681.51(07)
М - 545

№5109

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра синергетики и процессов управления

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы №1
по дисциплине

МИКРОПРОЦЕССОРНЫЕ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Для студентов направления подготовки
27.03.03 Системный анализ и управление

Таганрог 2016

УДК 081.51.01(07.07)

Составитель: Скляров А.А.

Методические указания к выполнению лабораторных работ №1 по дисциплине «Микропроцессорные информационно-управляющие системы». – Таганрог: Изд-во ЮФУ, 2016. -13 с.

Представлен лабораторный практикум по ряду базовых разделов дисциплины «Микропроцессорные информационно-управляющие системы». К каждой работе прилагаются краткие теоретические сведения и рекомендации по использованию среды разработки IAR Embedded Workbench.

Указания предназначены для студентов направления подготовки 27.03.03 Системный анализ и управление.

Лабораторная работа № 1
Создание и компиляция первого проекта в среде IAR. Написание простейшей программы на языке С.

Целью данной работы является ознакомление с одной из сред разработки программного обеспечения для микроконтроллеров IAR Embedded Workbench IDE.

Введение

Для написания программ, которые потом работают в микроконтроллерах, в настоящее время в основном используются персональные компьютеры или ноутбуки. В силу определённых причин мы выбираем среду разработки IAR. Адрес в Интернете для загрузки оценочной версии http://supp.iar.com/Download/SW/?item=EWARM-EVAL

Компания IAR бесплатно предлагает для ознакомления две версии своего продукта:

- версию evolution с полным функционалом и ограничением времени использования 30 дней
- версию kickstart (в имени дистрибутива есть буквы KS) с ограничением на размер генерируемого исполняемого кода), но без ограничения времени использования.

Для выполнения цикла лабораторных работ по дисциплине «Микропроцессорные информационно-управляющие системы» размера кода в 32К более чем достаточно. Поэтому мы рекомендуем загрузить и установить именно эту версию.

После того, как среда программирования IAR будет установлена можно приступить к созданию первой программы для микроконтроллера STM32F429ZI или любого другого с ядром Cortex-M3.

Создание нового проекта

Создадим новый проект Project => Create New Project.

Выбираем шаблон проекта (ProjectTemplates): C-main (рис. 1)

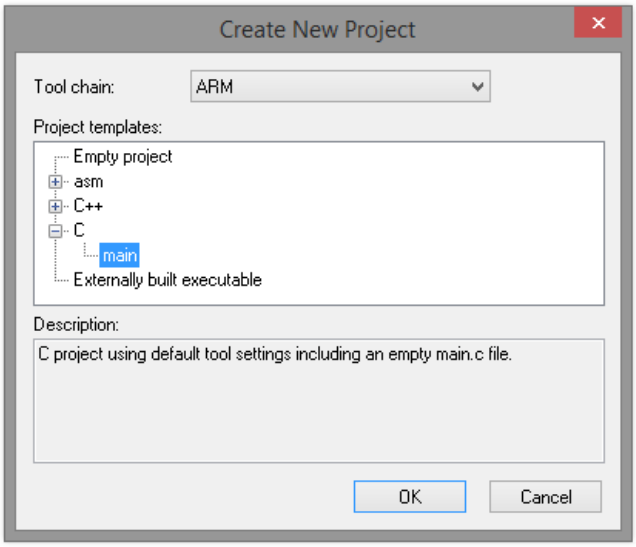

Рисунок 1. Диологовое окно создания новного проекта.

Сохраняем проект под именем Lab1.

Далее в свойствах проекта выбираем модель микроконтроллера STM32F429ZI (рис. 1 – 4). Для этого в окне Workspace правой кнопкой мыши щелкаем по названию созданного проекта (Lab1-Debug) и выбираем Options. Далее в окне «Options for nod “Lab1”» выбираем GeneralOptions и на закладке «Target» в графе «Device» выбираем необходимую модель микроконтроллера.

