UM0834
User manual

Developing and debugging your STM8S-DISCOVERY application code

Introduction

This document complements the information in the STM8S datasheets by describing the software environment and development recommendations required to build an application around the STM8S-DISCOVERY kit. It also explains how to use the STM8S firmware and STM8S touch sensing libraries provided by STMicroelectronics, in order to develop cost-effective applications.

In addition, ST provides a development package which can be used to build an application running on the STM8S-DISCOVERY.

Reference documents

- ST Visual Develop (STVD) user manual (UM0036)
- Adjustable LED blinking speed using STM8S-DISCOVERY touch sensing key (UM0833)
- STM8S-DISCOVERY user manual (UM0817)

Contents

List of tables

List of figures

1 Overview of STM8 software development toolchain

To develop, compile and run an application software on an STM8S microcontroller, the following software toolchain components are required:

- Integrated development environment (IDE) composed of the ST Visual Develop (STVD) and the ST Visual Programmer software interface (STVP)
- Compilers
- Firmware libraries: they are optional, and allow to easily create a new application

STMicroelectronics provides a free software package including STVD and STVP.

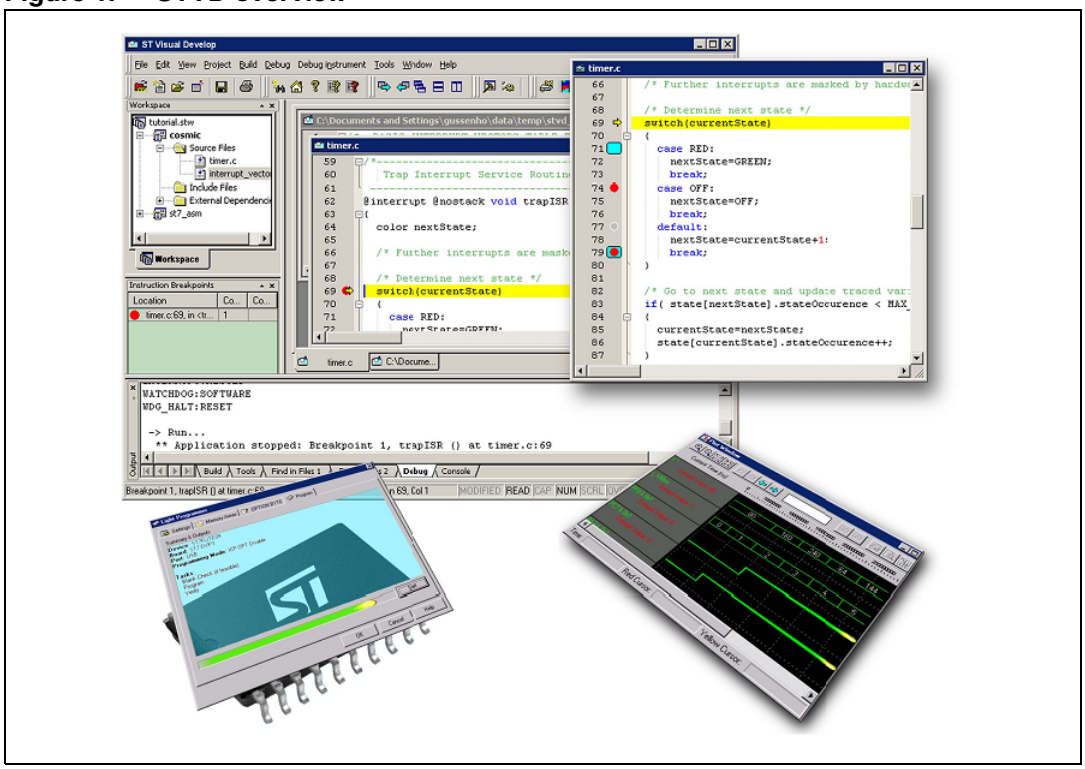
1.1 ST Visual Develop (STVD)

STVD is a full-featured development environment. It is a seamless integration of the Cosmic and Raisonance C compilers for STM8 microcontroller family. These compilers are free when developing code up to 16 Kbytes.

STVD main features are:

- Seamless integration of C and ASM compilers
- Full-featured debugger
- Project management
- Syntax highlighting editor
- Integrated programming interface

Figure 1. STVD overview



1.2 ST Visual Programmer (STVP)

STVP is a easy-to-use graphical interface allowing to read, write and verify the code and data programmed in your STM8 microcontroller Flash program memory, data EEPROM and option bytes. STVP also features a project mode for saving programming configurations and automating programming sequences.

1.3 C and assembly compilers

The C and assembly compiler is seamless integrated into the STVD development environment. They allow to directly configure and control the building of your application from an easy-to-use graphical interface.

The supported compilers are the following:

- Cosmic C compiler for STM8 (free version up to 16 Kbytes of code)

  For more information, refer to http://www.cosmic-software.com.
- Raisonance C compiler for STM8 (free up to 16 Kbytes of code)

  For more information, refer to http://www.raisonance.com.
- STM8 assembler linker

  This is a free assembly toolchain included in the STVD toolset. It allows to assemble and link your application source code.

1.4 Firmware libraries

The STM8S standard firmware library is a complete package consisting of drivers for all the standard peripherals of Performance line STM8S20x and Access line STM8S10x microcontrollers. It is written in strict ANSI-C code and is fully MISRA C 2004 compliant.

The STM8S touch sensing library endows any STM8 microcontroller with touch sensing capabilities. This firmware offers a complete and robust solution to manage capacitive sensing keys, wheels, and sliders.

Refer to *Section 3.1: STM8S standard firmware library* and *Section 3.2: STM8S touch sensing library* for details about library installation and configuration.

2 Installing the development toolchain

2.1 Downloading and installing STVD

STVD software is available at
http://www.st.com/stonline/products/support/micro/files/sttoolset.exe.

To install STVD, download the installation software and follow each step of the installation wizard.

When the installation is complete, the executable is available from ***START>Programs>ST Toolset>Developement Tools>ST Visual Develop*** or under **C:\Program Files\STMicroelectronics\st_toolset\stvd**.

2.2 Downloading and installing the compilers

Cosmic and Raisonance compilers are compatible with STVD. They are available together with their documentation at the following urls:

- Cosmic: http://www.cosmicsoftware.com/download_stm8_16k.php
- Raisonance: http://www.mcu-raisonance.com

Note: A free license is required to use the compilers.

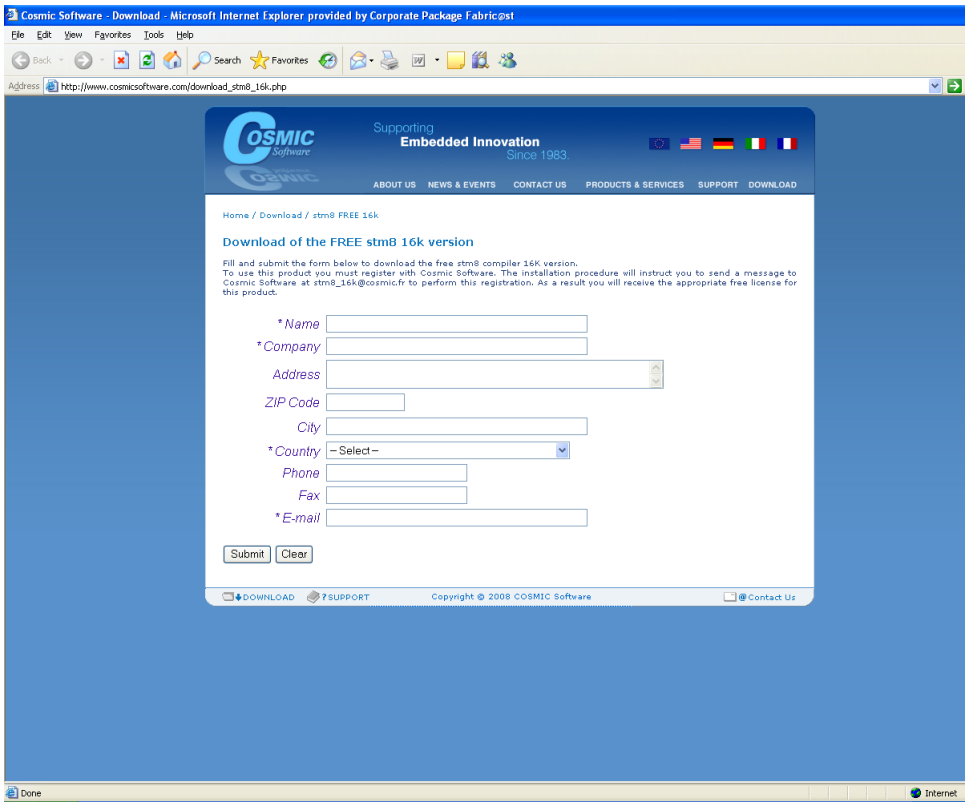
2.2.1 Installing the Cosmic compilerTo install the Cosmic compiler, follow the sequence described below:

1. Step 1

   Connect to http://www.cosmicsoftware.com/download_stm8_16k.php, and fill in the information form (see *Figure 2*).

   Click **Submit** to access the download page.

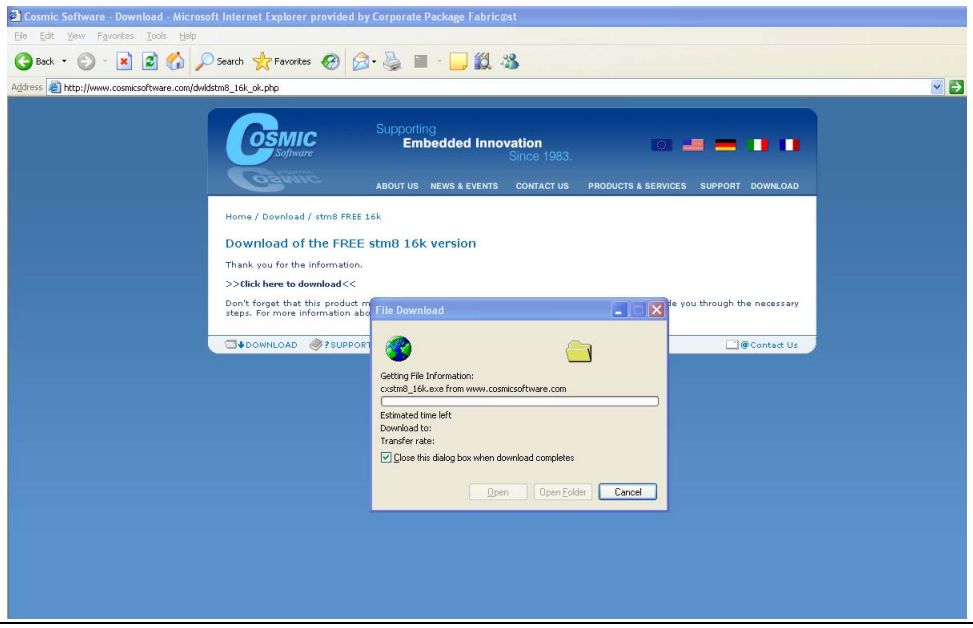
Figure 2. Cosmic compiler form



2. Step 2

   Download and run the installation software wizard (see *Figure 3*).

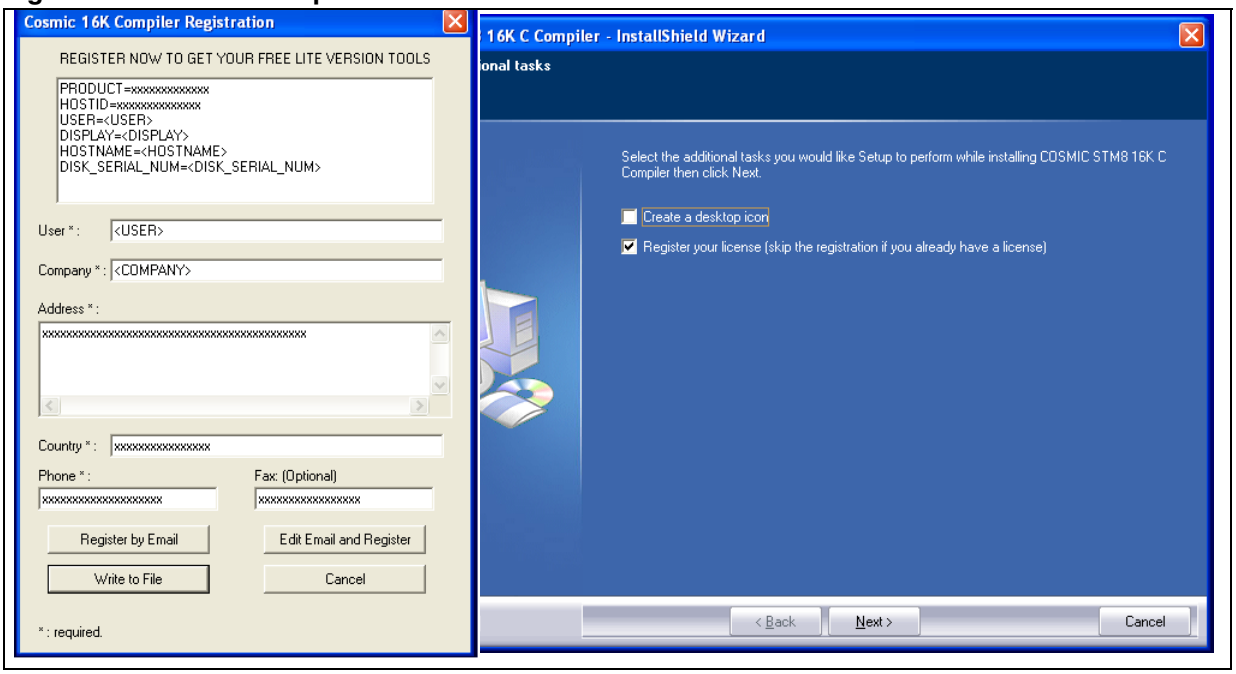
Figure 3. Downloading Cosmic compiler software wizard



3. Step 3

   Follow the wizard instructions to install the Compiler (see *Figure 4*). Do not forget to register for a free license and click **Register by Email**. You will receive the license file by e-mail within a few minutes' delay.

