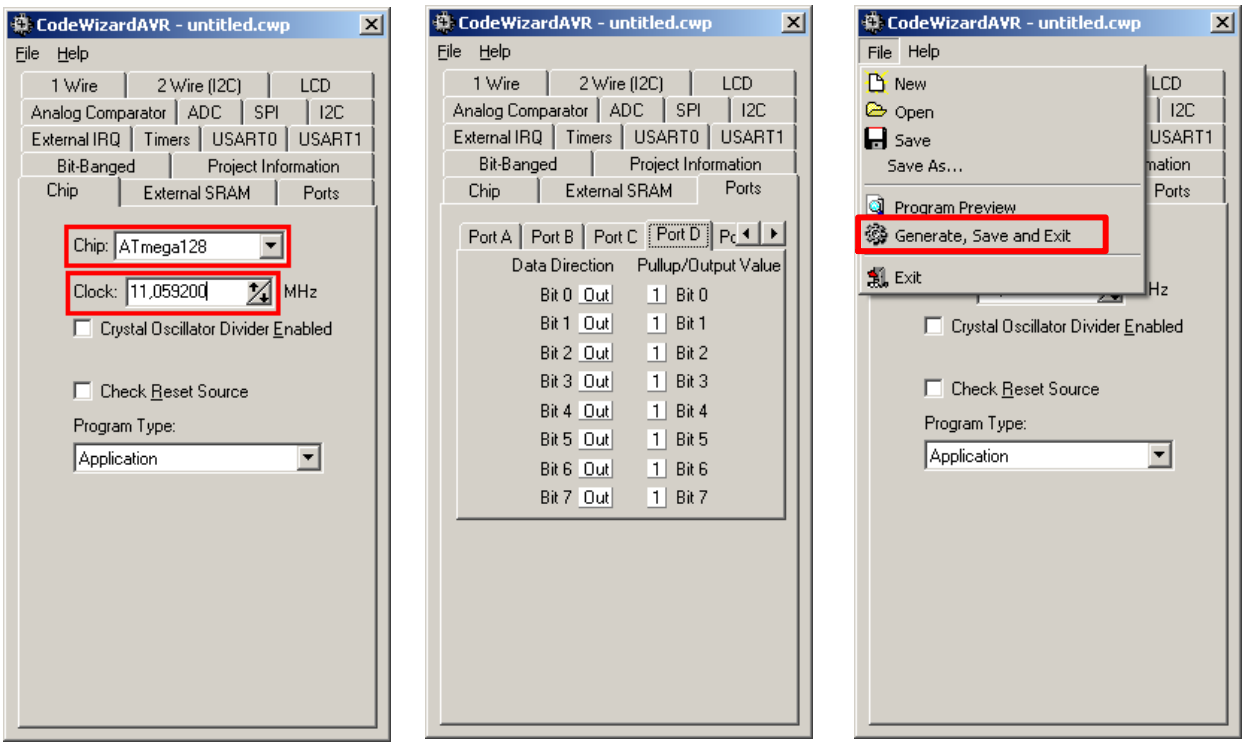
2.3.1. Создание проекта с использованием мастера CodeWizardAVR

При согласии на использование мастера CodeWizardAVR (рис. 2.5в) откроется диалоговое окно, показанное на рис. 2.6а.

На вкладке **Chip** в выпадающем списке **Chip:** необходимо выбрать модель МК ATmega128, а в поле **Clock:** вписать тактовую частоту задающего кварцевого резонатора, установленного на плате отладочного комплекса, которая равна 11,0592 MHz (рис. 2.6а).

При необходимости можно выполнить настройку дополнительных ресурсов МК: портов ввода/вывода, таймеров/счетчиков, приемопередатчиков USART, TWI, SPI и пр. Например, на рис. 2.6б изображена настройка порта D на вывод, с начальной инициализацией всех выводов на высокий логический уровень («1»).

После этого необходимо сгенерировать исходный код, который реализует все выбранные настройки в виде инструкций языка С, и сохранить его в файл исходного кода проекта. Для этого необходимо в меню мастера настройки **File** выбрать пункт **Generate, Save and Exit** (рис. 2.6в). Предыдущий пункт меню **Program Preview** позволяет просмотреть итоговый исходный код с настройками в отдельном окне до его сохранения в файл.