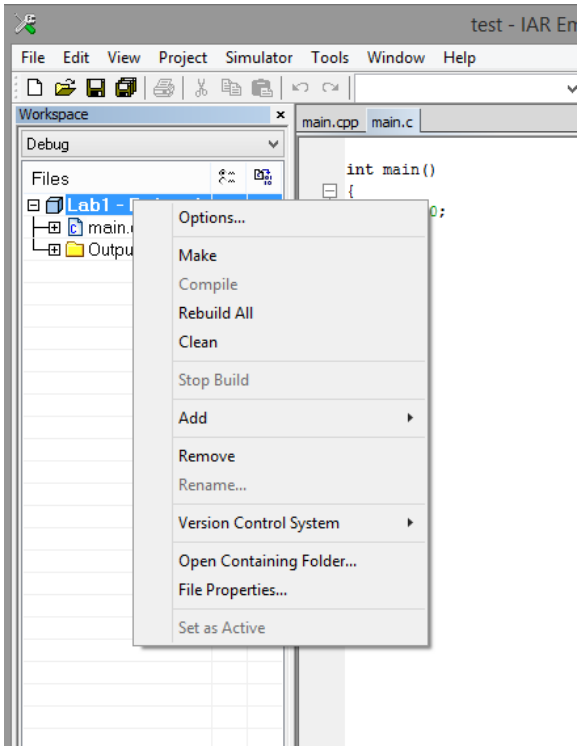

Рисунок 2. Выбор Options для проекта из окна Workspace.

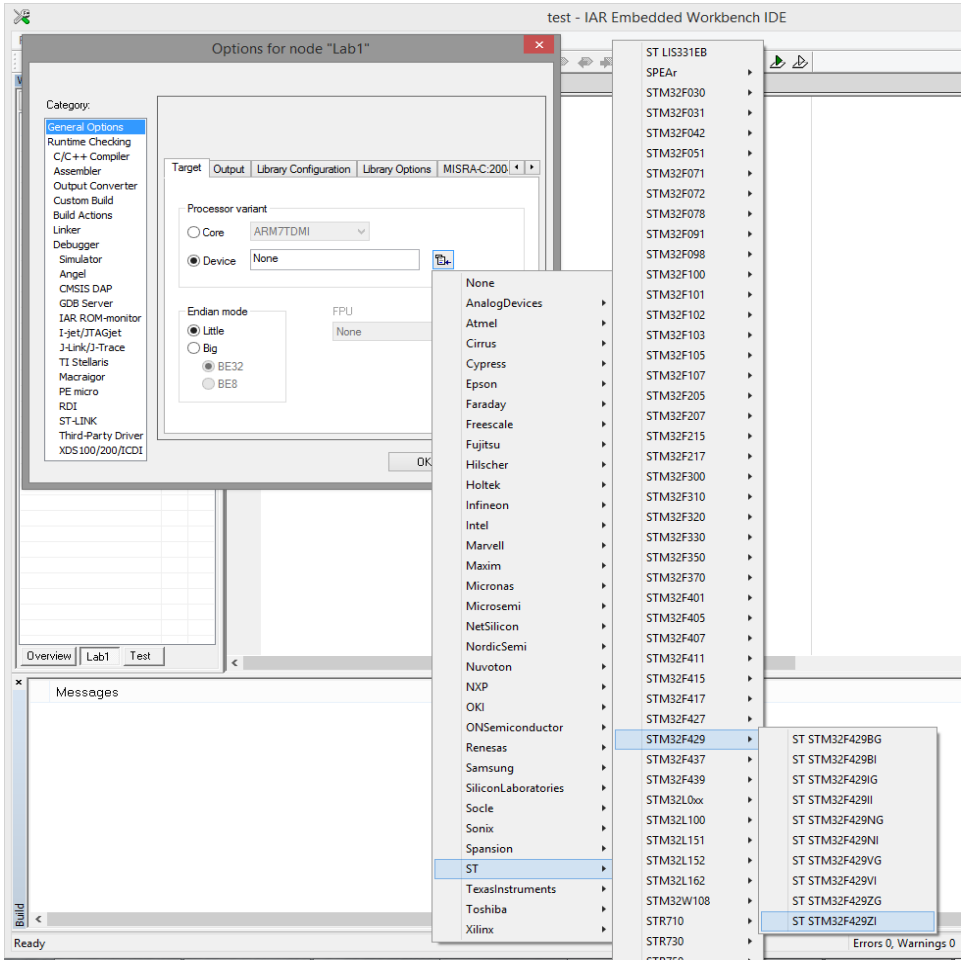

Рисунок 3. Выбор микроконтроллера STM32F429ZI.