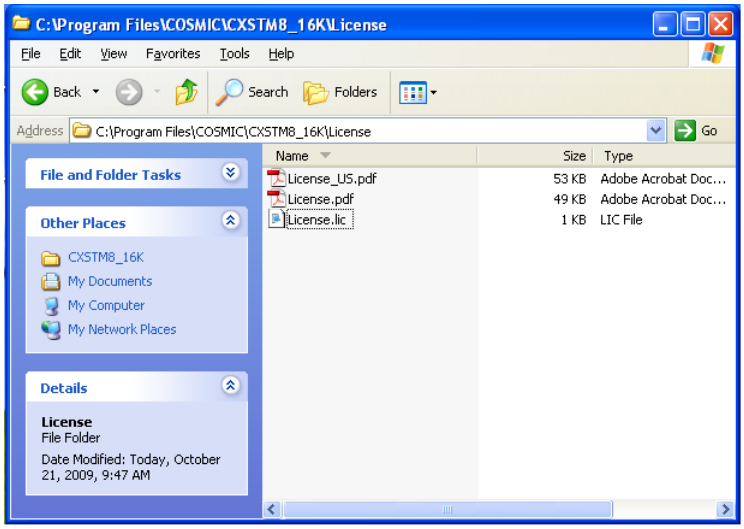
Figure 4. Cosmic compiler installation wizard



4. Step 4

   Copy the licence file that you have received into the directory ***\Program files\COSMIC\CXSTM8_16K\License*** *(see Figure 5).*

Figure 5. Cosmic compiler

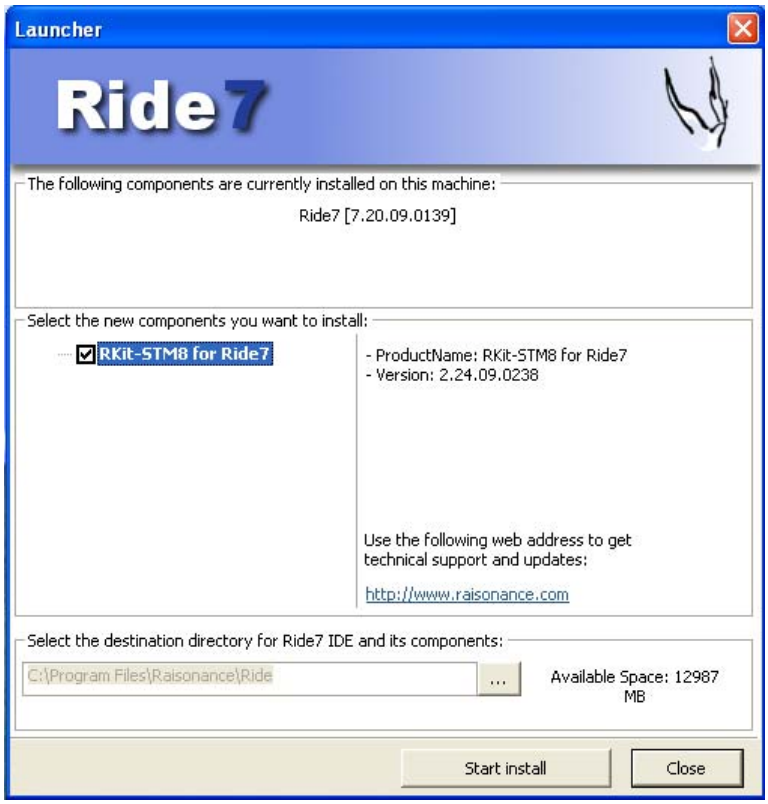


2.2.2 Installing the Raisonance compilerTo install the Raisonance compiler, follow the sequence described below:

1. Step 1

   Download Ride and the Raisonance kit RKit-STM8 from http://www.raisonance-mcu.com, and follow the installation wizard instructions (see *Figure 6*).

Figure 6. Raisonance compiler installation wizard



2. Step 2 - Register for a free 16 Kbyte license

   a) You need a computer serial number. To get it, open Ride, click **Help**>**Licence**, and enter your name, you company, and select **Manual installation**. The serial key will appear as shown in *Figure 7*.

   b) To get your free 16 Kbyte activation code, go to http://www.raisonance-mcu.com. Fill the question form with your personal information (see *Figure 8*), and the registration form with your serial key (see *Figure 9*).

   After this operation, you will receive your activation code by e-mail, and you will be granted a free 16 Kbyte Raisonance compiler licence to be used by STVD.

Figure 7. Activation code registration

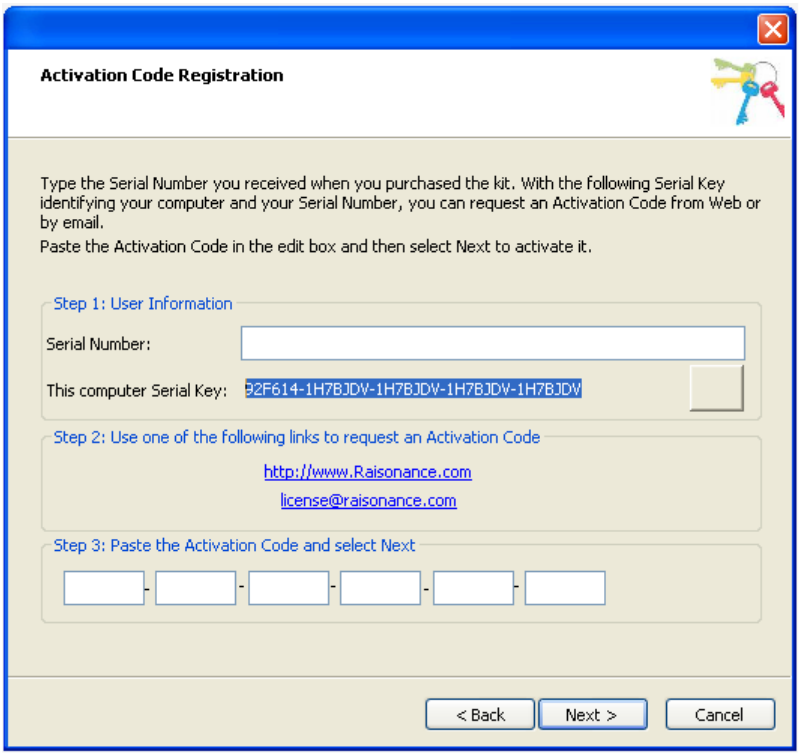


Figure 8. Personal information form

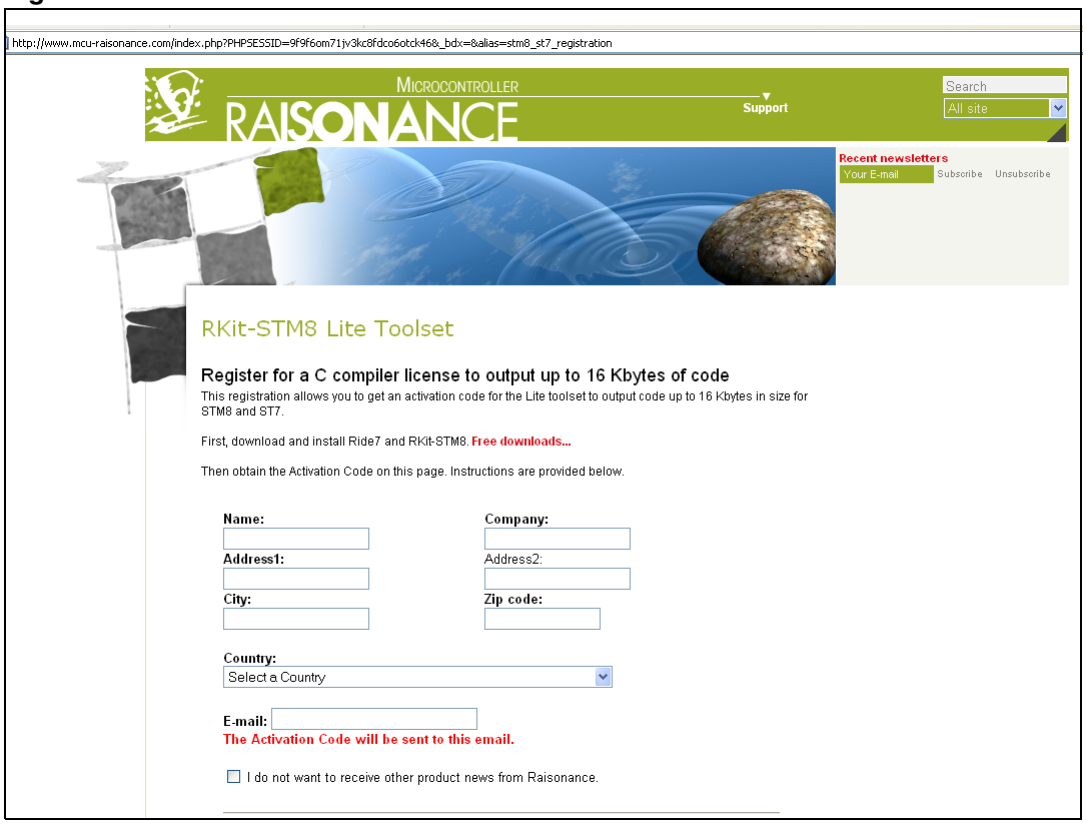
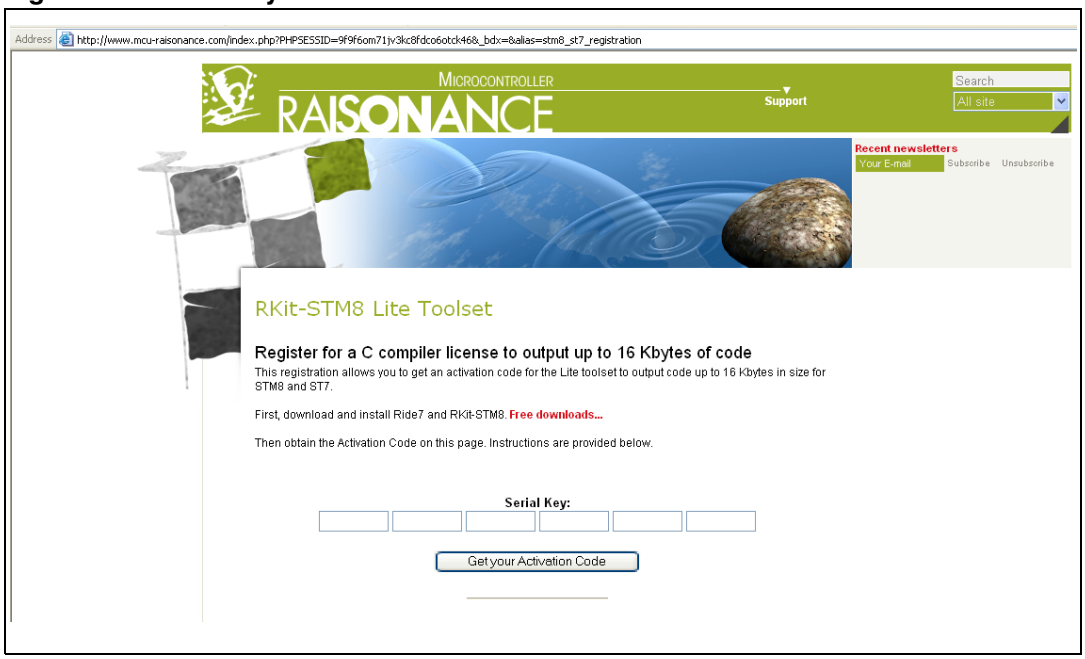