а) б) в)

Рис. 2.6 — Работа с мастером начальной настройки CodeWizardAVR

После выбора пункта **Generate, Save and Exit** мастер предлагает подряд сохранить несколько файлов, связанных с создаваемым проектом: файл исходного

кода (*.c), файл настроек проекта (*.prj) и файл ресурсов CodeWizardAVR (*.cwp). Прежде чем начать сохранять файлы, настоятельно рекомендуется создать в файловой системе ПК необходимые каталоги: в рабочей папке студенческих работ (обычно, это папка **students**, расположенная в **C:\Documents and settings\<username>\Мои документы** или на диске **D:**) необходимо создать папку с названием группы, в ней — папку с названием предмета, в ней — папку с фамилией исполняющего лабораторную работу, в ней — папку с названием лабораторной работы, т.е. путь к файлам создаваемого проекта должен быть похож на: **%students%\AP10a\MCDev\lab01**. При подборе имен рекомендуется избегать использования кириллических символов и пробела.

После сохранения всех файлов проекта можно приступать непосредственно к написанию рабочей программы. Заготовка, создаваемая с помощью мастера автоматической генерации кода CodeWizardAVR содержит инициализацию всех регистров ввода/вывода для всех ресурсов МК. Для уменьшения итогового размера управляющей программы и облегчения читаемости исходного кода описания и инициализации тех регистров, которые не используются в рабочей программе, рекомендуется удалить.

2.3.2. Создание нового проекта без использования мастера CodeWizardAVR

Если при создании нового проекта отказаться от использования мастера, то среда проектирования предложит сразу сохранить файл создаваемого проекта (*.prj) и откроет диалоговое окно настроек проекта (рис. 2.7), которое можно позже в любой момент вызвать из меню (**Project→Configure**) или соответствующей кнопкой на панели инструментов (рис. 2.2).

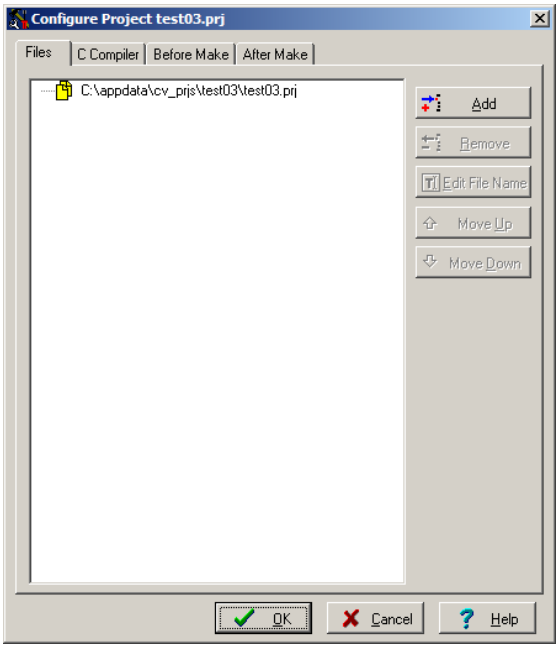


Рис. 2.7 — Диалоговое окно свойств проекта

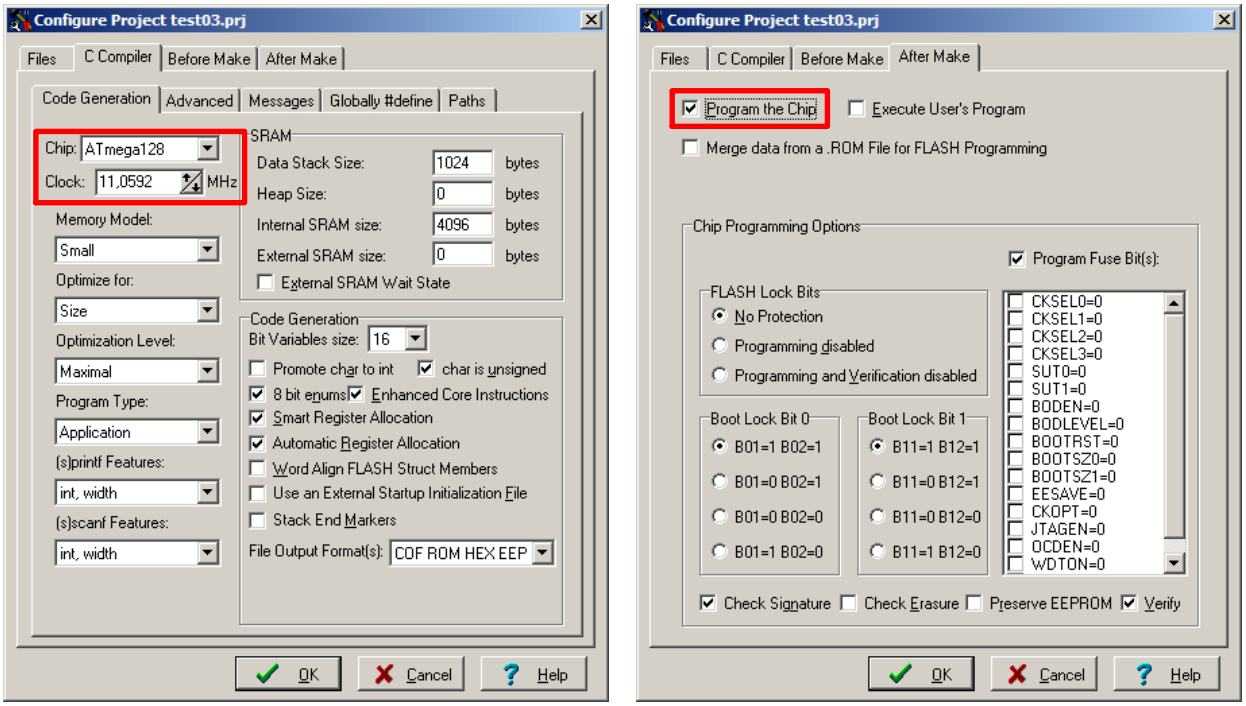
Как и в предыдущем случае, рекомендуется предварительно в файловой системе ПК создать все необходимые каталоги согласно рекомендациям, приведённым ранее.

