

Лабораторная работа №1

Основы работы в программе ST Visual Develop

1. Цель работы

Целью работы является знакомство с программой *ST Visual Develop*. Компиляция и сборка программы на языке Си. Выявление ошибок.

Workspace – рабочее пространство, в котором содержатся ваши проекты. Каждая лабораторная работа это отдельный проект, который будет вложен в ваше рабочее пространство.

2. Создание проекта в программе ST Visual Develop

1. На компьютере необходимо создать папку своего рабочего пространства.

Расположение папки *F:\STVD\GRUPPA\FIO\LabProjects* (Если на вашем ПК нет диска F:\, используйте вместо него диск C:\) Где GRUPPA – папка с номером вашей группы;

FIO – папка с вашей Фамилией (Именем отчеством – по желанию)

Обратите внимание, чтобы в названии всех папок присутствовали только латинские буквы (НЕ РУССКИЕ!!!)

2. Открыть программу *ST Visual Develop*.

3. В меню «*File*» выбрать «*New Workspace*».

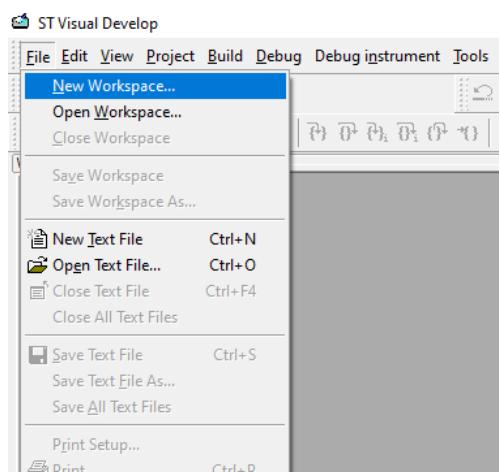


Рис. 1. Меню «File»

4. Далее необходимо выбрать «*Create workspace and project*». Далее нажимаем «*OK*».

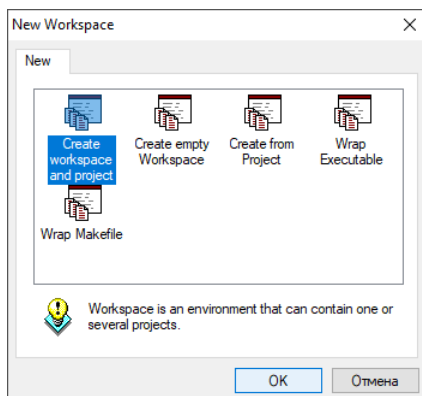


Рис. 2. Внешний вид окна «*New Workspace*»

5. Далее определяем имя файла рабочего пространства в окне «*Workspace filename*». **Имя должно содержать только буквы латинского алфавита и цифры!!!** В окне «*Workspace location*» необходимо задать путь к папке рабочего пространства, созданной в пункте 1. Далее нажимаем «*OK*».

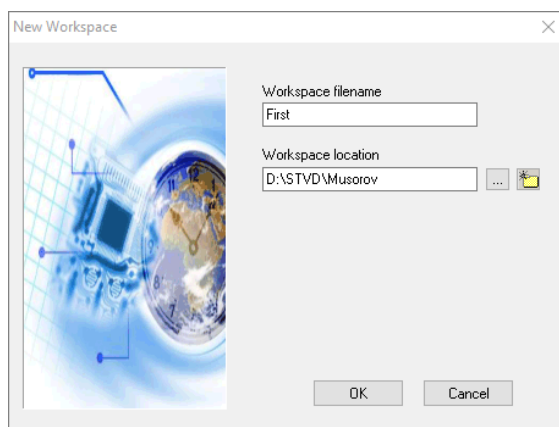


Рис. 3. Определение имени файла и директории рабочего пространства в окне «*New Workspace*»

6. Далее определяем имя файла проекта в окне «*Project filename*». **Имя должно содержать только буквы латинского алфавита и цифры!!!** В окне «*Project location*» необходимо задать путь к папке проекта. Для этого необходимо создать для проекта отдельную папку внутри папки, в которой расположено рабочее пространство. Например:
F:\STVD\GRUPPA\FIO\LabProjects\First.

7. Далее в окне «*Toolchain*» необходимо выбрать, каким компилятором Вы будете пользоваться в проекте. Если Ваша программа будет написана на языке ассемблер, то выбираем «*ST Assembler Linker*», а если на языке Си, то выбираем «*STM8 Cosmic*». Первые лабораторные

работы будут выполняться на языке Си. Также в окне «*Toolchain root*» необходимо указать путь к папке компилятора - *C:\Program Files (x86)\COSMIC\FSE_Compilers* (в случае отсутствия папки компилятора по указанному пути необходимо обратиться к преподавателю). Далее нажимаем «*OK*».

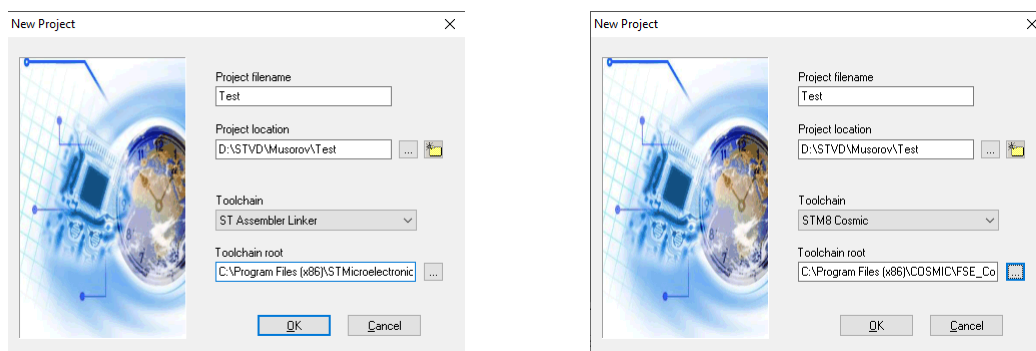


Рис. 4. Определение имени файла, директории и компилятора проекта в окне «*New Project*»

8. Далее необходимо выбрать микроконтроллер. В появившемся окне выбираем *STM8S207RB* и нажимаем кнопку «*Select*». Далее нажимаем «*OK*».

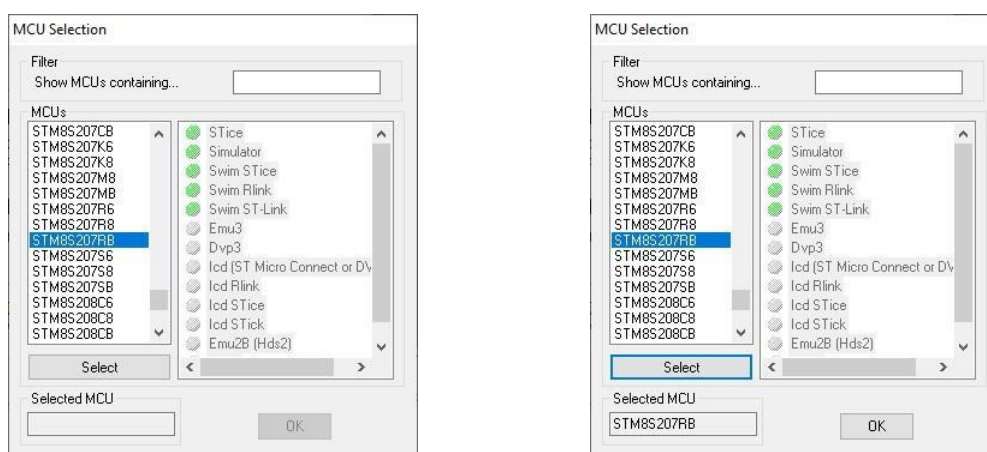


Рис. 5. Меню выбора микроконтроллера

9. После этого в программе появится Ваш проект. Далее необходимо сохранить рабочее пространство следующим образом: *File – Save Workspace*.

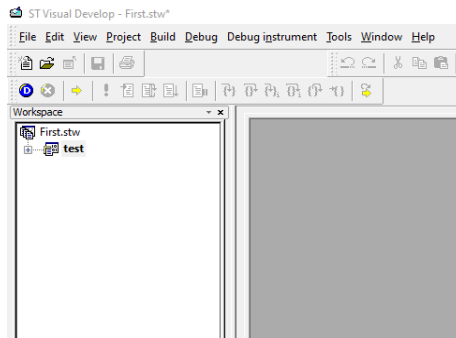


Рис. 6. Внешний вид проекта в ST Visual Develop

10. В проекте автоматически добавлены некоторые файлы. Один из них – *main.c*. В данном файле будем писать программу для МК.