Figure 9. Serial key information form

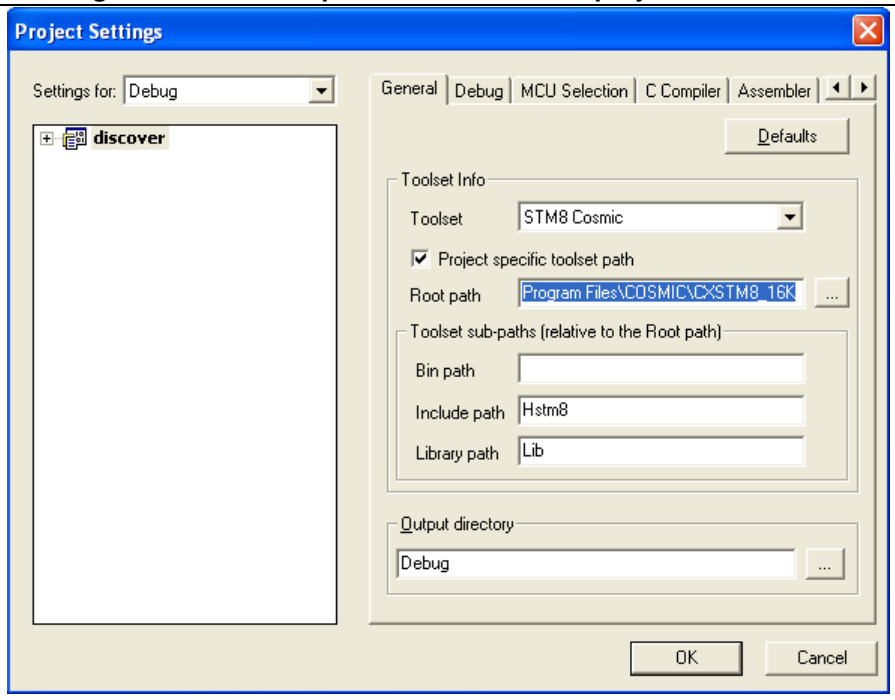


2.2.3 Compiler settings

STVD uses the default compiler defined at the first launch of the toolchain. This compiler is not defined during the installation.

You can select a different compiler for a specific project in the **General tab** of the STVD **Project**>**Settings** window.

Figure 10. Selecting the Cosmic compiler for the Discover project



3 Description of firmware libraries

The STM8S microcontroller family is provided with both the STM8S standard firmware library and the touch sensing firmware library.

It is strongly recommended to use the libraries to develop your project as proposed in the the STM8S-Discovery development package, and implemented in the Discover, PWM and ADC projects.

The STM8S standard firmware library and the STM8S touch sensing library are part of the STM8S-DISCOVERY development package that is available from the STM8S-DISCOVERY web page at http://www.st.com/mcu.

Warning: The STM8S touch sensing library includes the STM8S standard firmware library. Make sure you have the latest version of each library (see http://www.st.com/mcu).

3.1 STM8S standard firmware library

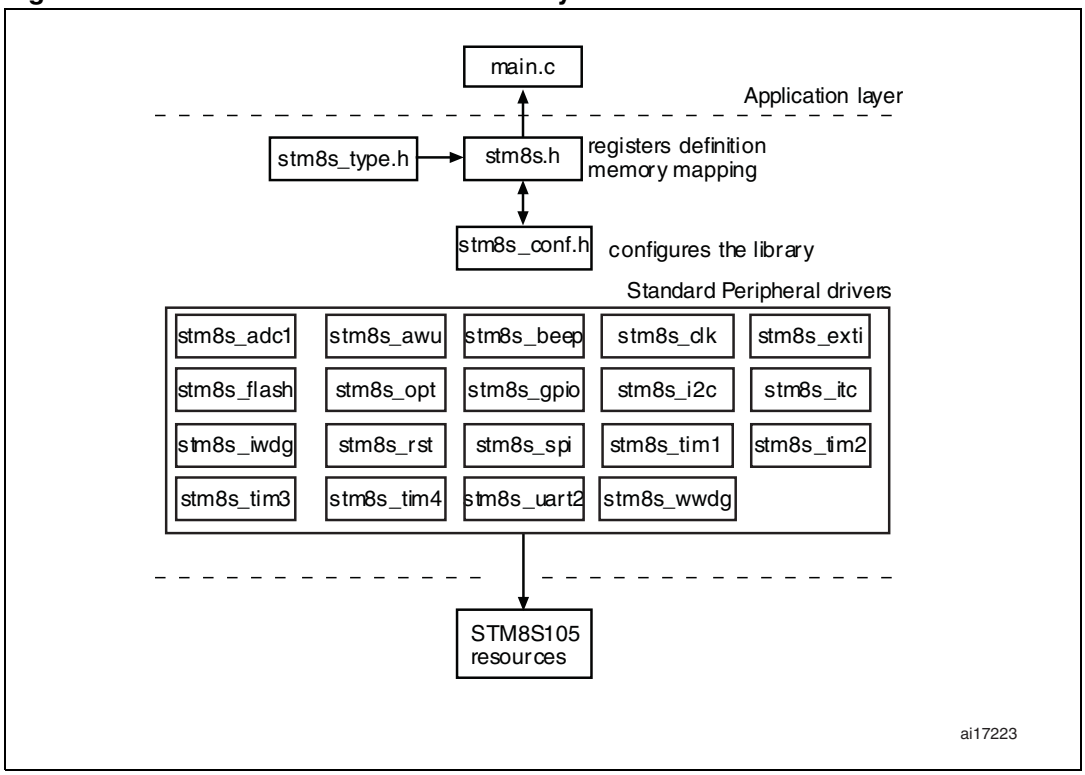
The STM8S standard firmware library contains a collection of routines, data structures, and macros covering the features of the STM8S peripherals, as well as a description of the device drivers (see *Figure 11*).

The STM8S standard firmware library allows to develop an application on any STM8S device without the need for in-depth study of each peripheral specifications. It saves significant time that would otherwise be spent in coding, while simultaneously reducing application development and integration cost.

It contains a set of example for each peripheral. All these examples are provided with workspace and project definition files for STVD and Cosmic C compiler to allow loading and compiling them easily into you development environment. These examples are developed to run on an STM8S208xx device with STMicroelectronics STM8/128-EVAL evaluation board.

They have to be considered as example codes to develop your own application. Some examples have already been tailored to run on the STM8S-DISCOVERY. They are available from the STM8S-DISCOVERY web pages at http://www.st.com/mcu.

Figure 11. STM8S standard firmware library architecture



3.1.1 Using the STM8S standard firmware library

To develop your own application using the STM8S standard firmware library, copy and tailor the Discover, PWM or ADC project delivered with the STM8S-DISCOVERY development package.

The *stm8s.h* header file contains the definitions of peripheral constants and register structures for all peripherals. Uncomment USE_STDPERIPH_DRIVER define statement when using the STM8S standard firmware library. Make sure that stm8s.h is included in your main() routine.

The *stm8s_conf.h* file of the STM8S standard firmware library is used to configure the library by enabling the peripheral functions that are only used by your application (see *Figure 12*). In *stm8s_conf.h*, some peripheral define statements are conditioned by supported devices. For example:

```
#if defined(STM8S208) || defined(STM8S207) || defined(STM8S105)
#define _TIM3 (1)
#endif
```

In *stm8s_conf.h*, the HSE value define statement may be adjusted to the oscillator frequency or to the external clock generator frequency. It is also conditioned by supported devices. Make sure you have the correct value for the STM8S-DISCOVERY external oscillator (expressed in Hz). For example:

```
#define HSE_VALUE ((u32)16000000)
```

The flow that must be followed to create your application software using the STM8S standard firmware library is described in *Section 5: Building your STVD project*.

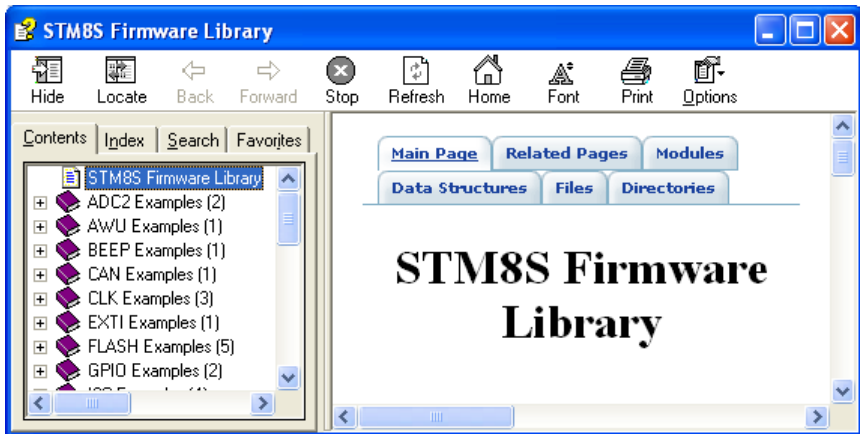
Figure 12. *stm8s_conf.h* peripheral define statements

```
/*************************************** CLK ***********************
 #define _CLK (1)

/*************************************** EXTI **********************
/* #define _EXTI (1) */

/********************************* FLASH/DATA EEPROM ***************
/* #define _FLASH (1) */

/********************************** OPTION BYTES *******************
/* #define _OPT (1) */

/*************************************** GPIO **********************
#define _GPIO (1)
```

3.1.2 STM8S standard firmware library online help

An online help, *stm8s_fwlib_um.chm,* is available inside the firmware installation directory to help you with the structure of the library (see *Figure 13*).

Figure 13. STM8 firmware library online help home page



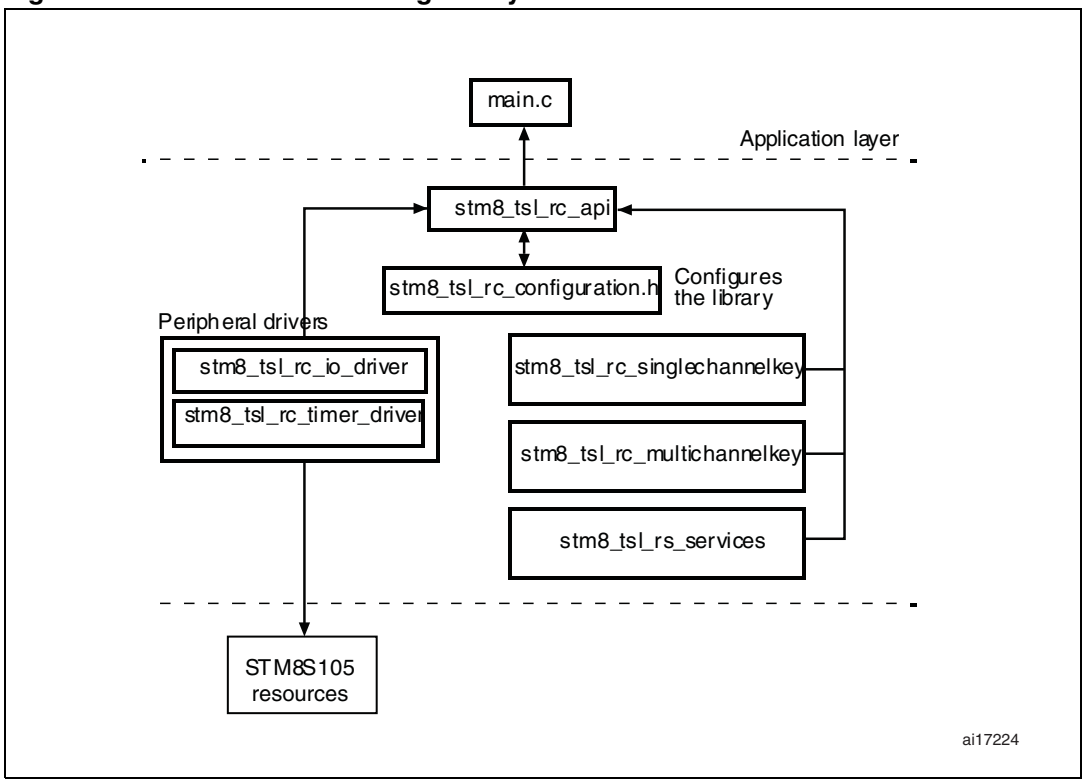
3.2 STM8S touch sensing library

The STM8S touch sensing library is composed of a set of functions, variables and structures (see *Figure 14*).

The STM8S touch sensing library allows detecting physical touch on the capacitive sensing keys by controlling the charge/discharge timing cycle of an RC network composed of a single resistor and the touch electrode capacitance. Any variation in the RC charge/discharge timing resulting from a change in the electrode capacitance is detected and then filtered.

For detailed information related to the RC acquisition principle, refer to the application note AN2927 available on http://www.st.com/mcu.