На вкладке **Files** диалогового окна свойств проекта отображается файловая структура текущего проекта. По умолчанию во вновь создаваемом проекте файлы исходного текста отсутствуют: их нужно создавать отдельно и вручную с помощью кнопок в правой части окна добавлять в проект.

Прежде чем приступить к написанию рабочей программы следует выполнить настройку параметров проекта на вкладках **C Compiler** и **After Make** (рис. 2.8).

Вкладка **C Compiler** содержит несколько вложенных вкладок с настройками параметров работы компилятора языка С. На этой вкладке следует обязательно задать модель используемого МК в выпадающем списке **Chip** и тактовую частоту его работы в поле **Clock** (рис. 2.8а).

На вкладке **After Make** задаются события, которые должны произойти после успешной сборки итогового файла прошивки МК. Опция **Program the Chip** добавляет кнопку в диалоговое окно результатов сборки проекта, которая позволяет непосредственно запрограммировать МК после успешной сборки файла прошивки.



а) б)

Рис. 2.8 — Диалоговое окно настройки проекта

Для создания файла с исходным кодом необходимо воспользоваться меню или соответствующей кнопкой на панели инструментов и в появившемся диалоговом окне (рис. 2.5б) выбрать **Source**. Среда разработки создаст файл **untitled.c**, но поместит его в категорию файлов, не относящихся к проекту (рис. 2.9):

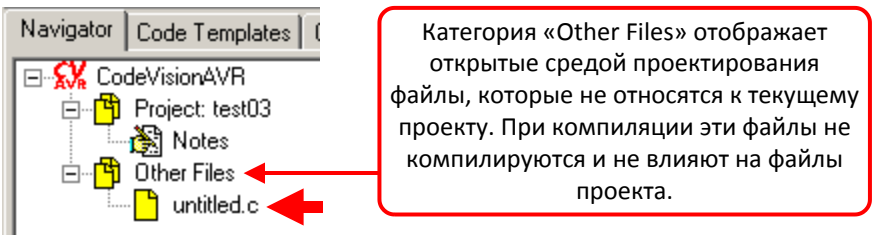
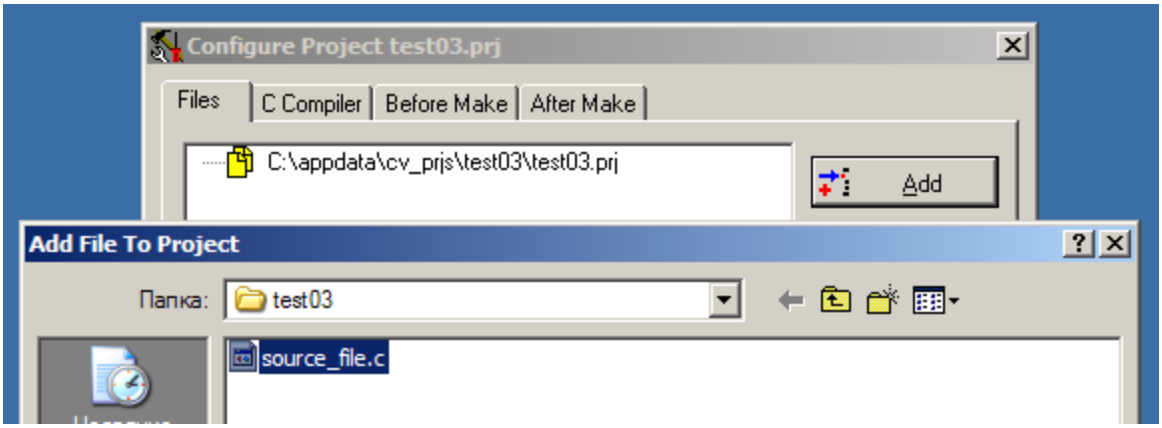