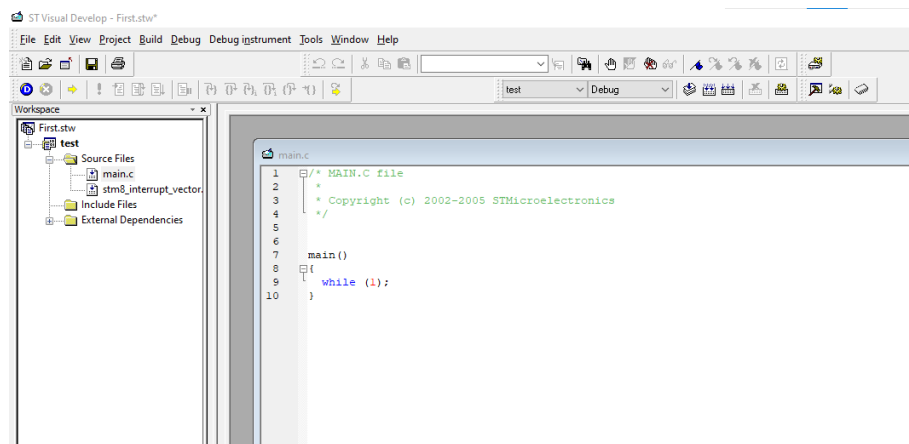


Рис. 7. Файл основной программы *main.c* в проекте

11. Данный пункт нужно прочесть, но выполнять сейчас его не нужно. Добавление нового проекта необходимо в случае, когда вы приступаете выполнять следующую лабораторную работу. Для добавления в существующее рабочее пространство нового проекта необходимо нажать ПКМ на имени рабочего пространства и выбрать пункт *Add new project to workspace*.

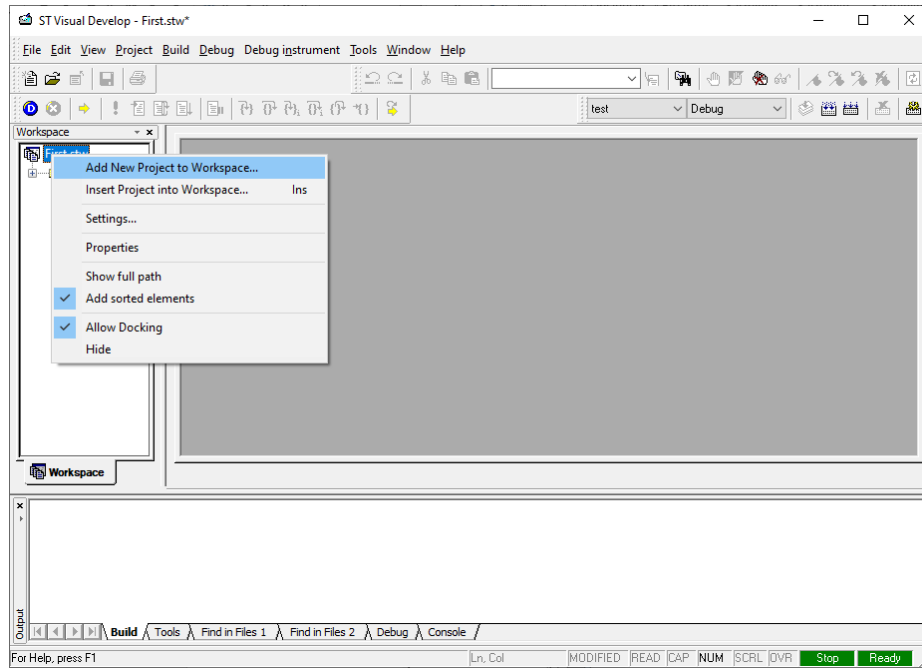


Рис. 8. Добавление нового проекта в существующее рабочее пространство

3. Процесс загрузки и выполнения программного кода.

1. Далее необходимо в программе выполнить подключение заголовочного файла, в котором объявлены переменные с именами регистров. Каждой такой переменной назначен специально заданный адрес, который равен адресу соответствующего регистра. Таким образом, изменение значения переменных, определенных в этом файле, ведет к изменению содержания соответствующих регистров микроконтроллера. Подключение файла осуществляется следующим образом:

#include <iostm8s207.h>

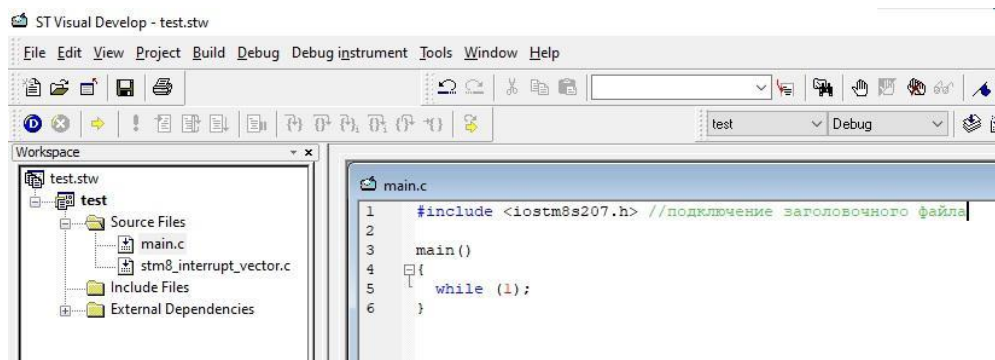


Рис. 9. Подключение заголовочного файла в основной программе

2. Для демонстрации процесса загрузки программы в микроконтроллер необходимо подключить отладочную плату к ПК. Подключение осуществляется посредством кабеля *USB-USB Mini*.

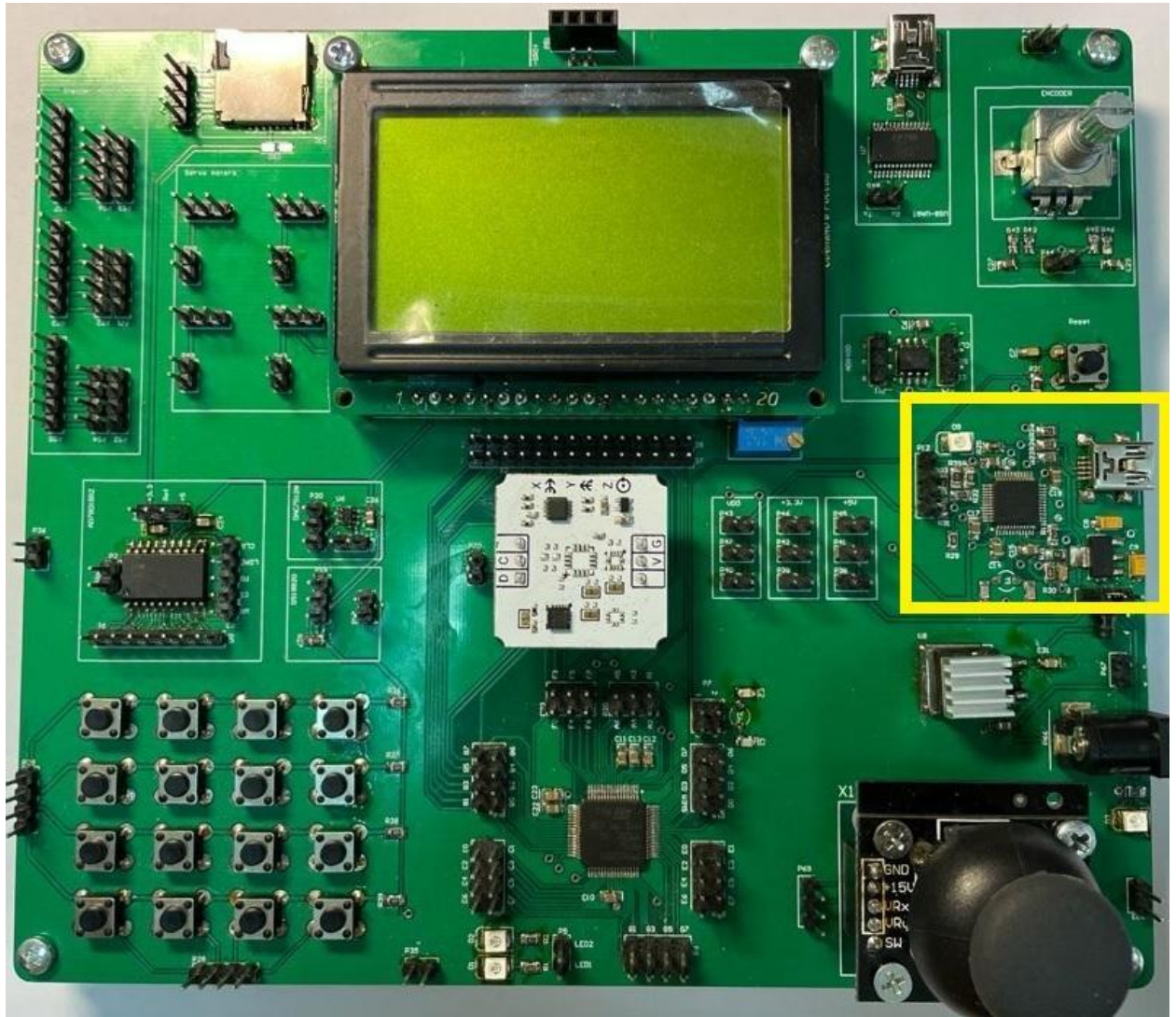


Рис. 10. Внешний вид отладочной платы на базе микроконтроллера STM8S207RB

3. Для демонстрации подготовим программу зажигания светодиода на плате. Для этого в функции *main* () необходимо написать код следующей программы:

```
PA_DDR = 0x02;  
PA_CR1 = 0x02;  
PA_CR2 = 0x02;  
PA_ODR = 0x02;
```