Figure 14. STM8S touch sensing library software architecture



3.2.1 STM8S touch sensing library working principles

The STM8S touch sensing library is a set of compatible C files. It includes the API that makes the interface with the other software layers, a core set of files performing basic functions (main state machine, key state machine, services), plus the drivers to control timers and GPIOs.

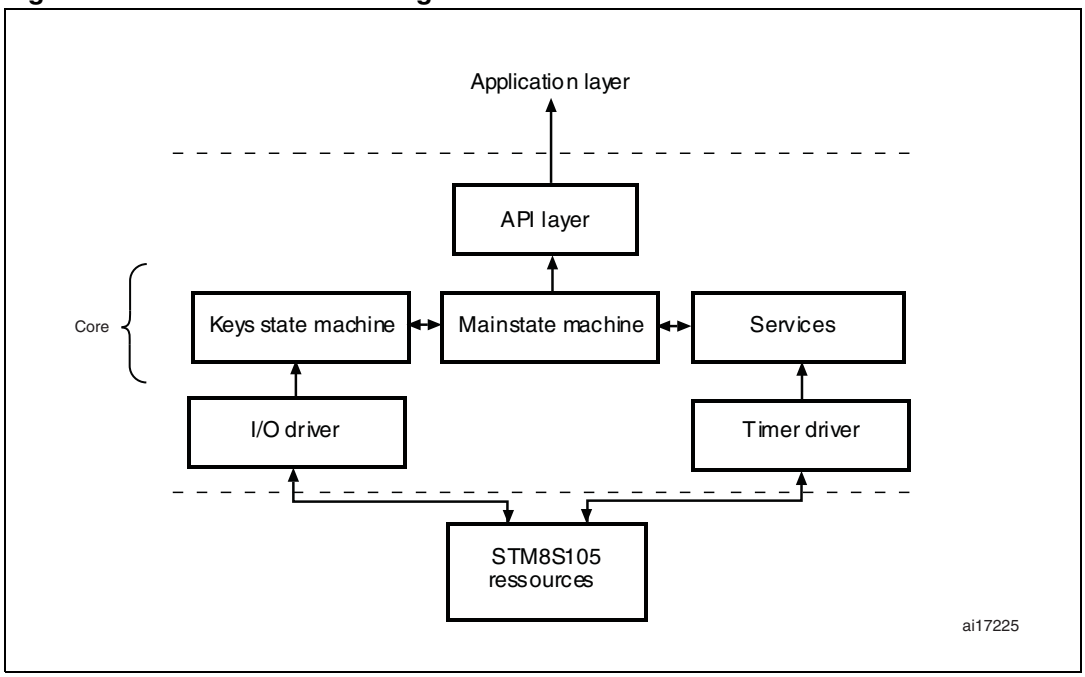
The touch sensing library is based on a 2 state machines (see *Figure 15*):

- Main state machine

  The main state machine manages the sequence of the common actions concerning the touch sensing keys:
  - Signal acquisition
  - Data interpretation
- Key state machine

  Each key has its own state machine to manage its states: calibration, idle, detect, etc...

Figure 15. STM8S touch sensing functional architecture



3.2.2 Using the STM8S touch sensing library

In applications using the STM8S touch sensing library, configure and link the library to the project. Make sure that:

1. The header file *stm8_tsl_rc_api.h* is included in your main routine.
2. The touch sensing library configuration parameters are correctly set in compliance with your application hardware.

   Open the file *STM8_TSL_RC_Configuration.h*, and check that the define statements are filled in with the correct values. The other define statements must remain unchanged (thresholds, integrator settings, IIR filter settings,...).

See *Table 1* for an example of define statement configuration for an application using the single touch sensing button of STM8S-DISCOVERY (TS1).

Note: The STM8_TSL_RC_Configuration.h file can be updated to support up to 24 keys and 2 wheels or sliders.

Table 1. Touch Sensing library configuration for STM8S-DISCOVERY using TS1 touchkey

Function	#define statement	Value	Comment
MCU selection	STM8S	1	-
Acquisition timer	TIMACQ TIMACQ_CNTR_ADD	TIM3 0x5328	TIM3 base address
Time-base timer	TIMTICK	TIM4	-
Load I/O	LOADREF_PORT_ADDR LOADREF_BIT	GPIOC_BaseAdress 0x04	Port PC4 selected
Single channel key	SCKEY_P1_KEY_COUNT SCKEY_P1_PORT_ADDR SCKEY_P1_A SCKEY_P1_DRIVEN_SHIELD_MASK SCKEY_P2_COUNT SCKEY_P3_COUNT	1 GPIOC_BaseAddress 0x02 0x08 0 0	Number of keys = 1 Port PC selected Pin 1 selected as acquisition input Pin 3 for active shield Key port P2 not used Key port P3 not used
Multichannel key	NUMBER_OF_MULTI_CHANNEL_KEYS	0	Multichannel key feature disabled
Electrode mask	GPIOA_ELECTRODES_MASK GPIOB_ELECTRODES_MASK GPIOC_ELECTRODES_MASK GPIOD_ELECTRODES_MASK GPIOE_ELECTRODES_MASK GPIOF_ELECTRODES_MASK GPIOG_ELECTRODES_MASK GPIOH_ELECTRODES_MASK GPIOI_ELECTRODES_MASK	0x00 0x00 0x0A 0x00 0x00 0x00 0x00 0x00 0x00	Defines the electrode mask for each GPIO used. Mask must be set to 0x00 for unused GPIOs.

3.2.3 Developing your application with the STM8S touch sensing library

Figure 16 shows the main flowchart that is to be followed to develop an application using the touch sensing library with the STM8S-DISCOVERY. The application code must call some API functions to initialize the library and manage the acquisition state machines.

The function *TSL_Init()* must be called once during the initialization of the complete system. It is usually done in the main routine.

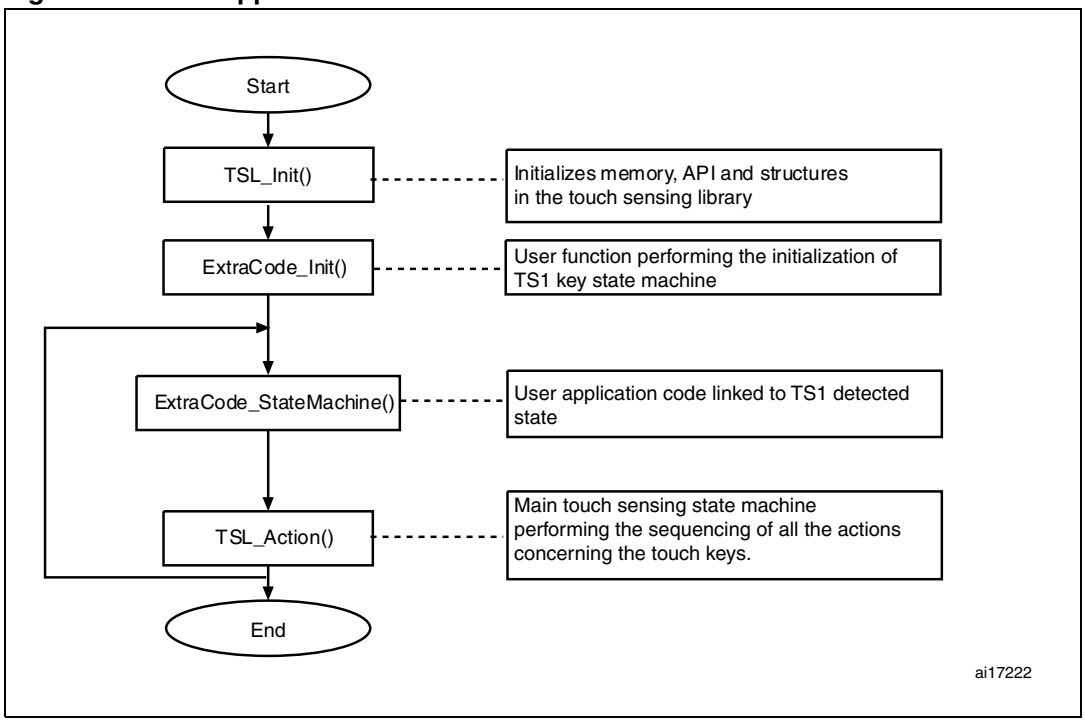
The *ExtraCode_Init()* performs the initialization of all the touchkeys implemented and enabled in the *STM8_TSL_RC_Configuration.h*.

The function *TSL_Action()* must be called periodically during system execution. It is the main state machine which sequences all the actions concerning all the touchkeys.

The application user code must be developed inside the function *ExtraCode_StateMachine()*. As an example, download the Discover application demonstration available in the STM8S-Discovery web pages.

To read the data set by the library (for example to check flags), ensure that the touch sensing main state machine is in idle state to avoid spurious values to be retrieved.

Figure 16. User application flowchart



3.2.4 STM8S touch sensing library online help

An online help, *stm8_tsl_um.chm*, is available in the STM8S touch sensing library installation directory to help you with the structure of the library (see *Figure 17*).

Figure 17. STM8S touch sensing library online help home page

4 Configuring the option bytes for your application

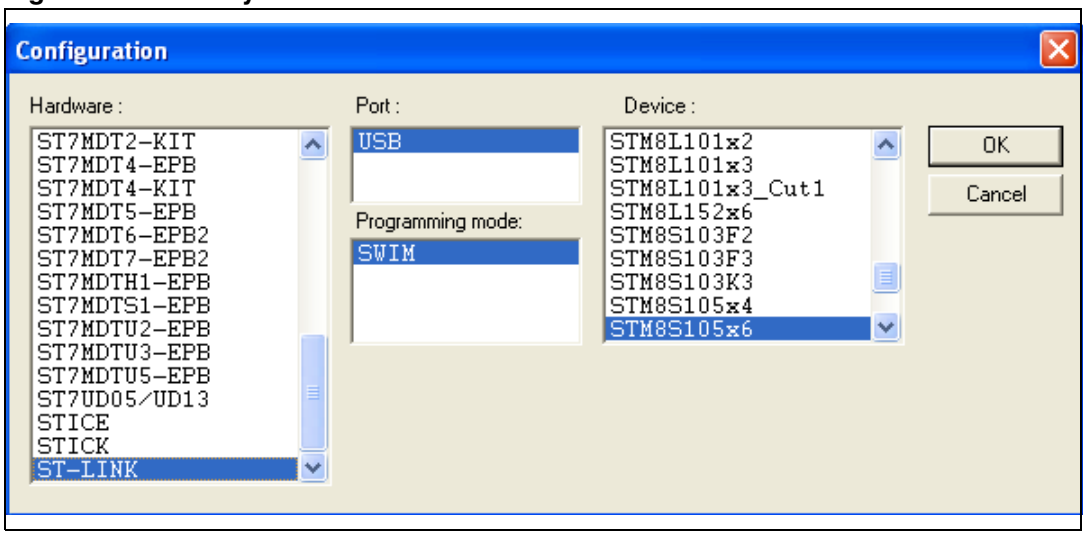
The option bytes allow to configure the device hardware features and memory protection. They are stored in a dedicated memory block. Refer to the option bytes section of the STM8S105xx datasheet for a detailed description.

STVP can be used to program the STM8S option bytes. It is part of the free software package provided by STMicroelectronics. This easy-to-use tool has a graphical interface allowing to read, write and verify the STM8 Flash programming memory, data EEPROM and option bytes.

The STVP executable is available in available in the Windows **start** menu.

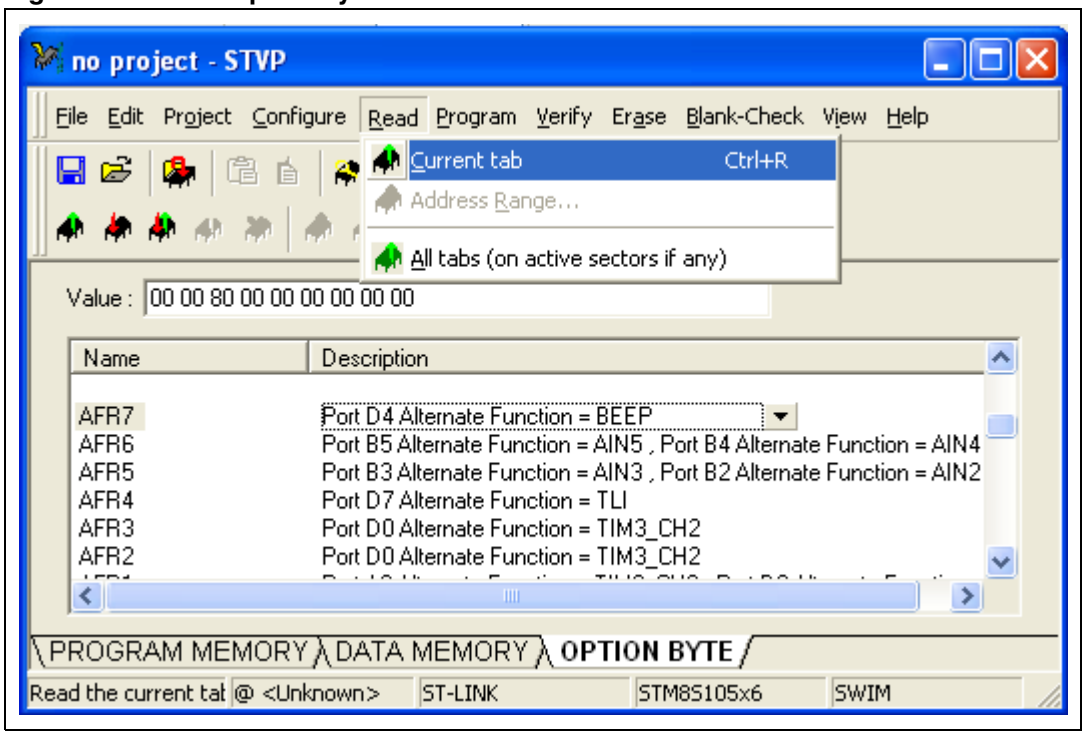
1. Open the STVP GUI and select **Configure>Configure ST Visual Programmer** from the main menu toolbar. The configuration dialog box opens.
2. In the **hardware** list, select ST-LINK as programming board.
3. In the **port** list, select USB as the host PC port to which the ST-LINK is connected. Only the ports that are compatible with the selected hardware are displayed.
4. in the **programming mode** list, select SWIM. The programming modes displayed depend on your programming equipment.
5. In the **device** list, select STM8S105x6 as the ST microcontroller you are going to program (see *Figure 18*).