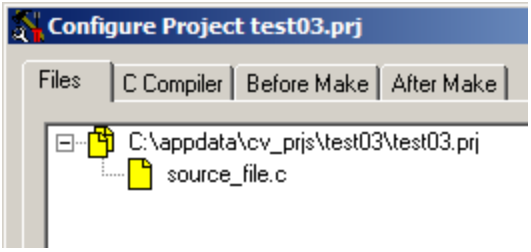
Рис. 2.9 — Структура проекта и категория «Other Files»

Для подключения его к проекту необходимо сохранить файл в папке проекта (меню **File→Save As..**), задав ему желаемое имя, и открыть настройки проекта (рис.

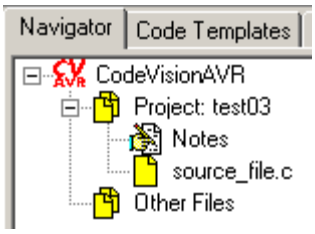
2.7). На вкладке **Files** необходимо нажать кнопку **Add** и выбрать только что сохранённый файл (рис. 2.10а). После этого файл должен отобразиться в дереве проекта как в диалоговом окне настроек проекта на вкладке **Files** (рис. 2.10б), так и в навигаторе по проекту в основном окне среды разработки (рис. 2.10в):



а)



б)



в)

Рис. 2.10 — Подключение файла исходного кода к текущему проекту

После присоединения файла исходного кода к проекту можно приступать к написанию рабочей программы. Описанным способом можно присоединять к проекту и уже существующие файлы исходного кода.

2.4. Компиляция и отладка рабочей программы

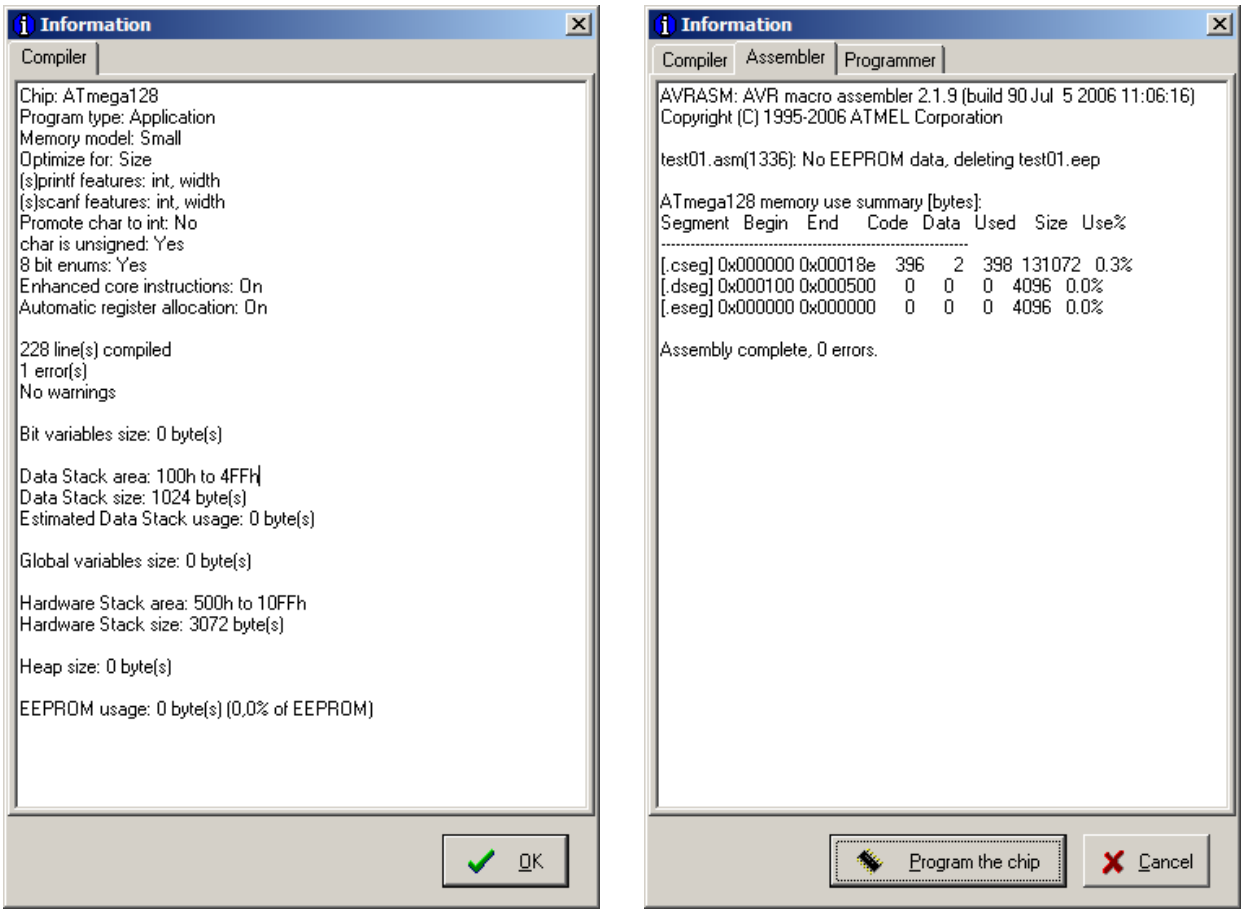
После того как код рабочей программы набран в среде, можно произвести проверку правильности синтаксиса исходного кода, выполнить компиляцию проекта или выполнить финальную сборку файла прошивки. Для этого необходимо воспользоваться меню **Project**, соответствующими кнопками на панели инструментов (рис. 2.2), или нажать соответствующую комбинацию клавиш.