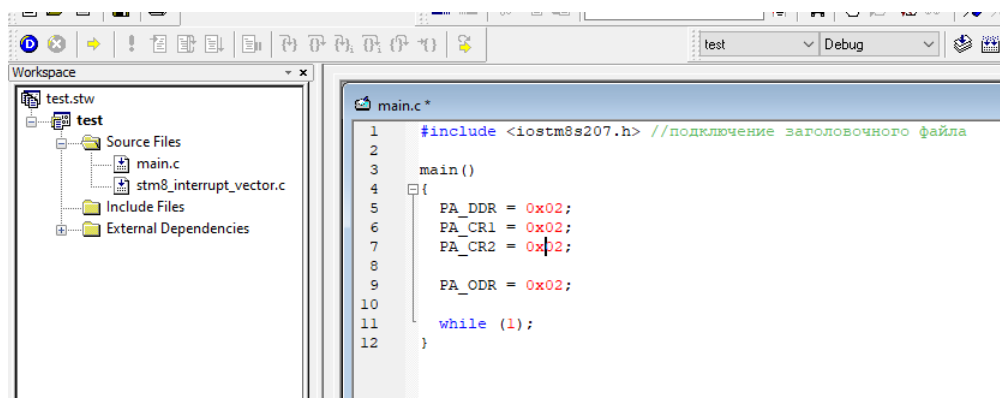


Рис. 11. Код программы для светодиода

4. Для загрузки программы в микроконтроллер необходимы выполнить ее компиляцию. Компиляция программы может быть запущена во вкладке «Build» или по кнопке на панели программы.

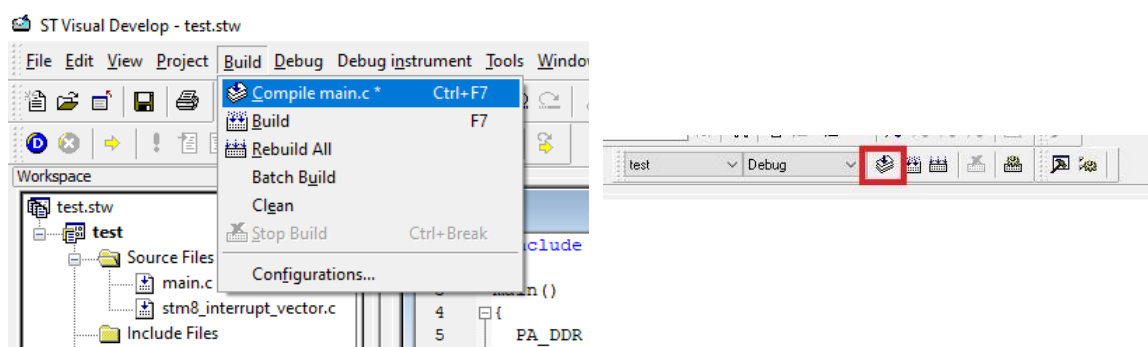


Рис. 12. Компиляция программы

5. После компиляции в нижней части экрана появится результат. Необходимо убедиться, что в программе отсутствуют ошибки.

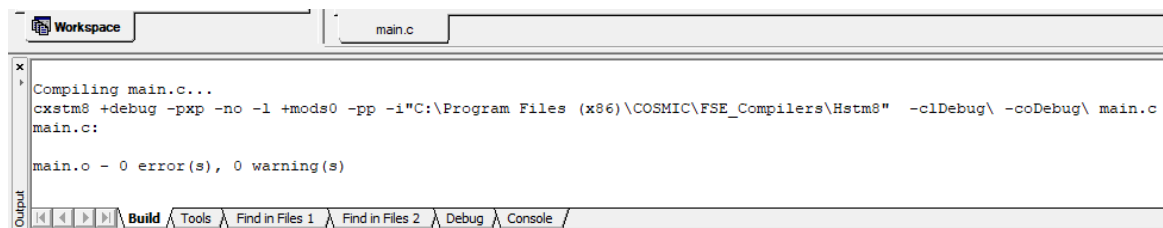


Рис. 13. Окно ошибок компилятора

6. Далее необходимо выполнить сборку программы. Сборка программы также может быть запущена во вкладке «Build» или по кнопке на панели программы.

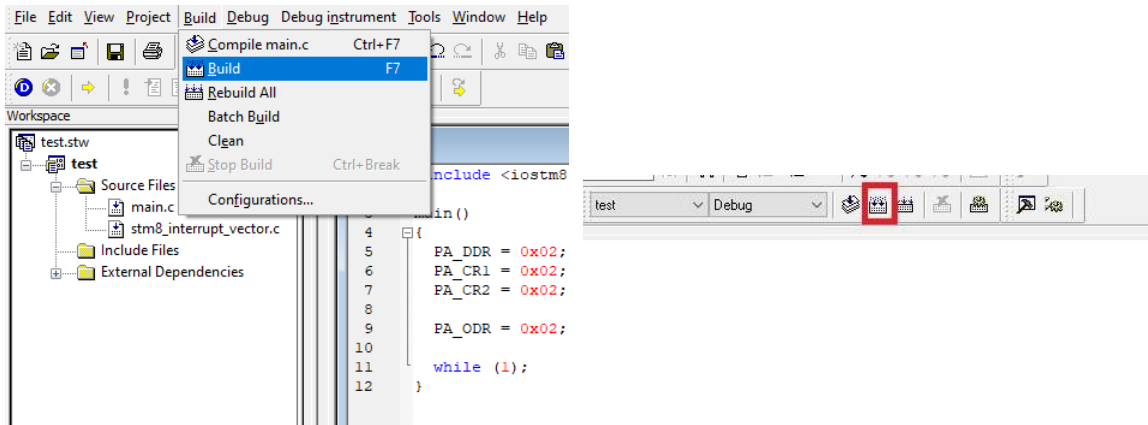


Рис. 14. Запуск программы

7. После сборки в нижней части экрана появится результат. Необходимо также убедиться в отсутствии ошибок. **Компиляцию и сборку, перед загрузкой программы в микроконтроллер, необходимо осуществлять после любых изменений в программе!!! В ином случае в микроконтроллер может быть загружена предыдущая версия программы!!!**

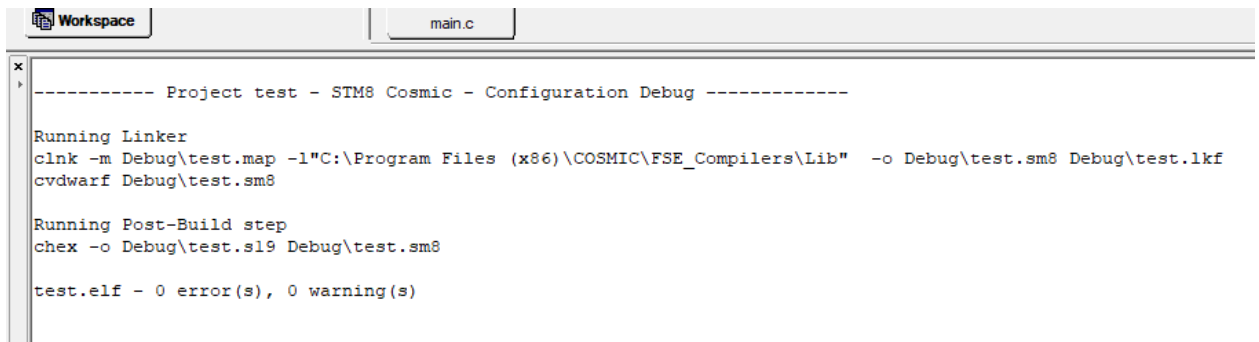
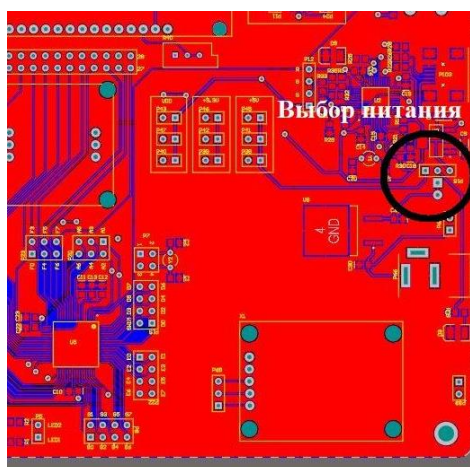


Рис. 15. Результат сборки программы

8. Перед загрузкой программы необходимо убедиться, что на плате присутствует перемычку (джампер) выбора источника питания.