Figure 18. Select your MCU



6. Select the **OPTION BYTE** tab, and click **Read >Current tab** in the menu toolbar. All the STM8S105 option bytes and their respective values are displayed in the window (see *Figure 19*).

Figure 19. STVP option byte selection menu



7. To program an option byte to a new value:
   a) Click the option byte description, and select the value.
   b) Click **Program>Current tab** in the main menu toolbar (see *Figure 20*).
   c) If the operation has completed successfully, the Output windows is displayed (see *Figure 21*).

Figure 20. STVP option byte programming menu

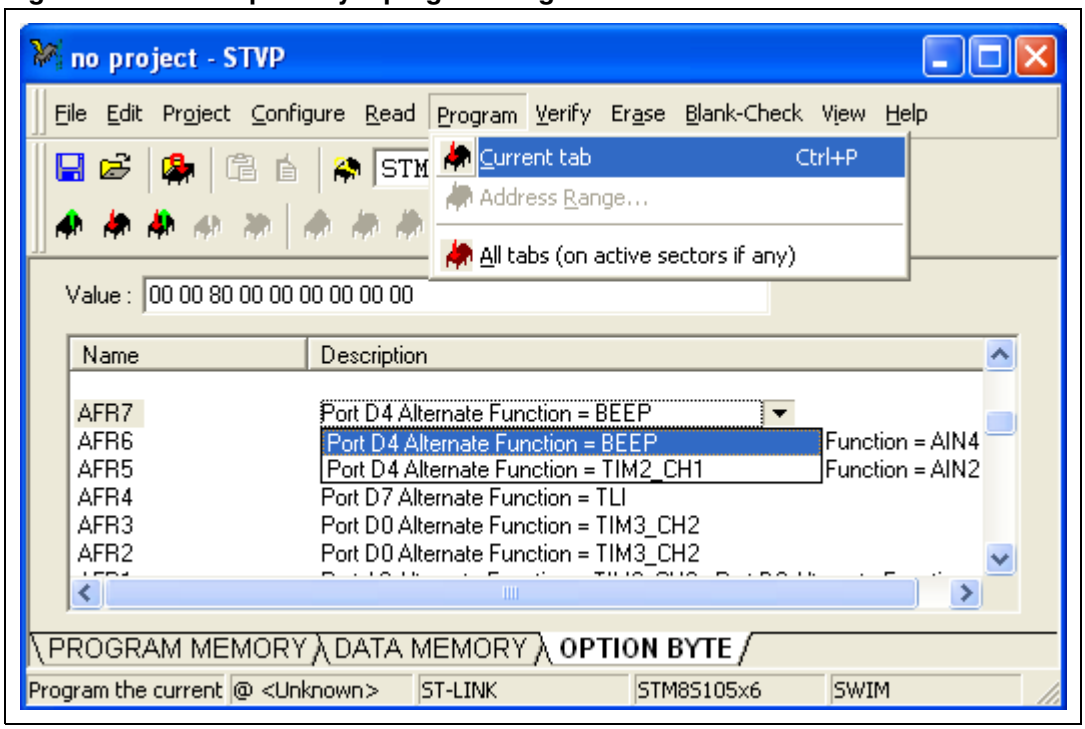
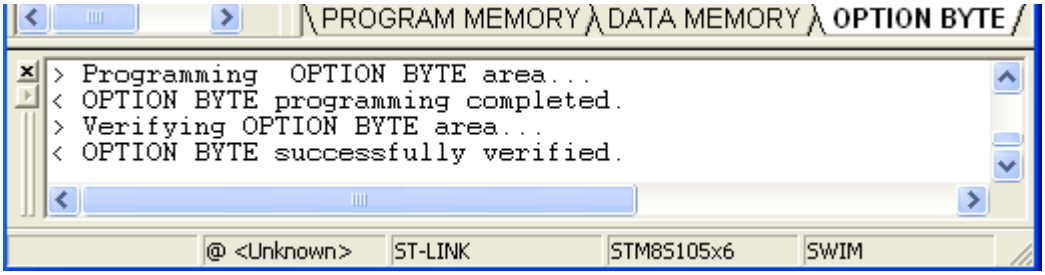


Figure 21. STVP option byte programming message



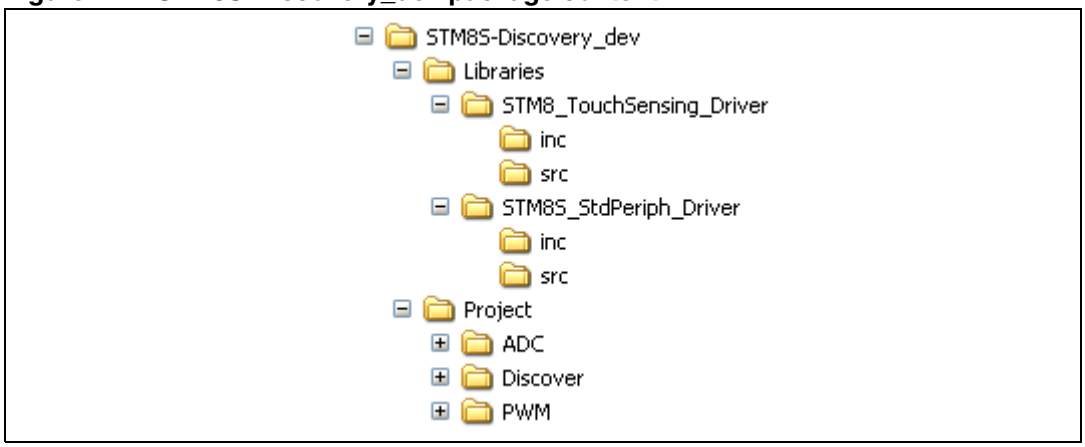
5 Building your STVD project

5.1 Description of the STM8S-DISCOVERY development package

To simplify the development of your application code, ST provides a development package, the STM8S-Discovery_dev which can be downloaded from the STM8S-Discovery web page. This package has a predefined structure and can be used as an example to develop an application (or project) running on the STM8S-DISCOVERY. The package directory is structured as follows (see *Figure 22*):

- **STM8_TouchSensing_Driver**: this directory contains the sources and header files of the STM8S touch sensing library.
- **STM8S_StdPeriph_Driver**: this directory contains the sources and header files of STM8S standard firmware library.
- **Discover**: this directory contains the STM8S-Discovery startup example provided with the kit. It is based on the STM8S standard firmware and STM8S touch sensing libraries. For more details on this example please refers to the description of Discover application (UM0833) available on STMS-Discovery web page.
- **PWM**: this directory contains an example using only the STM8S standard firmware library.
- **ADC**: this directory contains an example which does not use the libraries.

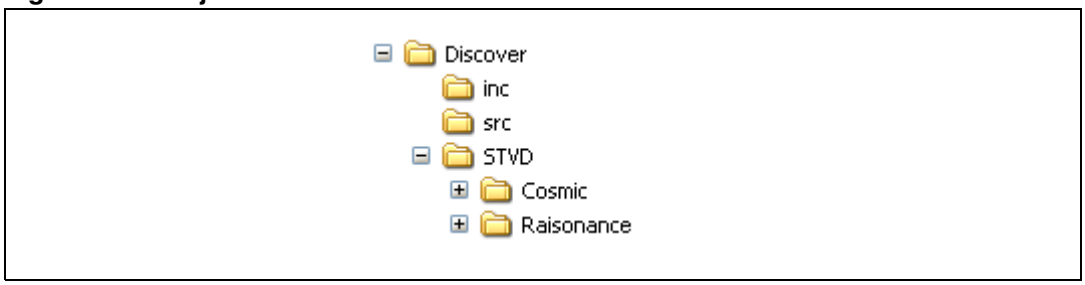
Figure 22. STM8S-Discovery_dev package content



All projects must be created starting from this package. They should be structured as follows (see *Figure 23*):

- **inc:** this directory contains all the header files for your application, including the configuration files for the touch sensing and the standard firmware libraries. These files are used to tailor the libraries for your application.
- **Cosmic/Raisonance directory**: this directory is used to store STVD workspace file (*dicover.stw*). It includes the project and workspace for the Cosmic or Raisonance compiler, depending on the compiler you have selected.
- **src:** this directory contains all the source files dedicated to your application code, such as *main.c* and *stm8_interrupt_vector.c*.

Figure 23. Project structure



5.2 Creating a project using STVD

This section explains step by step how to create and configure a project. The touch sensing and the firmware libraries can be used or not, according to the kind of application code to develop:

1. Step 1

   To create your project with the correct structure:

   a) Copy the **STM8S-Discovery_dev** directory and rename it "**STM8S-Discovery_My_own_project**".

   b) Copy one of the project examples according to the type of application you want to develop (see *Figure 24*); rename it using the same name as the root directory:

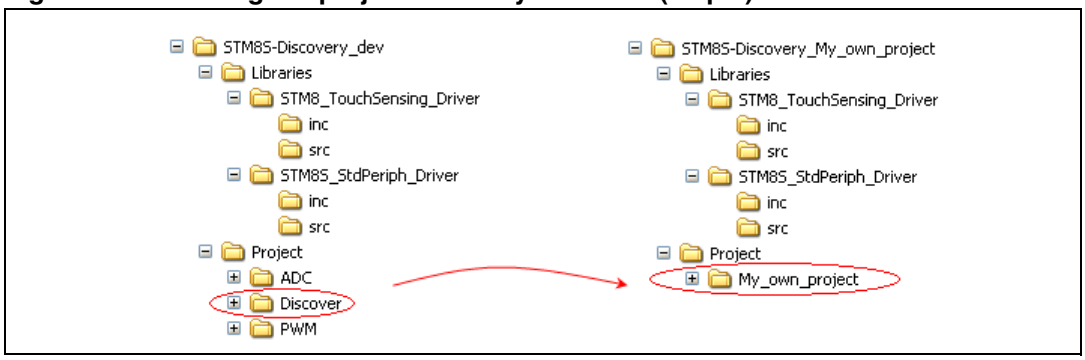
   – Use Discover to create an application using the STM8S standard firmware library and the touch sensing library.

   – Use PWM to create an application using only the STM8S standard firmware library.

   – Use ADC if you do not need the libraries.

Warning: Each project must have the same structure as the Discover, PWM or ADC project.

Figure 24. Creating the project directory structure (step 1)



2. Step 2
   a) Open STVD and use the **New workspace wizard** by clicking **File>New Workspace** (see *Figure 25*).
   b) Click **Create workspace and project**.
   c) Enter your workspace name (referred to as 'My_workspace') in the **workspace filename** field.
   d) Browse for the **Workspace location** and select Cosmic (**\"STM8S-Discovery_My_own_project"\Project\"My_own_project"\STVD\Cosmic**) or Raisonance (\"**STM8S-Discovery_My_own_project"\Project\"My_own_project"\STVD\Raisonanc**e) depending on the compiler you need.
   e) Enter your project name in the **Project filename** field (see *Figure 26*).
   f) Enter the **Project location** in the **Project location** field.
   g) Browse for the **Project location** and select Cosmic (**\"STM8S-Discovery_My_own_project"\Project\"My_own_project"\STVD\Cosmic**) or Raisonance (**\"STM8S-Discovery_My_own_project"\Project\"My_own_project"\STVD\Raisonance**) depending on the compiler you have need.
   h) Select the toolchain to be used and verify that the toolchain root path is correct.
   i) Go to **Projects**>**Settings**>**MCU Selection**, and select the **STM8S105C6** from the MCUs list. Make sure that the MCU selected is displayed in the **Selected MCU** field before clicking OK (see *Figure 27*).