Проверка синтаксиса набранного исходного кода (**Project→Check Syntax**) выполняет проверку правильности выбранных идентификаторов, использованных команд и конструкций языка С и правильность построения структуры программы. Эта операция не производит создание ни файлов ассемблерного кода, ни файлов итоговой управляющей программы МК (так называемых «файлов прошивки»).

Компиляция проекта (**Project→Compile**, клавиша **F9**) помимо проверки синтаксиса выполняет синтез файла с кодом на языке ассемблера AVR (*.asm), а также создает дополнительных служебных файлов. После выполнения попытки компиляции исходных текстов проекта компилятор выдает диалоговое окно отчета (рис. 2.11а), в котором содержатся ключевые настройки проекта, параметры

препроцессора и результаты компиляции: число ошибок и предупреждений, ресурсы МК, зарезервированные для размещения переменных и констант.

Финальная сборка файла прошивки (**Project→Make**, сочетание клавиш **Shift+F9**) помимо проверки синтаксиса и компиляции включает в себя запуск программы ассемблера AVR AVRASM32 и создание финального файла управляющей программы в машинных командах ядра МК — так называемой «прошивки» МК — который будет загружен в память программ МК макета ML-1 с помощью программатора на этапе прошивки. Формат файла может быть задан в настройках проекта. Наиболее распространенным форматом этого файла является формат Intel Hex (*.hex). Результаты сборки отображаются в диалоговом окне отчета (рис. 2.11б), которое помимо данных о компиляции, отображает результаты работы ассемблера и кнопку экспресс-прошивки МК (для её отображения необходимо в диалоговом окне настроек проекта (**Project→Configure**) на вкладке **After Make** отметить опцию **Program the Chip** (рис. 2.8б)).



а) б)

Рис. 2.11 — Диалоговые окна отчетов по результатам компиляции и сборки

Кроме того, на этом этапе создаются файлы отладки в формате COFF и OBJ (*.cof и *.obj), которые используются для отладки разработанной программы с помощью внешнего отладчика (AVR Studio, меню **Tools→Debugger**, кнопка на панели инструментов или комбинация клавиш **Shift+F3**).

Как видно из описания, последний вариант включает в себя все предыдущие, кроме того, каждая последующая операция не будет выполняться, если на предыдущем этапе были обнаружены ошибки.

Если в процессе компиляции были выявлены ошибки, то их перечень можно найти в нижней части графической оболочки в области сообщений **Messages** и в дереве проекта **Navigator** в секции **Errors** (рис. 2.12).