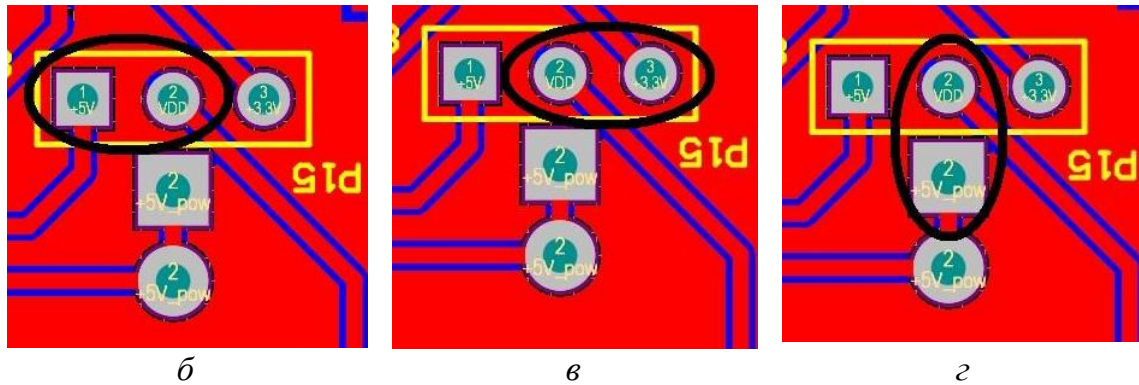


Рис. 16. Схема подключения питания: а – схема расположения коннектора на плате; б – выбор питания 5 В от USB; в – выбор питания 3.3 В от USB; г – выбор питания 5 В от внешнего источника

9. Далее можно осуществлять загрузку программы в микроконтроллер. Представленная выше программа осуществляет настройку вывода 1 порта А и вывод на нее сигнала высокого уровня. Для демонстрации необходимо подключить светодиод к выводу 1 порта А с помощью соединительного провода типа «мама-мама». На отладочной плате имеется два светодиода. Возможно подключение любого из них.

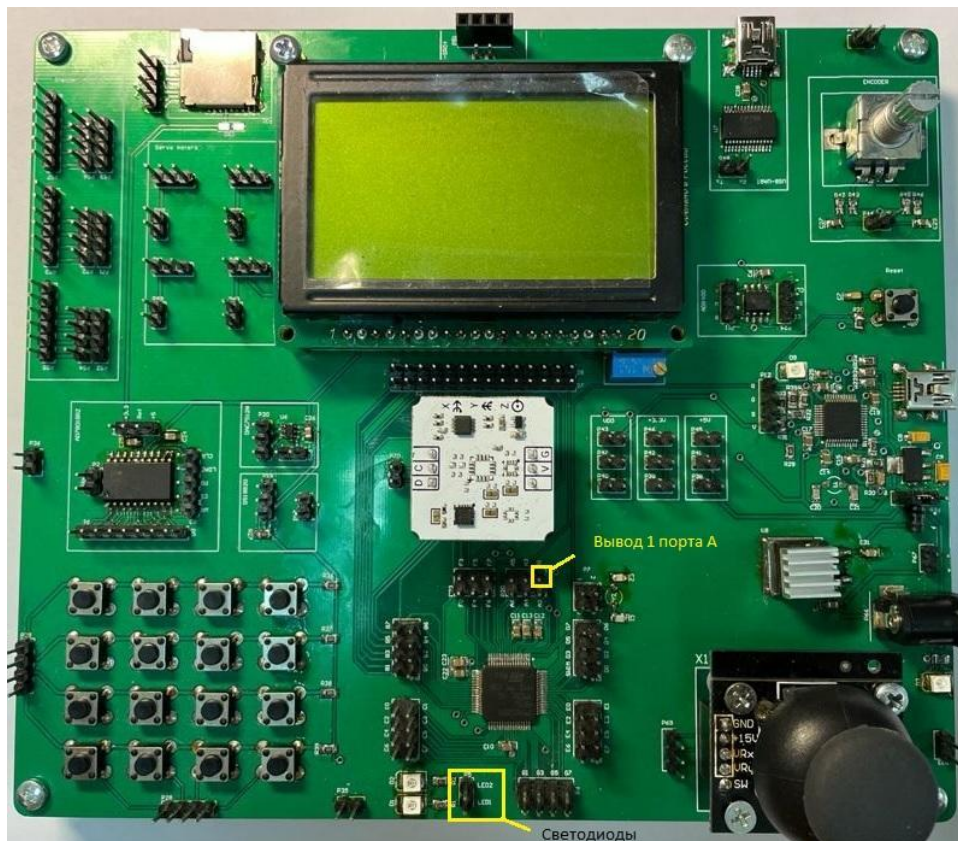


Рис. 17. Светодиоды на отладочной плате

10. Загрузка программы в микроконтроллер осуществляется во вкладке «*Debug*» или по кнопке на панели программы.

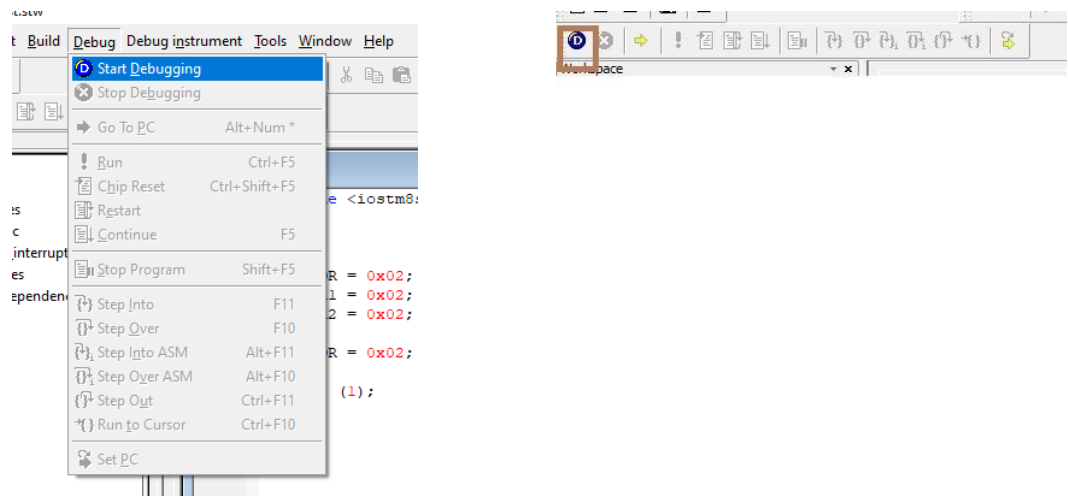


Рис. 18. Кнопка для загрузки программы

11. После загрузки программы станут активными кнопки панели «*Debug*».