Figure 25. Define your workspace (step 2)

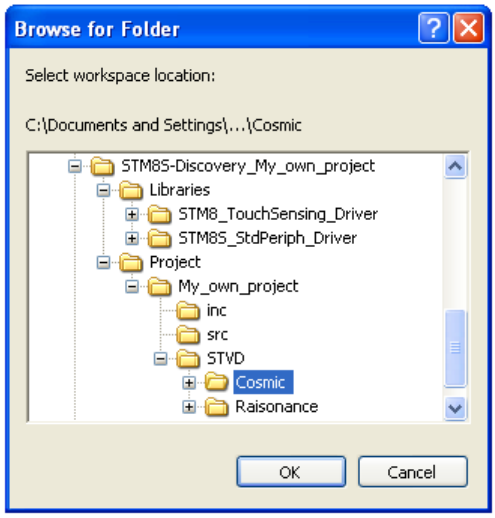


Figure 26. Define your project name and location (step 2)

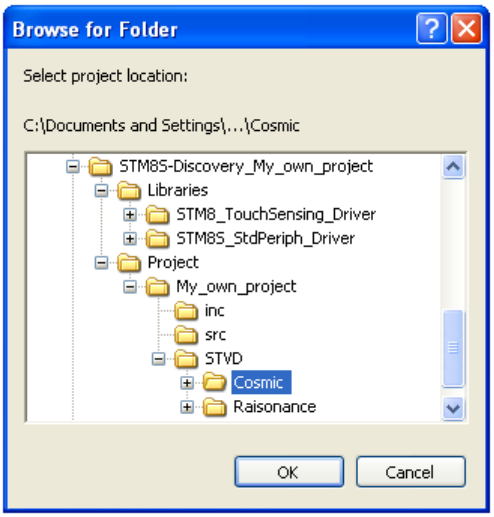
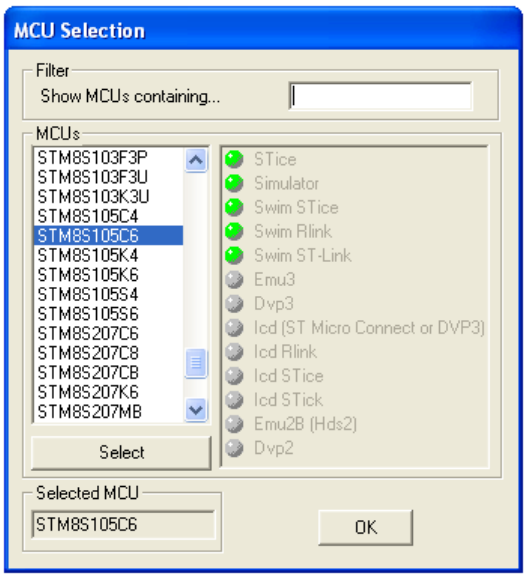
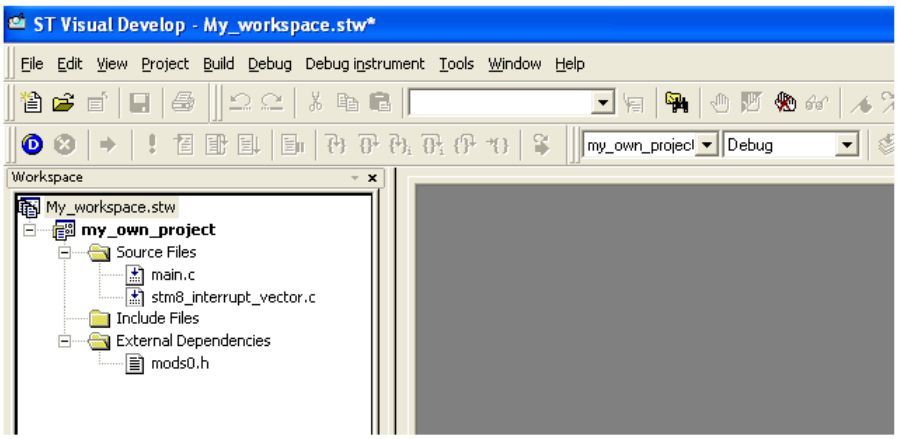


Figure 27. Select the MCU



The new project appears in the STVD workspace window as shown below. STVD automatically generates empty *main.c* and *stm8_interrupt_vector.c* files in the STVD project and under \"**STM8S-Discovery_My_own_project"\Project\"My_own_project"\STVD\Cosmic**.

Figure 28. My_Workspace window



3. Step 3
   a) To avoid errors, remove the automatically generated files from the STVD project directory and from (\"**STM8S-**

Discovery_My_own_project"\Project\"My_own_project"\STVD\Cosmic (see *Figure 29*).

b) Replace them by the files located in **\"STM8S-Discovery_My_own_project"\Project\"My_own_project"\src** (see *Figure 30*).

c) Click **Project>settings** from the STVD main menu. Select the **Linker** tab and the **Input** category (see *Figure 31*) and update the path to *stm8_interrupt_vector.c* file.

Figure 29. Removing automatically generated files

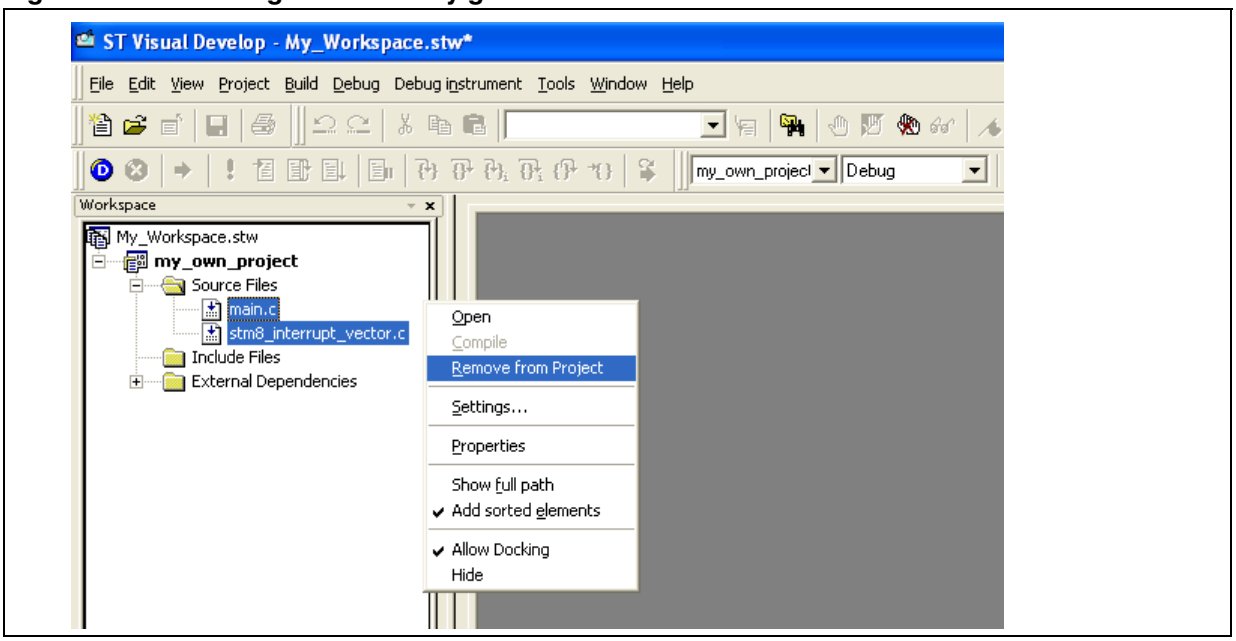


Figure 30. Replacing automatically generated files

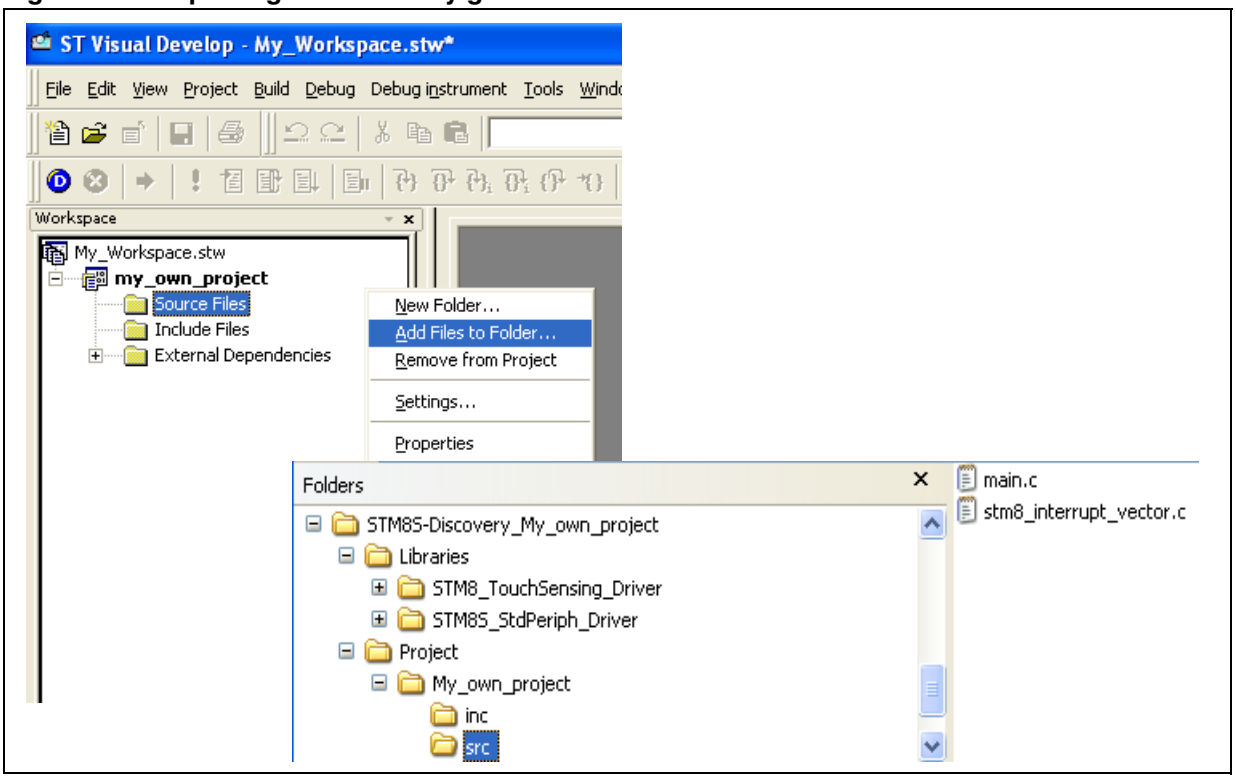
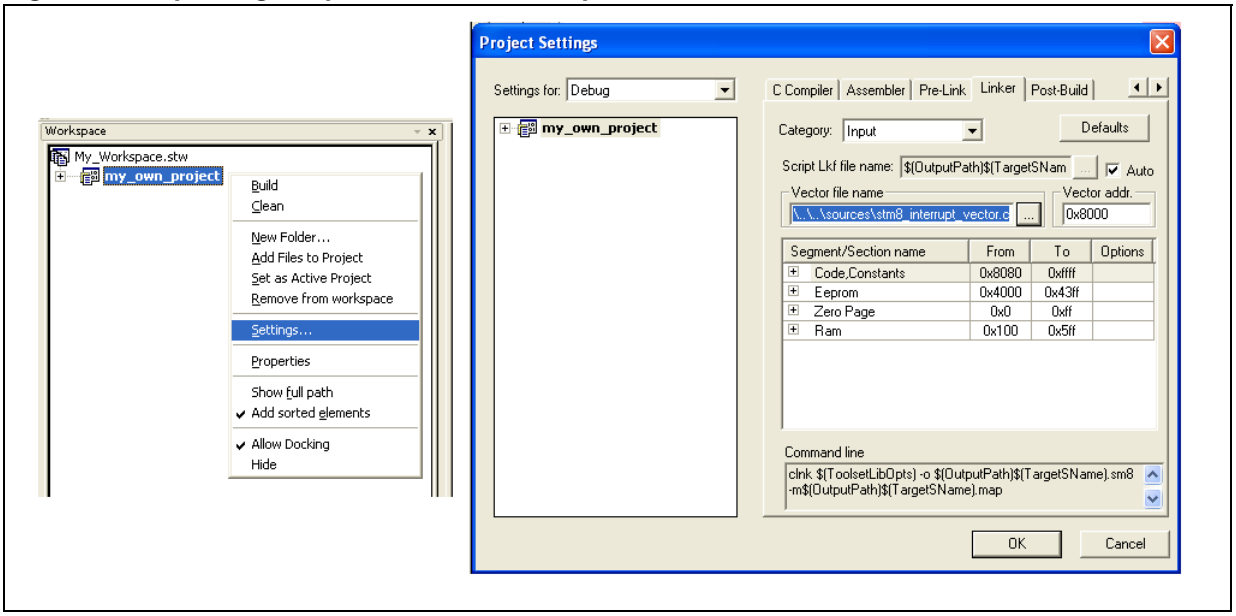


Figure 31. Updating the path to *stm8_interrupt_vector.c*



5.3 Updating the libraries

5.3.1 Updating the STM8S standard firmware library

To update the STM8S standard firmware library:

1. Download the new release from http://www.st.com/mcu.
2. Copy the **inc** and **src** directories in **\"STM8S-Discovery_My_own_project"\Libraries\STM8S_StdPeriph_Driver**.