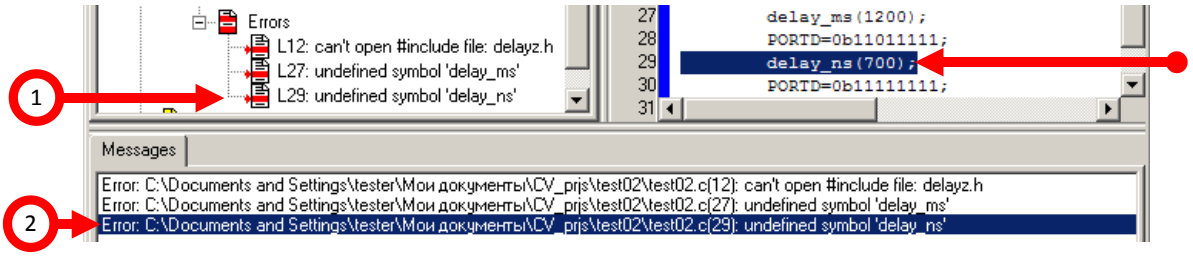


Рис. 2.12 — Навигация по ошибкам и предупреждениям в исходном коде

После успешной компиляции и создания файла прошивки можно приступать к программированию МК и отладке программы с использованием реального оборудования и ресурсов макета.

2.5. Программирование МК с помощью средств CodeVisionAVR

Загрузка файла прошивки осуществляется с помощью мастера **Chip Programmer** (меню **Tools→Chip Programmer**) или экспресс-кнопки **Program the chip** в диалоговом окне результатов сборки файла прошивки (рис. 2.11б). Диалоговое окно этого мастера представлено на рис. 2.13.

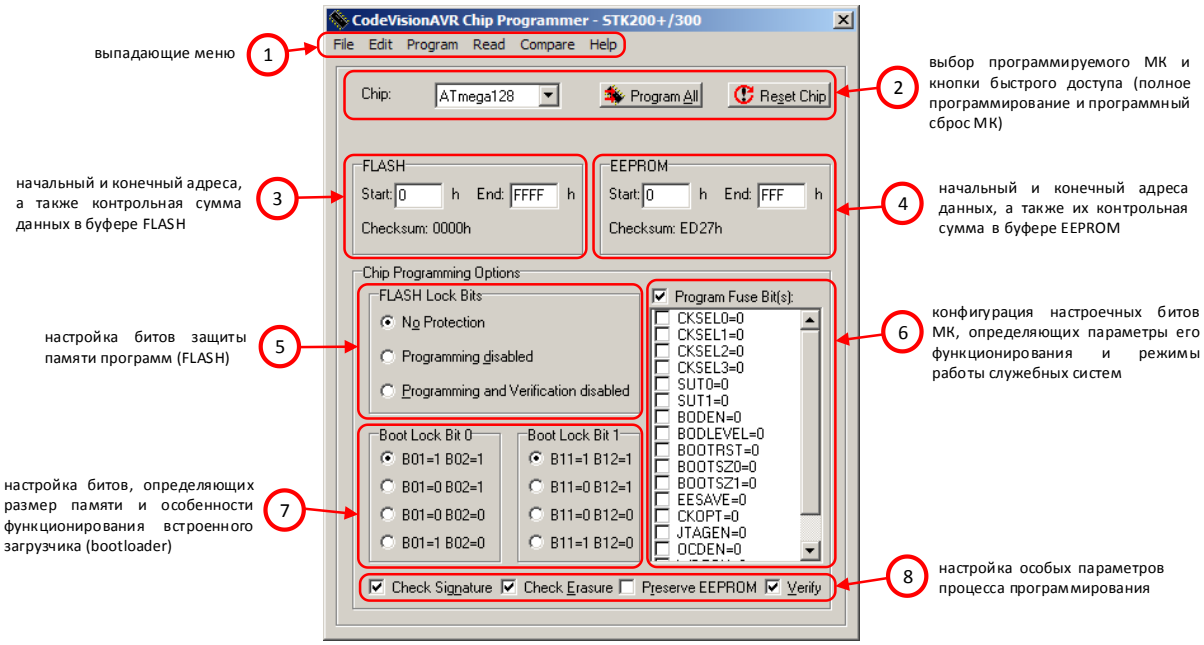


Рис. 2.13 — Диалоговое окно мастера Chip Programmer

Мастер программирования Chip Programmer при запуске организует два промежуточных буфера памяти: буфер памяти программ FLASH и буфер энергонезависимой памяти данных EEPROM. Вся работа мастера с МК осуществляется через эти два буфера.