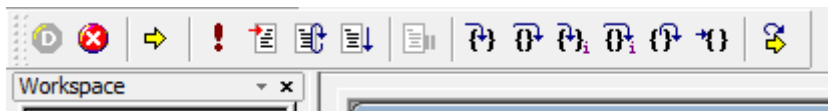


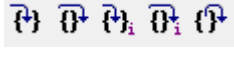



Рис. 19. Панель «*Debug*»

Таблица 1

Кнопки панели «*Debug*»

	<i>Stop debugging</i>	Выход из режима отладки
	<i>Run</i>	Запуск программы
	<i>Chip reset</i>	Сбрасывает микроконтроллер. Регистр <i>PC</i> возвращается к началу процедуры запуска (<i>startup</i>). Все что выполняется до перехода к функции <i>main</i> .
	<i>Restart Application</i>	<i>PC</i> на первую команду функции <i>main</i> для приложений на языке C. Для приложений на ассемблере регистр <i>PC</i> устанавливается на первую команду функции с пометкой <i>main</i> , если она существует. Команда Chip Reset:

	<i>Continue</i>	Продолжение выполнения программы после паузы
	<i>Pause</i>	Остановка процесса выполнения программы
	<i>Step into</i> <i>Step over</i>	Кнопки управления пошаговым выполнением программы
	<i>Run to cursor</i>	Выполнение программы до команды, на которую установлен курсор

12. Далее выполняем запуск программы (*Run*) и наблюдаем результат (должен загореться соответствующий светодиод). Обратите внимание, что запуск выполнения программы с помощью кнопки *Run*, допустимо использовать, только в случае, когда вы только загрузили программу. В случае, когда вы уже запускали программу и она остановила свое выполнение по какой-то причине (была нажата пауза, или встретила точка останова), то для продолжения выполнения программы нужно использовать кнопку *Continue*. Если в данном случае использовать кнопку *Run*, то осуществится сброс контроллера, и программа запустится с самого начала

13. В случае изменения программного кода необходимо выйти из режима отладки (*Stop debugging*), внести соответствующие изменения и вновь выполнить компилирование, сборку и загрузку.

4. Ошибки компиляции

1. Далее выполним процесс исправления ошибок компилирования. Для этого выйдем из режима отладки и изменим исходный код программы следующим образом:

- 1) Удалим «;» в строке «*PA_DDR = 0x02;*».
- 2) Заменим строчку «*PA_CR2 = 0x02;*» на строчку «*PA_CR = 0x02;*».

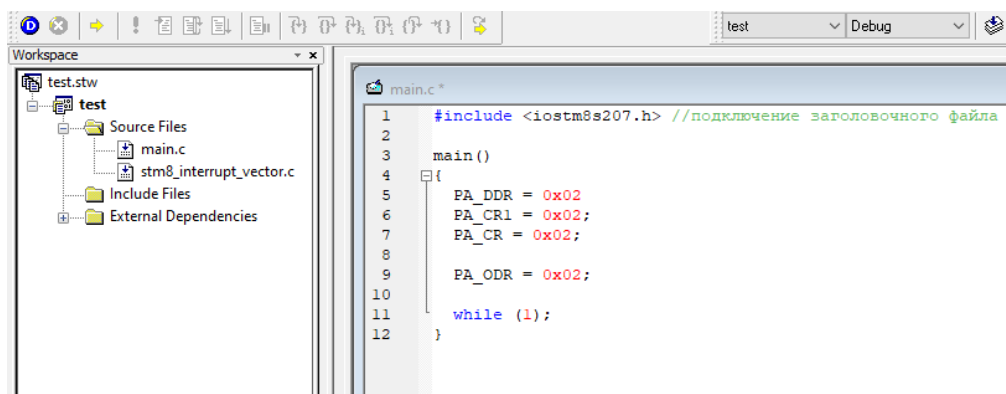


Рис. 20. Измененный код программы для светодиода

2. Выполним компиляцию программы. После компиляции нам будет показано, что в программе есть ошибки. В окне «*Build*» ошибки будут идентифицированы (см. рисунок ниже). В частности, компилятор показывает, что в строке 5 отсутствует «;», а в строке 7 запись *PA_CR* не идентифицирована (отсутствует ее объявление в библиотеке).

```
----- Project test - STM8 Cosmic - Configuration Debug -----
Compiling main.c...
cxstm8 +debug -pxp -no -l +mods0 -pp -i"C:\Program Files (x86)\COSMIC\FSE_Compilers\Hstm8" -clDebug\ -coDebug\ main.c
#error cpstm8 main.c:5(10+4) missing ;
#error cpstm8 main.c:7(1+5) PA_CR undefined
main.c:
The command: "cxstm8 +debug -pxp -no -l +mods0 -pp -i"C:\Program Files (x86)\COSMIC\FSE_Compilers\Hstm8" -clDebug\ -c
exit code=1.
main.o - 4 error(s), 0 warning(s)
```

Рис.21. Ошибки компиляции программы

5. Пошаговое выполнение программы. Точки останова.

1. Для демонстрации процесса пошагового выполнения программы и использования точек останова вернемся к исходному коду программы.

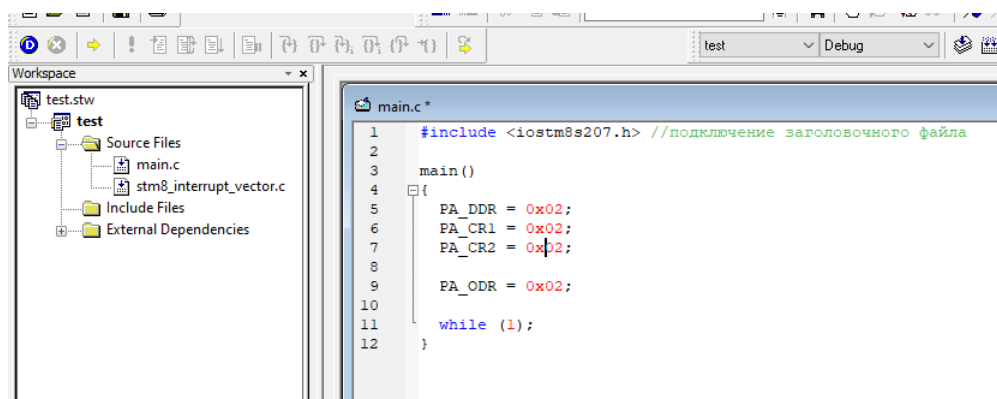


Рис. 22. Исходный код программы

2. Выполним компиляцию, сборку и загрузку программы.

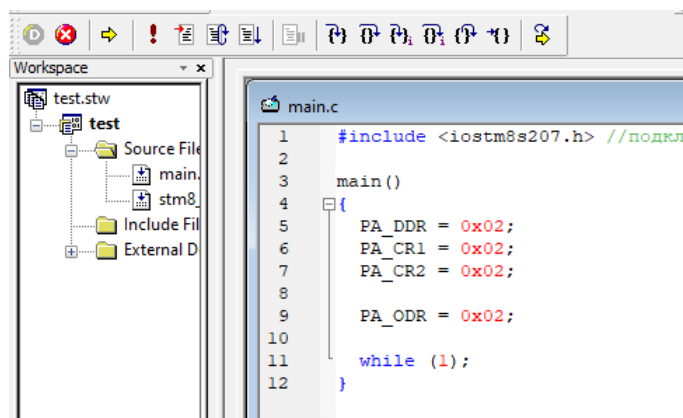


Рис. 23. Окно программы после загрузки

3. Для запуска программы в пошаговом режиме необходимо сначала нажать «*Restart application*» либо установить курсор на 5 строку и выполнить команду «*Run to cursor*». При этом в поле программы появится курсор (желтая полоска со стрелкой), указывающий на команду, которая будет выполняться следующей.

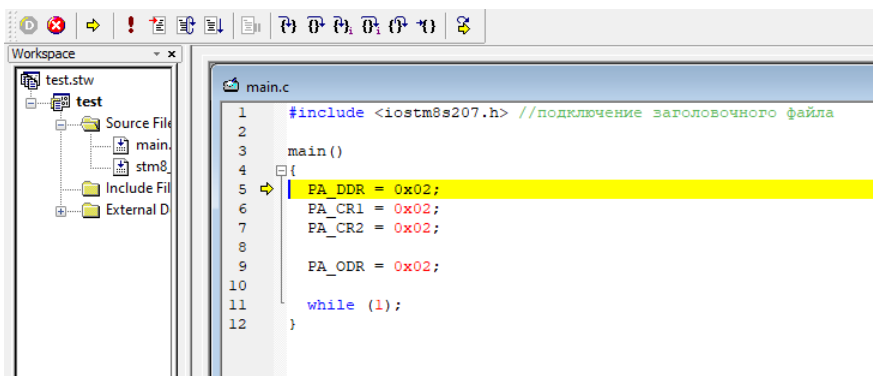



Рис. 24. Пошаговый режим работы программы

4. В пошаговом режиме работы будет выполняться только одна команда (для ассемблера) или одна операция (для C). Шаг вперед осуществляется нажатием кнопки «*Step into*» - . При этом курсор перейдет на следующую операцию, а программа будет в состоянии паузы.