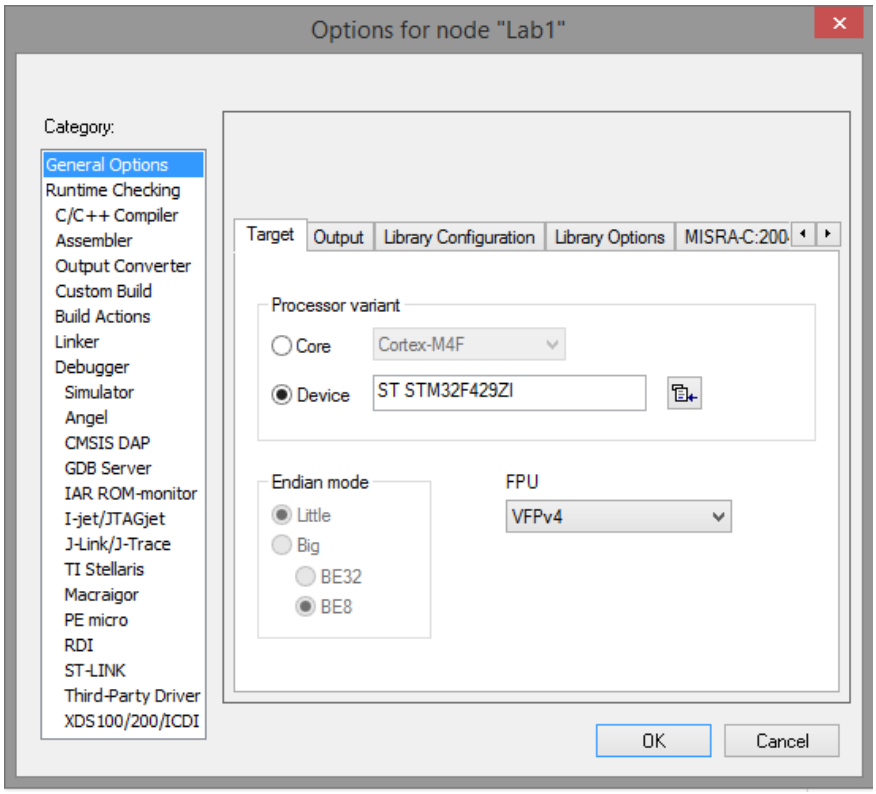

Рисунок 4. Окно Options для проекта Lab1.

В случае, если в списке отсутствует STM32F429ZI, то временно можно воспользоваться готовыми примерами проектов IAR для ARM. Для первого знакомства со средой разработки этого будет вполне достаточно.

Можно отрыть готовый проект evaluator7t из среды разработки IAR и выбрать микроконтроллер ST STM32F100xE. Поскольку у обоих микроконтроллеров одно и то же ядро: Cortex-M3, то наша программа, отлаженная в симуляторе для ST STM32F100xE , будет работать и на микроконтроллере STM32F429ZI.

Чтобы выбрать в качестве средства отладки Simulator нужно щёлкнуть мышью по категории Debugger и в открывшемся окне выбрать Simulator.

Далее, щелкнув мышью по категории C/C++ Compiler, выбрать вкладку Optimization и поставить отметку None (рис. 5).