5.3.2 Updating the STM8S touch sensing library

To update the STM8S touch sensing library:

1. Download the new release from http://www.st.com/mcu.
2. Copy the **inc** and **src** directories in **\"STM8S-Discovery_My_own_project"\Libraries\STM8_TouchSensing_Driver**.

5.4 Linking the STM8S standard firmware library to your project workspace

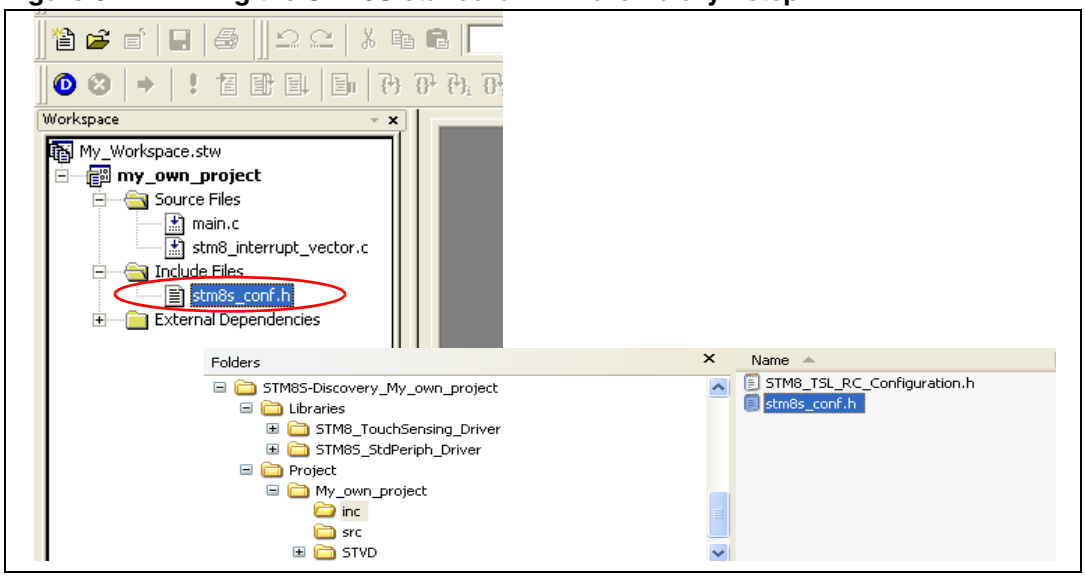
This section explains step by step how to link the standard firmware library to your project:

1. Step 1

   Copy *stm8s_conf.h* in your project directory (see *Figure 32*).

   Include the *stm8s.h* file in your *main.c* file, and uncomment the define statement `#define USE_STDPERIPH_DRIVER` in the *stm8s.h*.

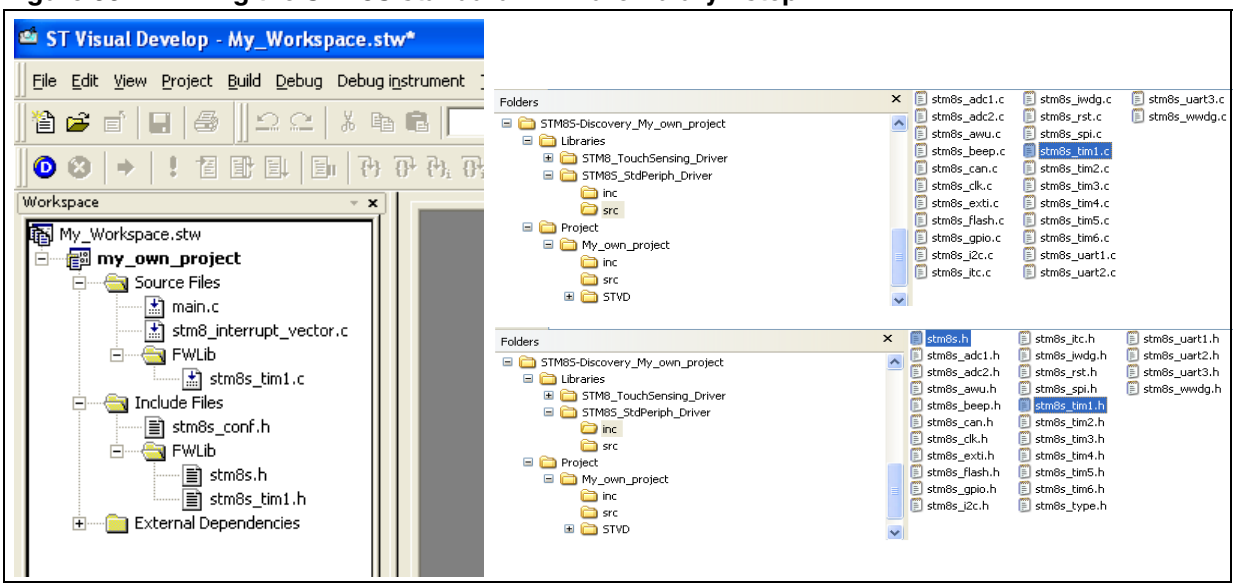
Figure 32. Linking the STM8S standard firmware library - step 1



2. Step 2
   a) Create two **FWLib** directories under **My_own_project>Source files** and **My_own_project>Include files** in your STVD project.
   b) Copy in both directories the source and header files associated to the STM8S and peripherals you are using. See *Figure 33* for an example showing how to add timer1.

*Note: It is mandatory to copy the stm8s.h file in your STVD project under **My_own_project\Include files** wether the STM8S standard firmware library is used or not.*

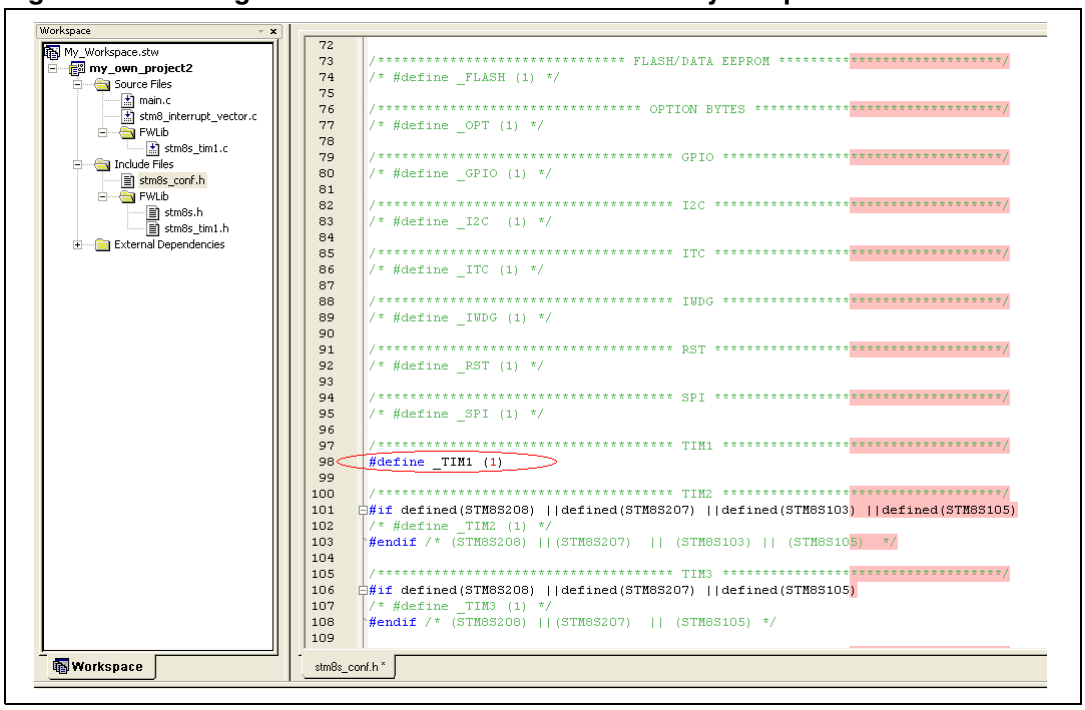
Figure 33. Linking the STM8S standard firmware library - step 2



3. Step 3

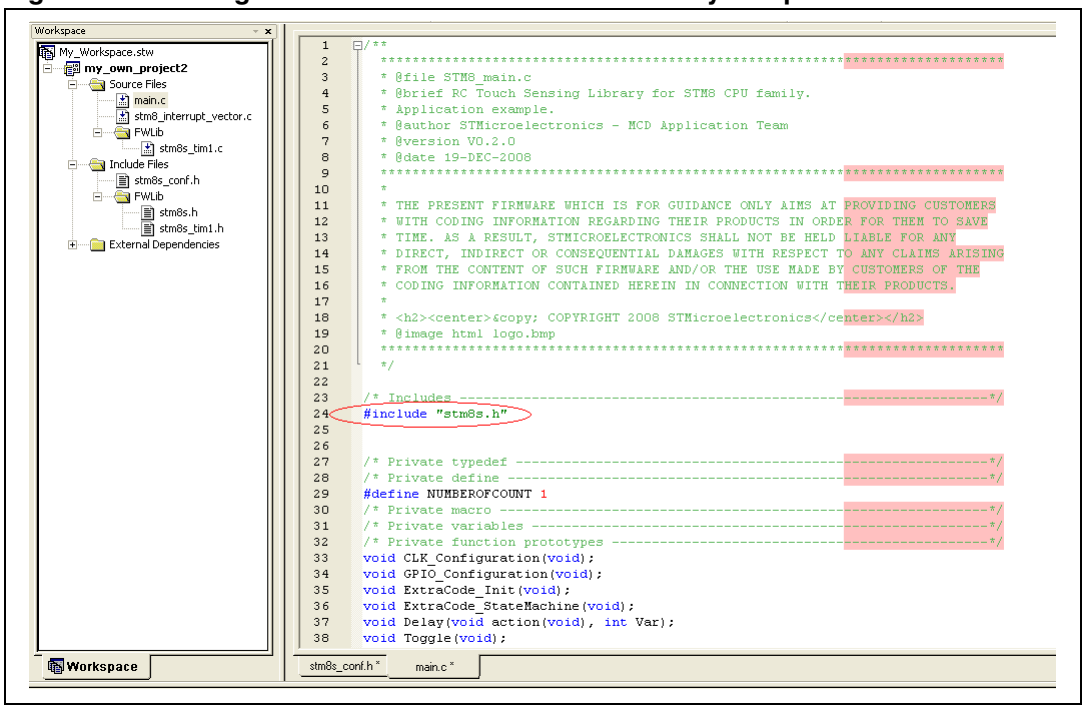
   In the *stm8s_conf.h* file, uncomment the define statements corresponding to the peripherals used by your application(see *Figure 34*). The corresponding drivers (.c and .h files) must be added in **My_own_project>Source files\FWLib** and **My_own_project>Include files\FWLib**.