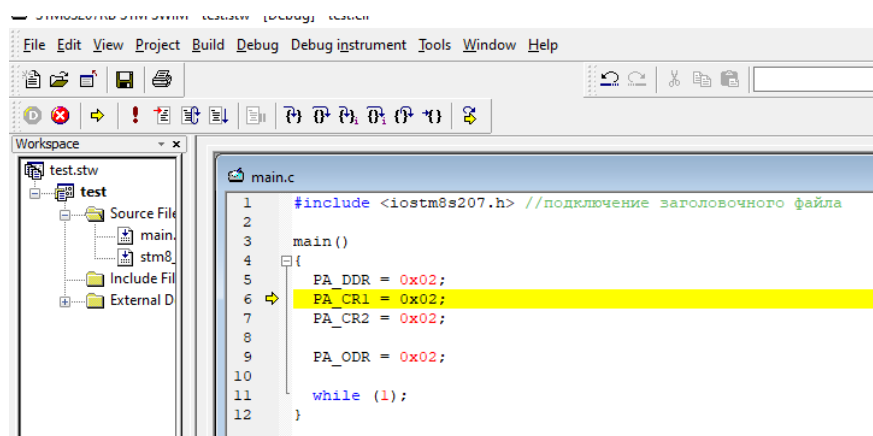


Рис. 25. Вид окна в пошаговом режиме после нажатия кнопки «*Step into*»

5. Выполняя программу шаг за шагом можно будет заметить, что светодиод на плате загорится только после выполнения операции «*PA_ODR = 0x02;*».

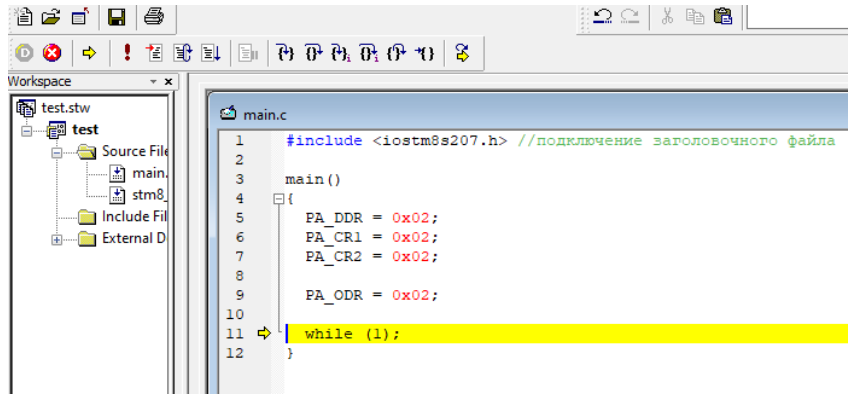


Рис. 26. Переход к бесконечному циклу работы программы

6. Для возврата к началу программы необходимо нажать «Restart application».

7. Далее продемонстрируем отладку программы с использованием точек останова. В случае наличия точки останова программа будет выполняться только до определенной операции. Для того чтобы поставить точку останова необходимо выполнить один клик напротив соответствующей операции (см рисунок ниже).

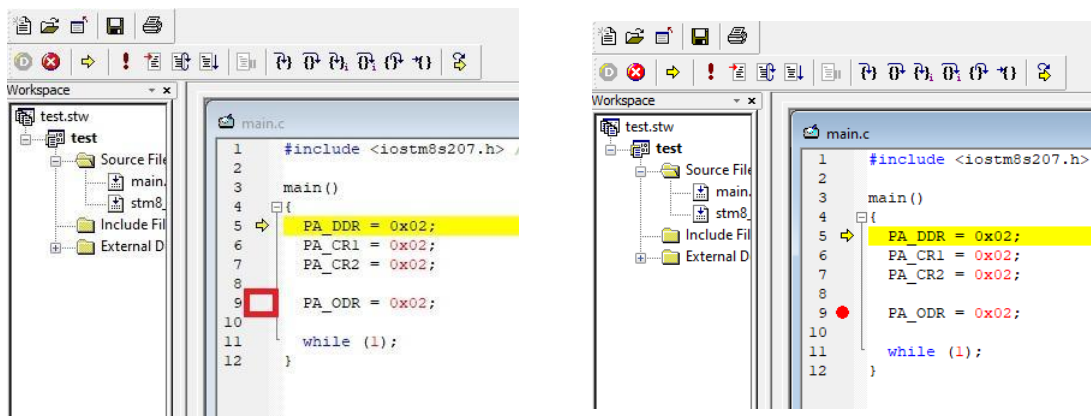


Рис. 27. Постановка точки останова в программе

8. Далее можно запустить программу («Run»). При этом будут выполнены все операции до точки останова и программа будет остановлена на операции «PA_ODR = 0x02;». При этом сама эта операция еще не будет выполнена!!!

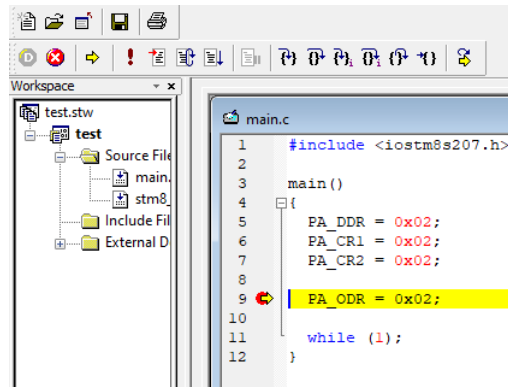


Рис. 28. Остановка программы на точке

9. Далее можно продолжить выполнение программы. Продолжение выполнения программы осуществляется нажатием кнопки «Continue». В нашем примере программа перейдет к выполнению пустого бесконечного цикла «while(1);» и больше к операции «PA_ODR = 0x02;» не вернется.

10. Отметим, что в программе может быть несколько точек останова и выполнение будет останавливаться на любой из них. Также следует отметить, что количество возможных точек останова ограничено и определяется конкретным программным обеспечением.

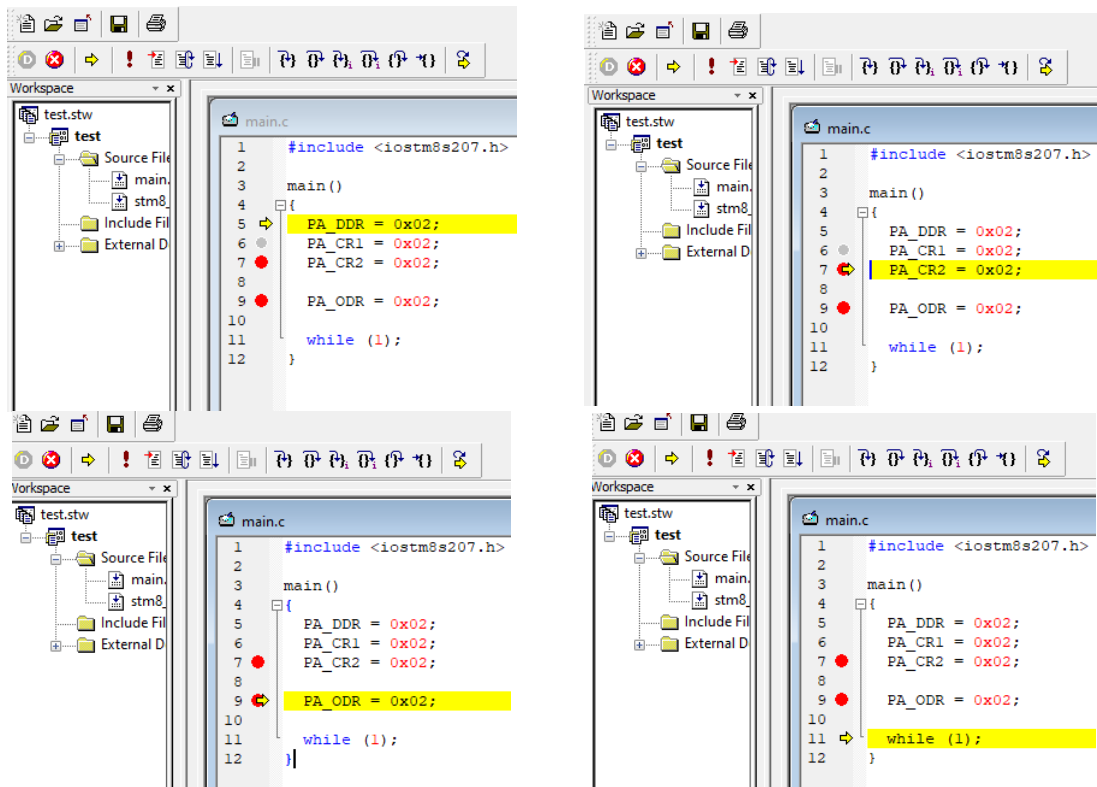


Рис. 29. Работа программы с точками останова

11. Исключение точки останова из программы осуществляется также кликом напротив соответствующей операции.

12. Перед отключением отладочной платы необходимо выйти из режима отладки (*Stop debugging*).

6. Просмотр «регистров периферии» микроконтроллера.