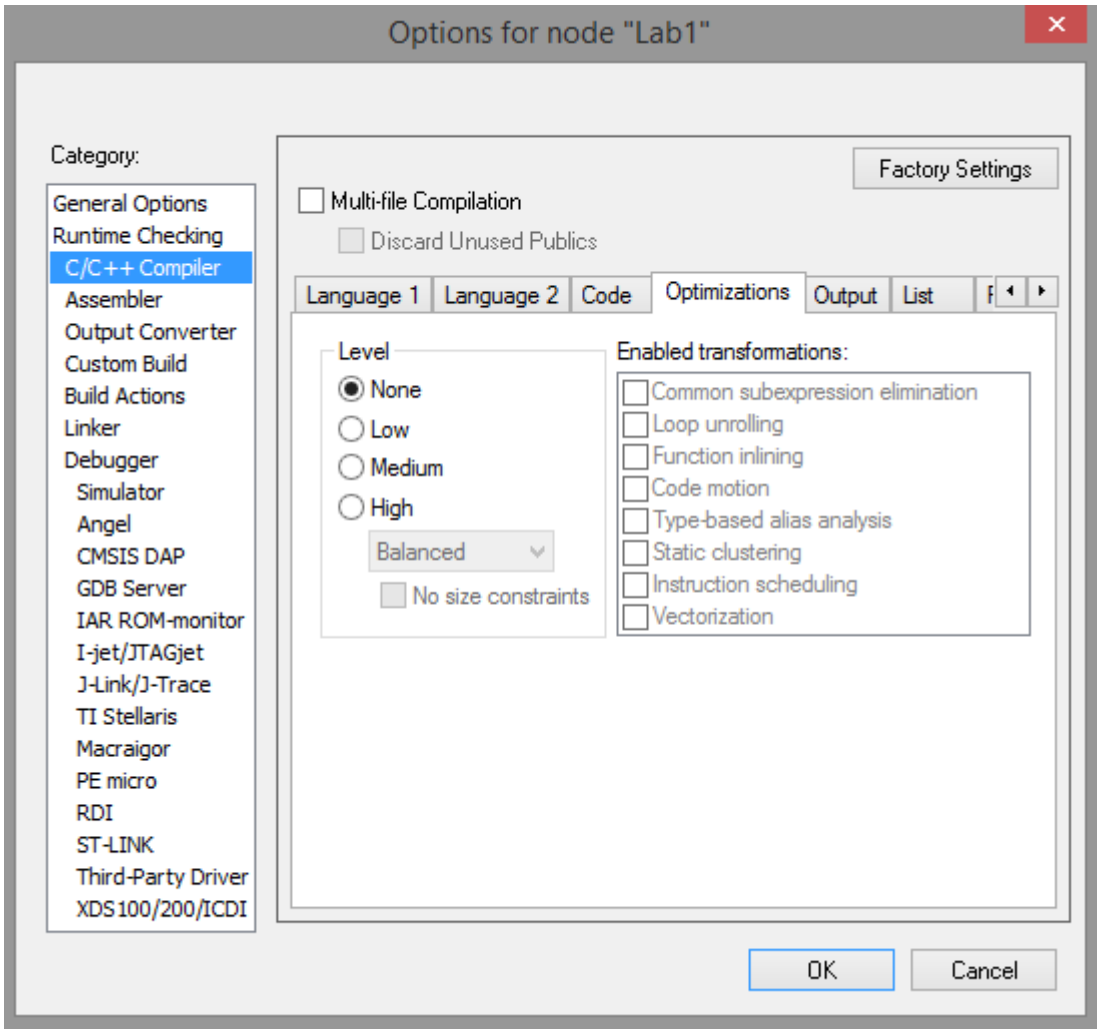

Рисунок 5. Вкладка Optimization окна Options.

Разработка первой программы для микроконтроллера

Исходный текст функции main() на языке С:

В открытом окне main введем следующий код

```
extern int func1(void); // прототип функции на языке C

int main(void)
{
    int k;
    k = func1();

    return k;
}// main
```

Для того чтобы создать новый файл с именем func1.c необходимо выбирать File => New => File (рис. 6) или нажать сочетание клавиш Ctrl+N.