Если запустить мастер программирования после успешной сборки текущего проекта (меню **Project→Make**, сочетание клавиш **Shift+F9**), то файлы прошивки

памяти программ (FLASH) и содержимого энергонезависимой памяти (EEPROM) будут загружены в буферы памяти автоматически.

С помощью меню **File** можно загрузить в буферы содержимое из любого произвольного файла прошивки (*.rom, *.hex, *.bin) или файла данных EEPROM (*.eep), находящегося в файловой системе ПК, воспользовавшись командами **File→Load FLASH** и **File→Load EEPROM**, а также сохранить содержимое буферов в файлы на ПК с помощью команд **File→Save FLASH** и **File→Save EEPROM**.

Меню **Edit** позволяет вручную изменять содержимое обоих буферов в интерактивном режиме (**Edit→FLASH** и **Edit→ EEPROM**) с помощью встроенного HEX-редактора.

Основным рабочим меню в процессе работы с мастером программирования является меню **Program**. С помощью пунктов этого меню можно выполнить полное стирание памяти программ и данных (пункт **Program→Erase Chip**), проверку очистки памяти программ и данных после стирания (пункт **Program→Blank Check**), выполнить запись содержимого буфера FLASH в память программ МК (**Program→FLASH**), выполнить запись данных из буфера EEPROM в энергонезависимую память МК (**Program→EEPROM**), а также выполнить конфигурацию настроечных (Fuse) и защитных (Lock) битов в специализированных регистрах МК (пункты **Program→Fuse Bits** и **Program→Lock Bits**). Для ускорения и автоматизации процесса программирования есть возможность выполнить одновременную запись программы, данных и служебной информации с помощью меню **Program→All** или кнопки **Program All** напротив выпадающего списка выбора модели МК (рис. 2.13 ②).

Вторым по значимости меню мастера **Chip Programmer** является меню **Read**. С помощью команд этого меню можно считать из МК в буфер мастера содержимое памяти FLASH или EEPROM (команды **Read→FLASH** и **Read→EEPROM**), определить сигнатуру МК — уникальный код, идентифицирующий модель МК подключённого к программатору (команда **Read→Read Chip Signature**), считать текущие значения настроечных (Fuse) и защитных (Lock) битов в специализированных регистрах МК (пункты **Read→Fuse Bits** и **Read→Lock Bits**), а также значение байтов калибровки системы тактирования МК (пункт **Read→Calibration Byte(s)**).

Перед началом работы с мастером программирования следует убедиться в правильности выбора программатора (тип выбранного программатора отображается в заголовке окна, как показано на рис. 2.4, при необходимости сменить тип программатора так, как это описано в разделе 2.2) и в правильности выбора модели МК в выпадающем списке **Chip** (рис. 2.13 ②).

Команды меню **Compare** позволяют выполнить сравнение содержимого буферов FLASH и EEPROM с содержимым соответствующей памяти МК. Использование этих команд позволяет узнать актуальность прошивки микроконтроллера и значений в энергонезависимой памяти EEPROM.

Ниже в окне мастера находятся поля ввода для указания размерности памяти программ и энергонезависимой памяти данных (рис. 2.13 ③ и ④). При выборе модели МК в выпадающем списке и загрузке файлов прошивки в соответствующие буферы эти значения устанавливаются автоматически.

В нижней секции находятся настройки управляющих и блокирующих битов (**Fuse Bits, FLASH Lock Bits** и **Boot Lock Bits**, рис. 2.13 ⑤, ⑥ и ⑦), а также настройки процесса программирования (рис. 2.13 ⑧).

Внимание! Неграмотная настройка настроечных и защитных битов может привести к некорректной работе МК или полному блокированию его связи с программатором. Поэтому самостоятельное изменение каких-либо настроек, связанных с настроечными или защитными битами без разрешения преподавателя ЗАПРЕЩЕНО!

В число настроек процесса программирования входят опции:

- проверка сигнатуры МК перед началом записи данных или значений конфигурационных битов в МК (**Check Signature**);

- проверка очистки памяти программ МК (**Check Erasure**);

- защита данных, содержащихся в памяти EEPROM, от стирания и перезаписи (**Preserve EEPROM**);

- проверка содержимого памяти программ МК после прошивки (**Verify**).

В меню **Help** находится ссылка на пункт раздела встроенной в среду CodeVisionAVR справки о мастере программирования **Chip Programmer**.