1. Вернемся к исходному коду программы.

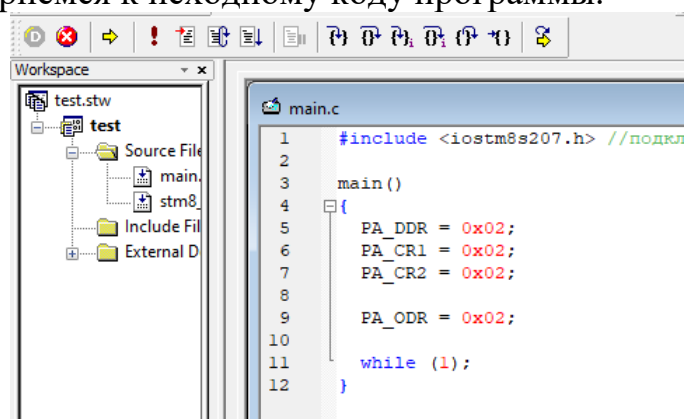


Рис. 30. Исходный код программы

2. Напомним, что в данной программе происходит настройка вывода 1 порта на выход ($PA_DDR = 0x02$; $PA_CR1 = 0x02$; $PA_CR2 = 0x02$;) и установка на данном выводе сигнала высокого уровня ($PA_ODR = 0x02$;).

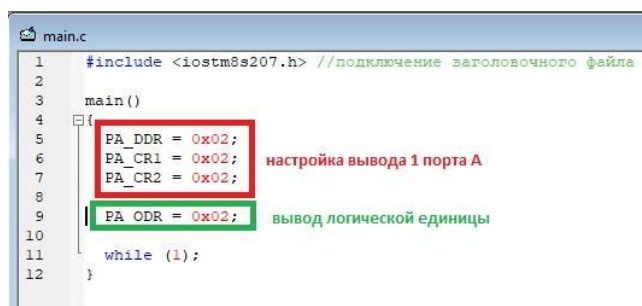


Рис. 31. Настройка и вывод логической единицы на порт

3. В данной программе PA_DDR , PA_CR1 , PA_CR2 и PA_ODR являются периферийными регистрами портов ввода/вывода. В микроконтроллере они относятся к классу регистров специального назначения. В микроконтроллере таких регистров специального назначения большое количество (регистры процессора, регистры периферийных устройств микроконтроллера). Значение большинства регистров специального назначения определяется разработчиком программы (как в нашем примере). Однако значение ряда регистров могут изменяться в процессе работы аппаратно, т.е. без «спроса» разработчика. Поэтому в процессе отладки программы часто появляется

необходимость посмотреть содержимое регистров специального назначения. В программе в программе *ST Visual Deveop* просмотр содержимого регистров специального назначения возможен после остановки выполнения кода (Пауза, Точка останова, Пошаговый режим). Приведем пример просмотра содержимого регистров в пошаговом режиме работы.

4. Осуществим компиляцию, сборку и загрузку программы. Подключим отладочную плату к ПК.

5. Для перехода в пошаговый режим нажимаем кнопку «*Run to cursor*».

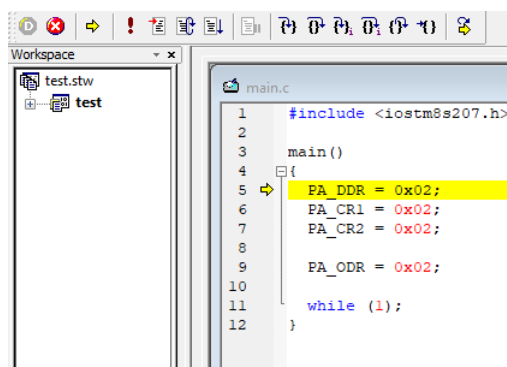


Рис. 32. Переход в пошаговый режим отладки

6. Для просмотра содержимого регистров микроконтроллера необходимо во вкладке «*View*» выбрать «*Peripheral registers*».

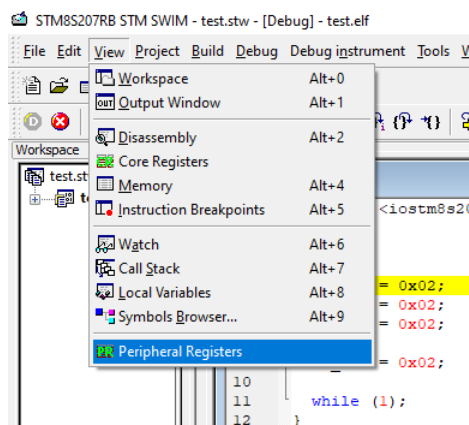


Рис. 33. Просмотр регистров микроконтроллера с помощью «*Peripheral registers*»

7. При этом будет открыто следующее окно:

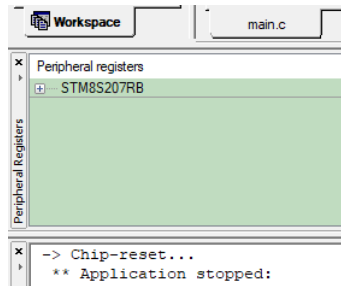


Рис. 34. внешний вид окна «Peripheral registers»

8. В данном окне можно посмотреть содержимое всех регистров периферии микроконтроллера. В нашем примере нас интересуют регистры порта *A*.

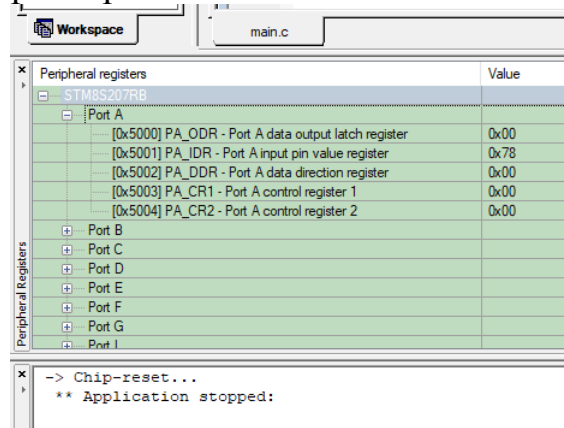
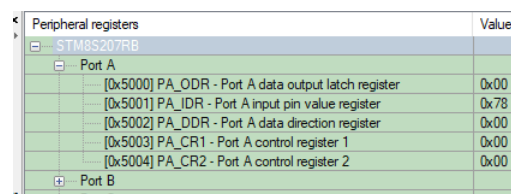
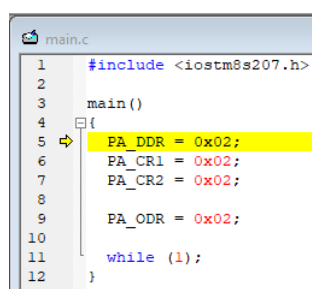


Рис. 33. Содержимое регистров порта *A*

9. Вы можете увидеть, что пока программа не запущена в регистрах *PA_DDR*, *PA_CR1*, *PA_CR2* и *PA_ODR* лежат значения 0x00. Это значения данных регистров после сброса микроконтроллера. Значения регистров после сброса можно найти в документации на микроконтроллер. **Они не всегда равны «0»!!!** Регистр *PA_IDR* – это регистр входных данных и его значение определяется тем какой уровень сигнала присутствует на соответствующем выводе в данный момент.

10. Далее выполняя пошагово программу до бесконечного цикла (4 шага) мы можем наблюдать изменения содержимого регистров порта *A*.