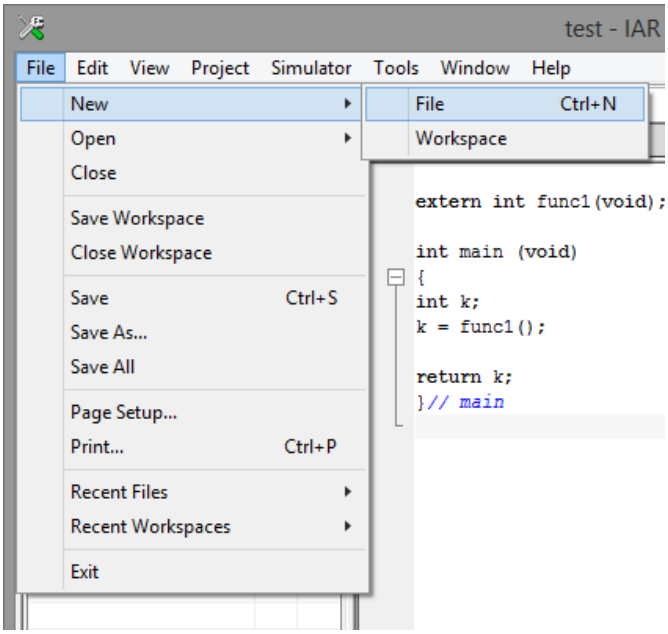

Рисунок 6. Создание нового файла проекта.

Исходный текст функции func1 на языке C:

В окне редактирования файла введем код

```
int func1(void)
{
    int i = 2;
    int k = 5;
    int x = k;
    x = x + i;
    return x;
}
```

Сохраним данный файл как func1.c и добавим его к проекту (рис. 7).

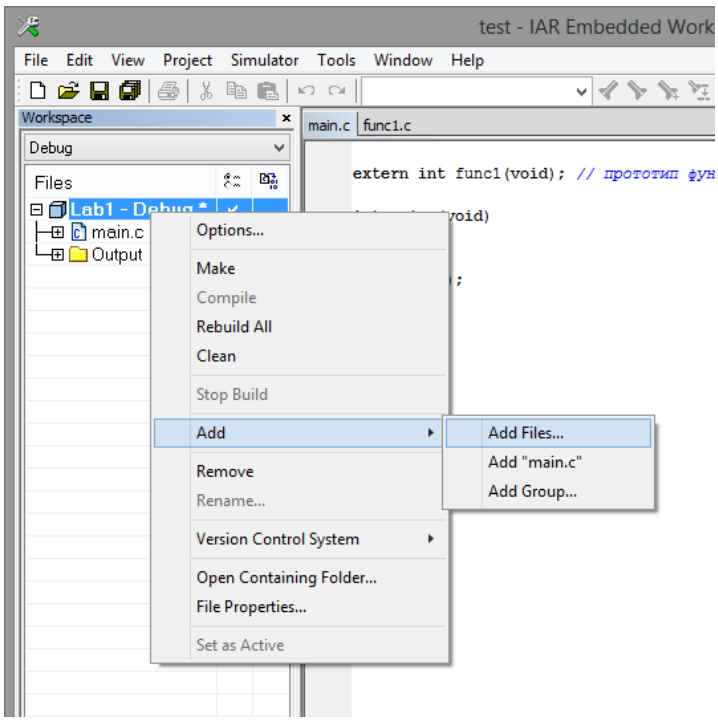

Рисунок 7. Добавление нового файла к проекту.

Запустим проект, нажав на кнопку  или Project => Download and Debug. По умолчанию откроется окно дизассемблера (Disassembly) и среда программирования перейдет режим отладки программы. В окне Disassembly представлена исходная программа на язык ассемблер для микроконтроллеров с ядром Cortex-M3 (рис. 8).

Ассемблер – самый древний язык программирования. Первые программы для первых цифровых вычислительных машин писались в машинных кодах. Вместо двоичной системы счисления чаще пользуются записью программы в шестнадцатиричной системе счисления, что в прочем, кардинально существа дела это не меняет. Всё равно написание программы в машинных кодах было очень трудоёмким занятием, поэтому машинный код решили заменить на представление инструкций мнемокомандами, несущими какую-то смысловую нагрузку.

Язык ассемблер – по сути, система записи машинного кода в виде мнемокоманд. В так называемом листинге, специальном текстовом файле с расширением lst, можно увидеть таблицу, где в первой колонке располагаются адреса ячеек памяти, во второй машинный код команд, т.е. содержимое этих ячеек, а в третьей символьное обозначение команд – мнемокод – исходный текст на языке ассемблер.