Традиционная процедура записи прошивки в МК выглядит следующим образом:

1) после успешной сборки текущего открытого проекта (или при наличии ранее сформированного файла прошивки с расширением ***.rom**, ***.bin**, ***.hex** для записи в память FLASH или файла данных с расширением ***.eep** для записи в память EEPROM) открыть мастер программирования **Chip Programmer**;

2) убедиться, что выбран правильный тип программатора (см. п. 2.2 и рис. 2.4) и связь с выбранным МК установлена (воспользоваться командой **Read→Read Chip Signature** мастера программирования);

3) если происходит запись сторонних файлов прошивки и/или файла данных, выполнить загрузку этих файлов в соответствующие буферы с помощью команд мастера **File→Load FLASH** и **File→Load EEPROM**, если происходит запись файлов прошивки текущего открытого проекта, то на текущем шаге ничего делать не надо;

4) убедиться, что буферы FLASH и EEPROM заполнены данными для записи по значениям полей **Start**, **End** и **Checksum** в секциях **FLASH** и **EEPROM** диалогового окна мастера программирования (рис. 2.13 ③ и ④) (для МК ATmega128 значения в полях **End** должны отличаться от FFFF для секции **FLASH** и от FFF для секции **EEPROM**, а значения в полях **Checksum** не должны равняться 0000 в секции **FLASH** и F00 — в секции **EEPROM**);

5) выполнить очистку памяти программ МК с помощью команды **Program→Erase Chip**; если в настройках процесса программирования не установлена опция **Preserve EEPROM**, то будет очищена и энергонезависимая память данных; для ускорения процесса стирания можно отключить опцию проверки очистки памяти программ после стирания (**Check Erasure**) и проверку содержимого памяти FLASH и EEPROM после программирования (**Verify**);

6) выполнить индивидуальное программирование памяти программ FLASH и/или памяти данных EEPROM соответствующими командами из меню мастера **Program**; при необходимости можно воспользоваться кнопкой **Program All** (команда **Program→All**), нажатие на которую приводит к стиранию и проверке очистки памяти FLASH и EEPROM, поочередной прошивке содержимого буфера FLASH в память программ и содержимого буфера EEPROM в энергонезависимую память данных, а также записи настроечных (Fuse) и защитных (Lock) битов в соответствующие регистры МК; БУДЬТЕ ВНИМАТЕЛЬНЫ: перед применением этой команды следует убедиться что настроечные (Fuse) и защитные (Lock) биты выставлены правильно, в противном случае прошивка некорректных значений может привести к неправильному функционированию зашитой программы и блокированию памяти контроллера!

По умолчанию рекомендуется выполнение команд индивидуальной прошивки памяти программ (**Program→FLASH**), а также памяти данных (**Program→EEPROM**), если это необходимо.

3 УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

3.1 Общий алгоритм выполнения лабораторной работы

1) Убедиться в правильности подключения источника питания и шлейфа программатора к лабораторному макету. Включить лабораторный макет (установить выключатель электропитания в положение I, у макета должна загореться подсветка ЖКИ-дисплея).

2) Запустить среду разработки CodeVisionAVR.

3) Создать новый проект в соответствии с указаниями в разделе 2.

4) Создать файл для исходного кода программы (на этом этапе можно воспользоваться мастером CodeWisard) и включить его в состав файлов проекта (мастер CodeWisard подключает файл к проекту автоматически).

5) Ввести код программы в соответствии с вариантом задания.

6) Выполнить компиляцию программы (**Project→Compile**, клавиша **F9**, см. раздел 2.4). Устранить ошибки, выявленные на данном этапе и повторить компиляцию.

7) Создать загрузочный модуль программы (**Project→Make**, сочетание клавиш **Shift+F9**, см. раздел 2.4) и выполнить программирование микроконтроллера.

8) Запустить мастер **Chip Programmer** (см. раздел 2.5). Настроить параметры программатора и выполнить прошивку МК.

9) Проверить работоспособность загруженной в микроконтроллер программы и, если поставленное задание выполняется в полном объеме, показать результаты работы преподавателю.

10) В случае некорректной работы разработанной программы (поставленное задание не выполняется или выполняется, но не в полном объеме), выполнить очистку памяти программ МК (пункт **Program→Erase Chip** мастера **Chip Programmer**). Провести анализ алгоритма функционирования программы, настройки используемых ресурсов МК. Устранить логические ошибки и заново проверить функционирование программы на макете, повторно выполнив пункты 6-9 этого плана.