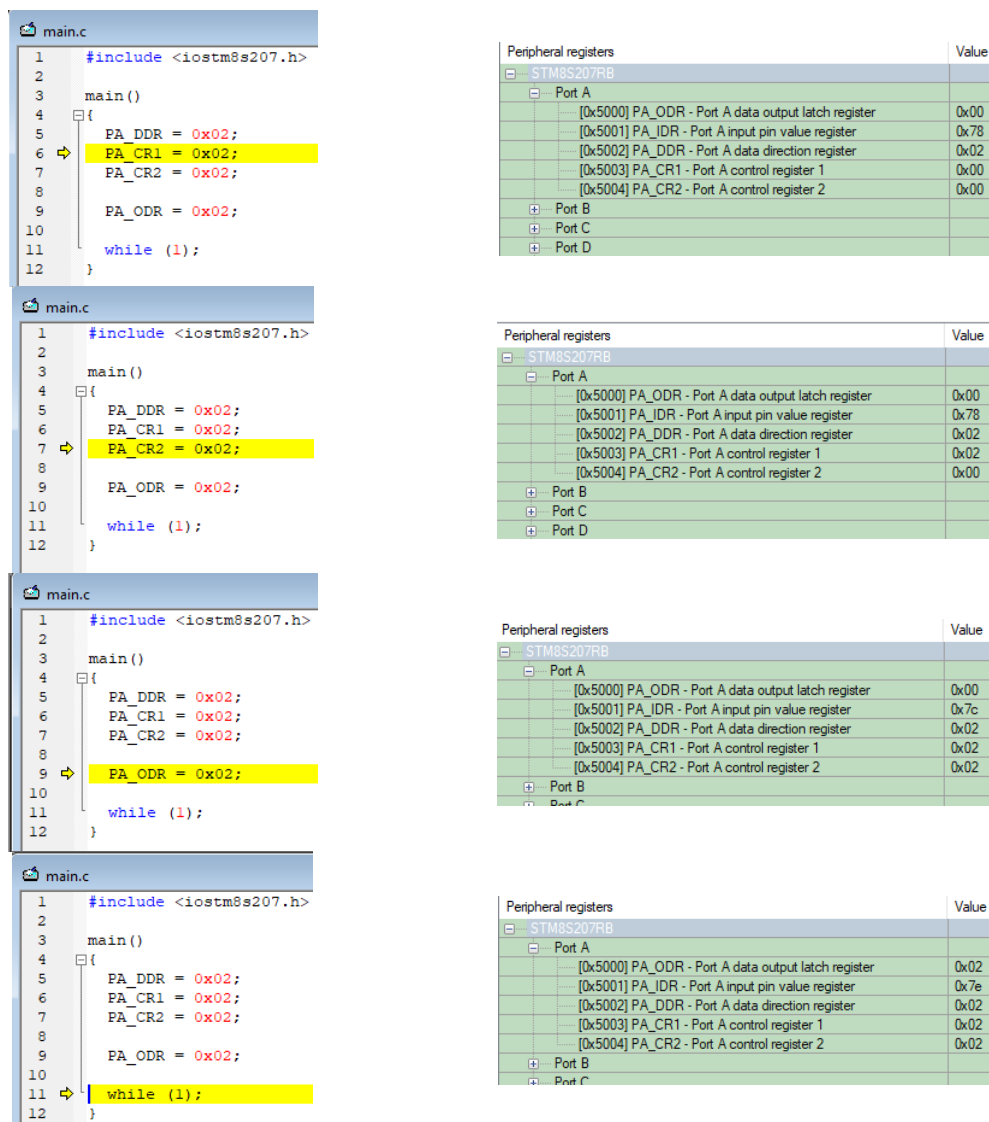


Рис. 34. Изменение содержимого регистров в процессе пошаговой отладки

11. Также в режиме отладки есть возможность изменять содержимое регистров и переменных вручную.

7. Просмотр значения переменных

1. Для просмотра значения переменных подготовим следующую программу.

```

main.c
1  #include <iostm8s207.h> //подключение заголовочного файла
2
3  int a,b;
4  char c;
5
6  main()
7  {
8      PA_DDR = 0x02;
9      PA_CR1 = 0x02;
10     PA_CR2 = 0x02;
11
12     PA_ODR = 0x02;
13
14     a=100;
15     b=a+1000;
16     c=0x0F;
17
18     while (1);
19 }

```

Рис. 35. Программа для просмотра значений переменных

2. Выполним компиляцию, сборку и загрузку в микроконтроллер программы.

3. Как и в случае с регистрами микроконтроллера, просмотр значения переменных возможен в режиме остановки программы (пауза, точка останова, пошаговый режим).

4. Для просмотра значения переменных необходимо во вкладке «View» выбрать «Watch».

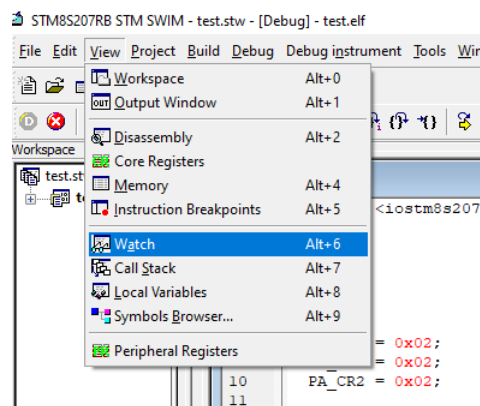


Рис. 36. Переход к окну просмотра значений переменных

5. При этом будет открыто следующее окно:

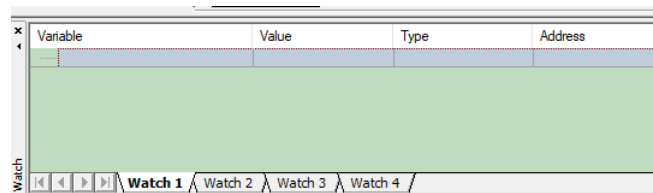
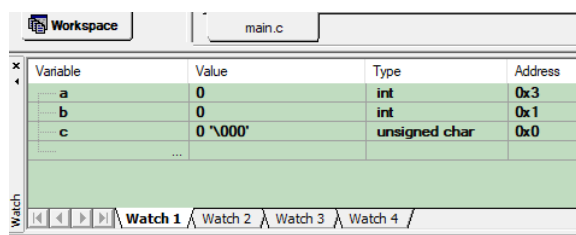


Рис. 37. внешний вид окна «Watch 1»

6. Далее в открывшемся окне можно ввести по очереди все переменные, значения которых мы хотим посмотреть. По умолчанию значения всех переменных равны 0x00. Отметим, что переменная типа *char* имеет размер 1 байт, а переменная типа *int* имеет размер 2 байта

(размер переменной данного типа определяется конкретным компилятором и может быть, как 2, так и 4 байта), что видно в столбце «Address».

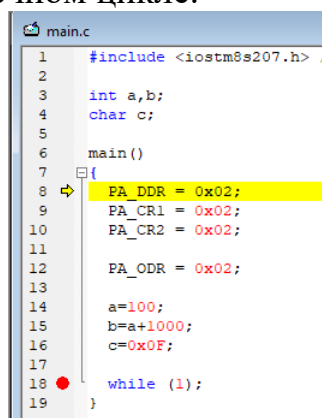


Variable	Value	Type	Address
a	0	int	0x3
b	0	int	0x1
c	0 '\000'	unsigned char	0x0

Рис. 38. Просмотр значений использованных в программе переменных

Пример (рис. 38): Переменная *c* типа *Char* помещается в ячейку памяти с адресом 0x0 и занимает в памяти 1 байт. Переменная *b* типа *int* имеет размер 2 байта и занимает в памяти ячейки с адресами 0x1 и 0x2. Переменная *a* типа *int* занимает в памяти ячейки с адресами 0x3 и 0x4.

7. Для того чтобы посмотреть значения переменных поставим точку останова на бесконечном цикле.

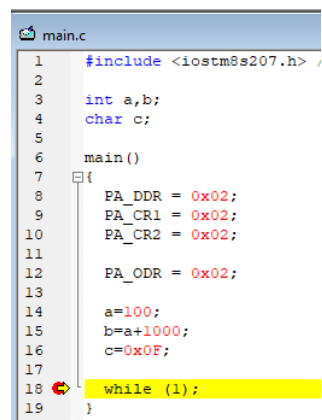


```

1  #include <iostm8s207.h>
2
3  int a,b;
4  char c;
5
6  main()
7  {
8      PA_DDR = 0x02;
9      PA_CR1 = 0x02;
10     PA_CR2 = 0x02;
11
12     PA_ODR = 0x02;
13
14     a=100;
15     b=a+1000;
16     c=0x0F;
17
18     while (1);
19 }
  
```

Рис.39. Постановка точки останова в бесконечном цикле

8. После запуска программа будет остановлена на бесконечном цикле после присвоения всем переменным определенных значений. И мы можем наблюдать в окне «Watch» значения всех трех переменных.



```

1  #include <iostm8s207.h>
2
3  int a,b;
4  char c;
5
6  main()
7  {
8      PA_DDR = 0x02;
9      PA_CR1 = 0x02;
10     PA_CR2 = 0x02;
11
12     PA_ODR = 0x02;
13
14     a=100;
15     b=a+1000;
16     c=0x0F;
17
18     while (1);
19 }
  
```

Рис. 40. Остановка программы в бесконечном цикле

Workspace		main.c	
Variable	Value	Type	Address
a	100	int	0x3
b	1100	int	0x1
c	15 '\017'	unsigned char	0x0
...			
Watch			
<div> <div> <div>◀</div> <div>▶</div> <div>◀▶</div> <div>◀▶</div> </div> <div>Watch 1</div> <div>Watch 2</div> <div>Watch 3</div> <div>Watch 4</div> </div>			

Рис. 41. Значения использованных в программе переменных