Главный недостаток ассемблера – это по-прежнему достаточно высокая трудоёмкость написания программ и относительно длительное время овладения навыками программирования на этом языке, точнее языках. Набор

или система команд для одного ядра могут существенно отличается от другого набора, например, система команд семейства МК MCS51 очень сильно отличается от системы команд AVR, хотя функционально они решают одинаковые задачи. В это связи в данном цикле лабораторных работ по курсу «Микропроцессорные информационно-управляющие системы» применение языка ассемблера не обязательно. При выполнении лабораторных работ студентам необходимо для написания исходного кода применять языки программирования высокого уровня C/C++.

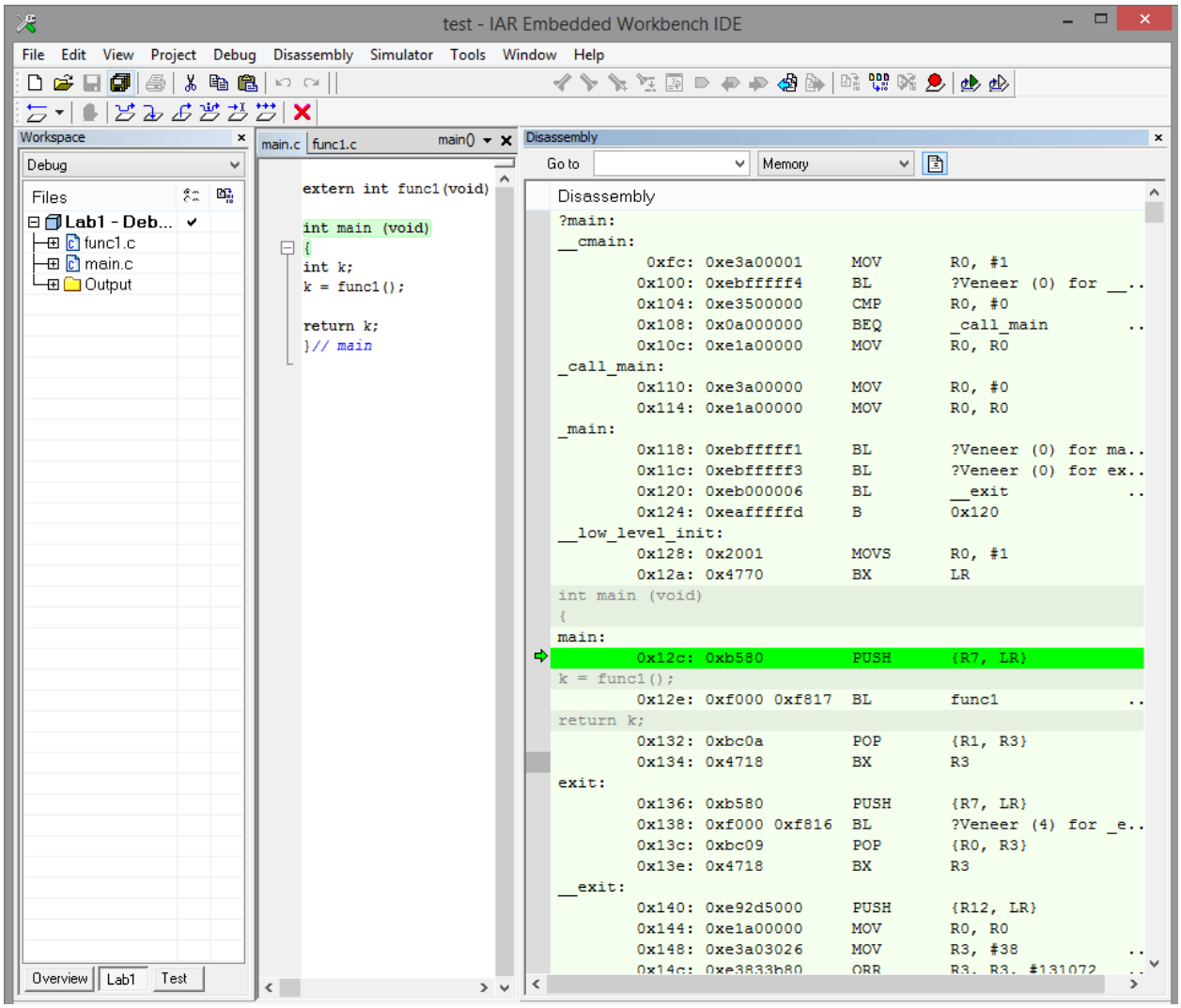

Рисунок 8. Программа, представленная на языке ассемблер.

Отладка первой программы для микроконтроллера

Под отладкой по-нимается процесс, позволяющий получить программу, функциони-рующую с требующимися характеристиками в заданной области изменения входных данных.

Процесс отладки включает:

- действия, направленные на выявление ошибок (тестирование);
- диагностику и локализацию ошибок;

- внесение исправлений в программу с целью устранения ошибок.

Невозможно гарантировать отсутствие ошибок в программе. В лучшем случае можно попытаться показать наличие ошибок. Если программа правильно ведет себя для большого набора тестов, нет оснований утверждать, что в ней нет ошибок. Если считать, что набор тестов способен с большой вероятностью обнаружить возможные ошибки, то можно говорить о некотором уровне уверенности (надежности) в правильности работы программы, устанавливаемом этими тестами.

Таким образом отладка программы направлена на установление точной природы известной ошибки, а затем на исправление этой ошибки.

В среде программирования IAR Embedded Workbench IDE обращение к основным команды отладки программы осуществляются через меню Debug или через кнопки стандартной панели.

К основным командам отладки относятся:

Reset – полный сброс программы микроконтроллера к исходному состоянию.

Step Over – выполнение очередной строки исходного кода и приостановку выполнения на следующей строке.

Step Into – позволяет трассировать код внутри любой функции, которая встречается при пошаговом выполнении программы. В отличие от команды Step Over, которая выполняет функцию и возвращается к следующей строке, команда Step Into помещает точку выполнения в первую строку исходного кода вызванной функции.

Step Out – позволяет трассировать код из любой функции в вызывающую, которая запущена в пошаговом выполнении.

Next statement – выполнение очередного оператора строки исходного кода и приостановку выполнения на следующей строке.

Run to cursor – позволяет осуществить приостановку выполнения на строке на которую указывает курсор.

Go – запускает стандартное выполнение программы из любой строки до следующей точки останова или стандартного завершения программы.

Break – останавливает выполнение программы.

Stop Debugging – останавливает процесс отладки программы.

Компиляция первой программы для микроконтроллера.

После того, как с помощью процесса отладки были выявлены и устранены ошибки в исходном коде необходимо произвести компиляцию проекта.

Компиляция относится к обработке файлов исходного кода (.c, .cc, или .cpp) и создании объектных файлов проекта. На этом этапе не создается исполняемый файл. Вместо этого компилятор просто транслирует

высокоуровневый код в машинный язык. Например, если вы создали (но не скомпоновали) три отдельных файла, у вас будет три объектных файла, созданные в качестве выходных данных на этапе компиляции. Расширение таких файлов будет зависеть от вашего компилятора, например *.obj или *.o. Каждый из этих файлов содержит машинные инструкции, которые эквивалентны исходному коду. Для создания конечного файла программы необходимо использовать компоновщик, путем создания из нескольких объектных файлов единого исполняемого файла.

В среде программирования IAR Embedded Workbench IDE обращение к основным командам сборки и компиляции программы осуществляются через меню Project или через кнопки стандартной панели.

К основным командам сборки и компиляции относятся:

Make – обновляет текущее состояние программы путем компиляции, сборки и компоновки только тех файлов, которые были изменены.

Compile – осуществляет сборки и компиляцию выделенного файла, файлов или групп файлов.

Rebuild All – осуществляет компиляцию и компоновку программы «с нуля».

Clean – удаляет промежуточные файлы проекта.

Stop Build – останавливает текущий процесс компиляции.

После запуска компиляции в среде IAR Embedded Workbench IDE появляется окно сообщений (журнал событий или лог) «Build» (рис. 9). Из данного окна сообщений можно получить информацию о файлах участвующих в процессе компиляции, количестве ошибок в синтаксисе языка программирования и предупреждениях системы.

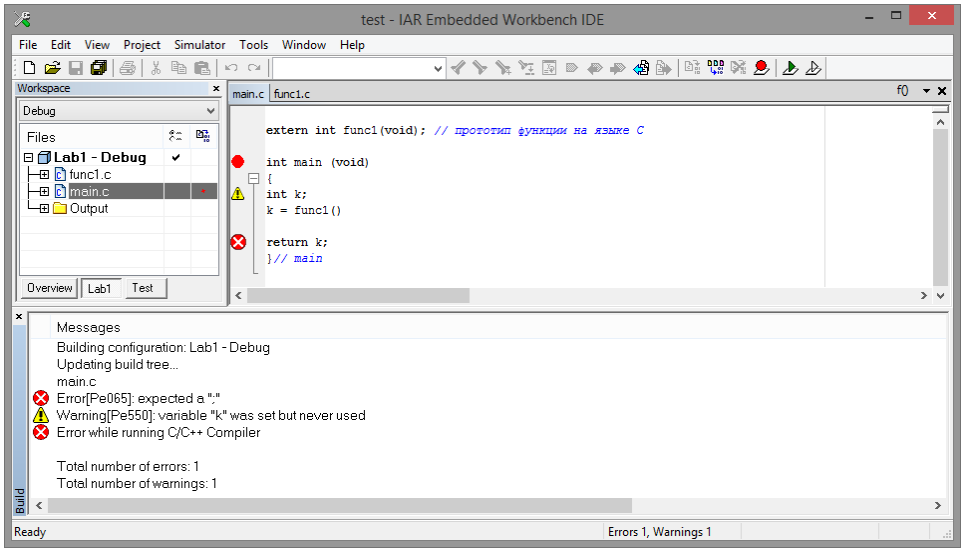

Рисунок 9. Исходный код программы с ошибками.

Порядок выполнения работы

1. Создать новый проект в среде программирования IAR Embedded Workbench IDE.
2. Написать простую функцию на языке высокого уровня С в соответствии с вариантом задания.
3. Скомпилировать и отладить написанную функцию в среде IAR Embedded Workbench IDE.
4. Провести тестирование работоспособности программы путем изменения исходных данных для основной функции.

Содержание отчета

1. Цель работы.
2. Описание последовательности действий для создания проекта.
3. Исходный текст функции main()
4. Исходный текст простой функции на языке С.
5. Результаты тестирования в виде скриншотов значения выходных переменных функции.
6. Выводы по работе.

Варианты заданий

№ варианта	Описание
1	Найти два наибольших элемента массива целых чисел. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
2	Отсортировать элементы массива целых чисел по возрастанию. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
3	Вычисление суммы положительных элементов массива целых чисел. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
4	Сформировать массива из положительных элементов массива целых чисел. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
5	Найти два наименьших элемента массива целых чисел. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
6	Подсчитать количество отрицательных элементов массива чисел с плавающей запятой. Написать процедуру и пример

	обращения к ней. Массив и его фактический размер – параметры.
7	Сформировать массива из отрицательных элементов массива с плавающей запятой. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
8	Вычисление суммы отрицательных элементов массива с плавающей запятой. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
9	Обнуление отрицательных элементов массива целых чисел. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
10	Поиск заданного элемента массива целых чисел. Написать процедуру и пример обращения к ней. Массив, его фактический размер и заданный элемент – параметры.
11	Отсортировать по убыванию элементы массива с плавающей запятой. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
12	Дан массив целых чисел. Найти сумму одинаковых элементов. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
13	Сравнить по модулю сумму элементов массив целых чисел, стоящих на четных местах, с суммой элементов, стоящих на нечетных местах. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.
14	Для одномерного массива целых чисел вычислить произведение первого, третьего и шестого положительных элементов и определить их номера в массиве. Написать процедуру и пример обращения к ней. Массив и его фактический размер – параметры.