Figure 34. Linking the STM8S standard firmware library - step 3



4. Step4: Check that the line `#include <stm8s.h>` is present in your main.c file. Otherwise add it (see *Figure 35*).

Figure 35. Linking the STM8S standard firmware library - step 4



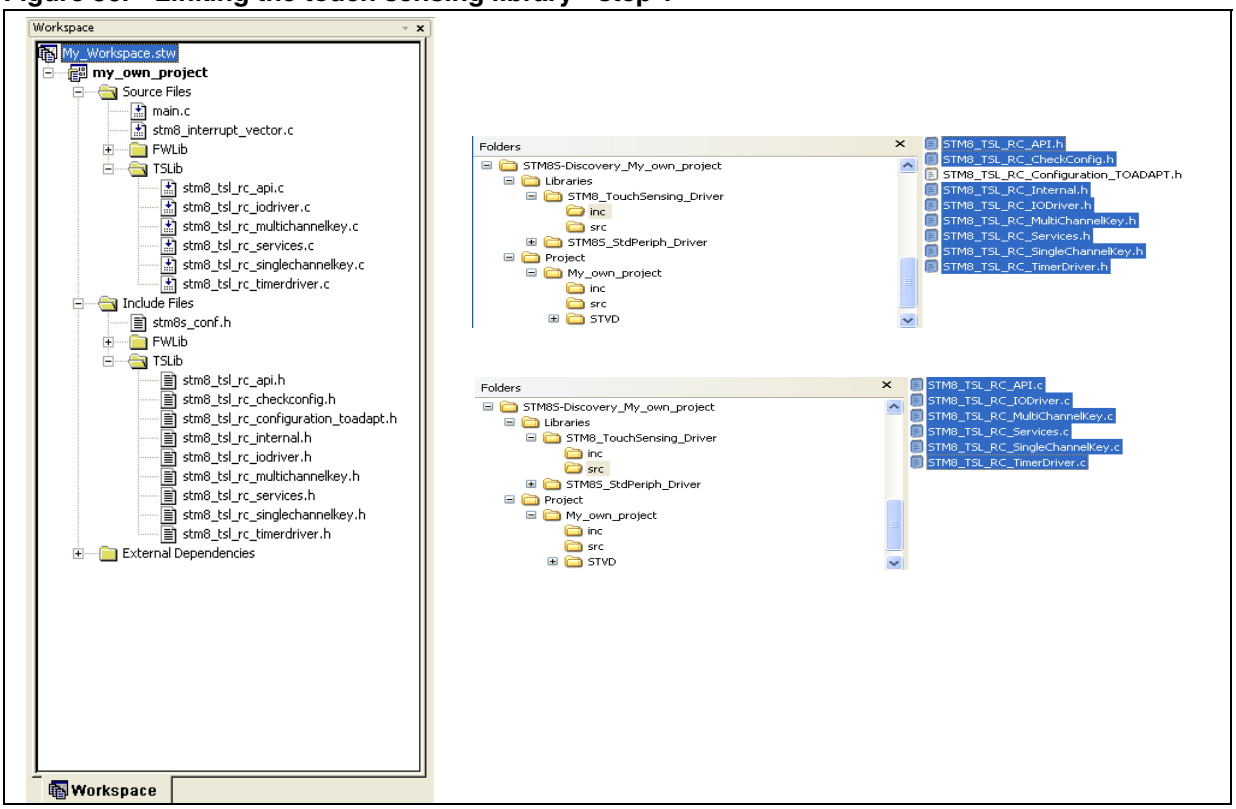
5.5 Linking the touch sensing library to your project workspace

All the source and header files from the touch sensing library are required to be able to use the touch sensing key or any external touch sensing peripherals.

Follow the steps below to link the touch sensing library to your workspace.

1. Step 1
   a) Create two **TSLib** directories under **My_own_project>Source files** and **My_own_project>Include files** in your STVD project.
   b) In **My_own_project>Source files**, and **My_own_project>Include files**, add all the touch sensing library source files and header files except for STM8_TSL_RC_Configuration_TOADAPT.h (see *Figure 36*).

Figure 36. Linking the touch sensing library - step 1



2. Step2
   Check that the line `#include <stm8_tsl_rc_api.h>` is present in your *main.c* file. Otherwise add it (see *Figure 37*).
3. Step 3
   Click **Project>settings** from the STVD main menu. Select the **Linker** tab and select the **Input** category (see *Figure 31*). Under **Segment/Section name**, add the TSL_IO_ALCODE in the **Code,Constants** section and assign the **-r2** option to it (see *Figure 38*).

Note: A specific main.c structure is required to correctly use the touch sensing features. This structure is already implemented in the Discover project. Please refer to Section 3.2: STM8S touch sensing library for more information.

Figure 37. Linking the touch sensing library - step 2

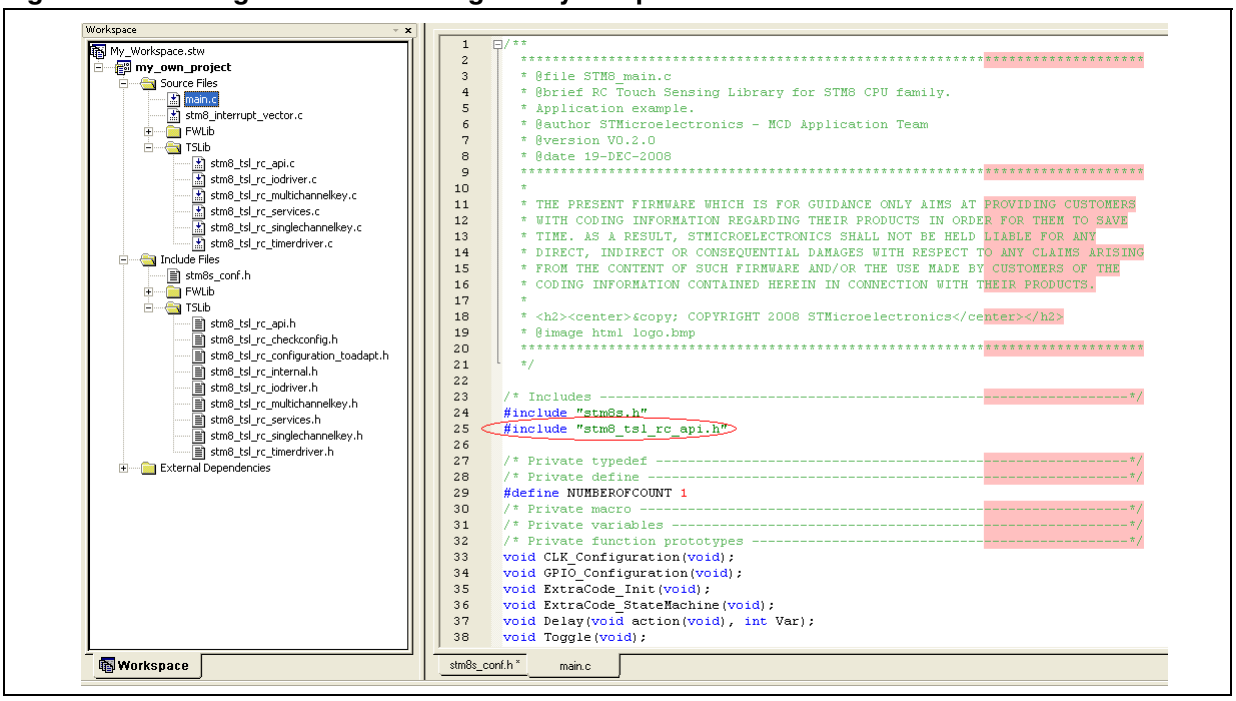
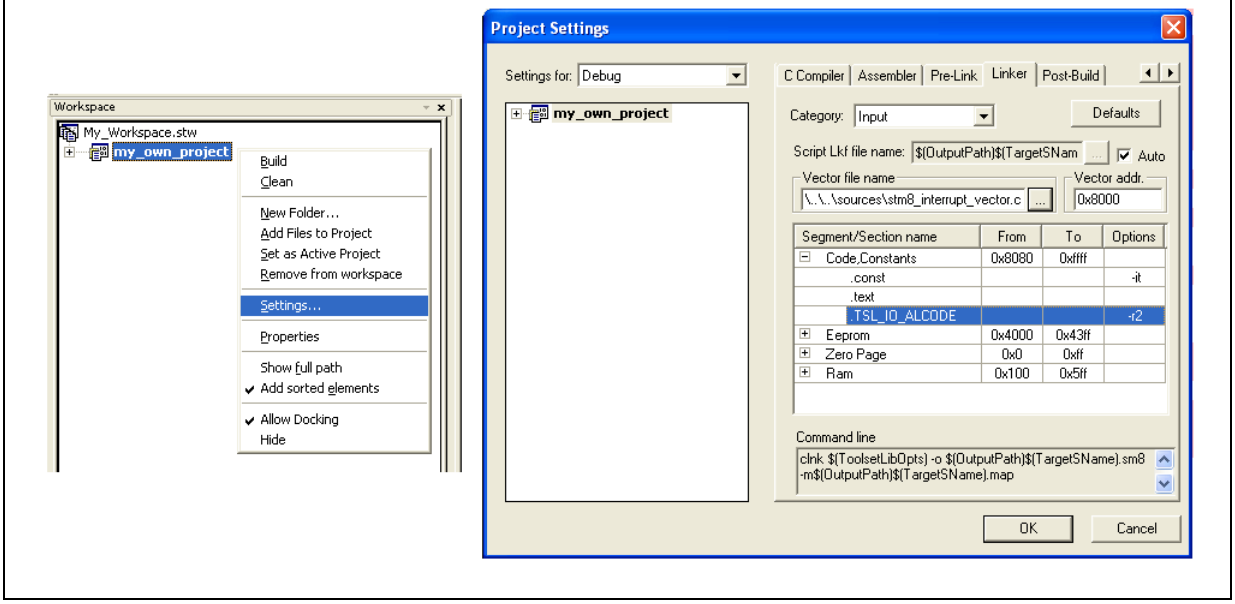


Figure 38. Linking the touch sensing library - step 3



5.6 No library linked to your project workspace

If your application does not require any library, tailor the ADC project.

Make sure that *stm8s.h* is included in your *main.c* file, and comment the `STD_PERIPH_DRIVER` define statement to be able to use the register structures, memory mapping, and constant definitions for each peripheral.

6 Building, debugging and running your application

Once your have developed your application code, created your workspace environment, and launched STVD, you can start building, debugging and programming your application to the target microcontroller.

6.1 Building your application

Once your project is created, the build context is enabled by default. It allows to access all the commands required to set up, customize, and build your application.

The build configuration is available by selecting ***Build>Configurations*** from the STVP main menu toolbar. It allows to change the application building settings.

Two preset configurations are available in STVD:

- **Debug**

  This configuration creates a version of your application that allows using all the STVD advanced debugging features. When using this configuration, output files are saved in the **Debug** directory in your **workspace** directory.

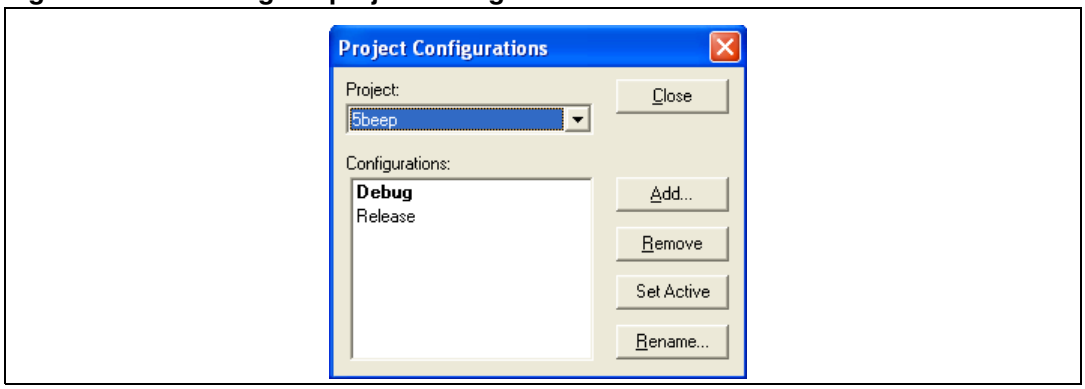
- **Release**

  This configuration creates a version of your application that uses the default optimization for your toolset. This version of your application is ready to be programmed to your target microcontroller.

Follow the sequence below to build a project:

1. Ensure that the **Debug** configuration is selected by clicking **Build>Configurations**. If the **Debug** configuration is not selected, highlight it and click **Set Active** (see *Figure 39*).

Figure 39. Selecting the project configuration



2. Once the building options are correctly configured, configure your project settings:
   a) Select ***Project > Settings*** from the STVD main menu toolbar.
   b) The **Project settings** window opens and displays all the options of your toolset compiler, assembler, and linker. You can then customize these options. For more

information on the available options for your toolset, refers to the STVD online help available by clicking **Help** in the STVD main menu toolbar.

3. Once your MCU is selected (***Project > Settings > MCU Selection***) and your building options are configured, use the **Build** menu to build and rebuild your application, or compile your source files (see *Figure 40*).
4. The build command lines and possible warnings and/or errors are displayed in the **build** tab of the **Output** window (see *Figure 41*).

Figure 40. Building your project

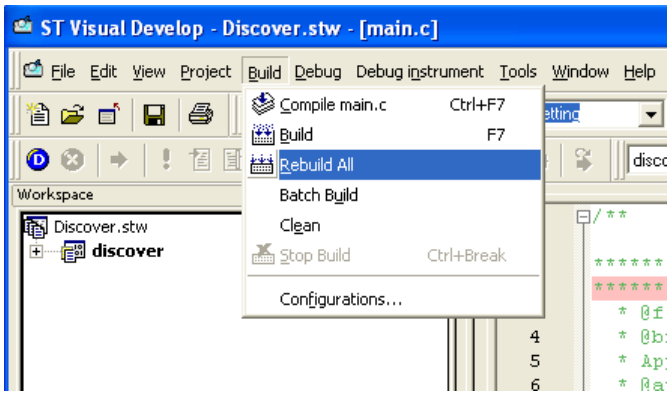
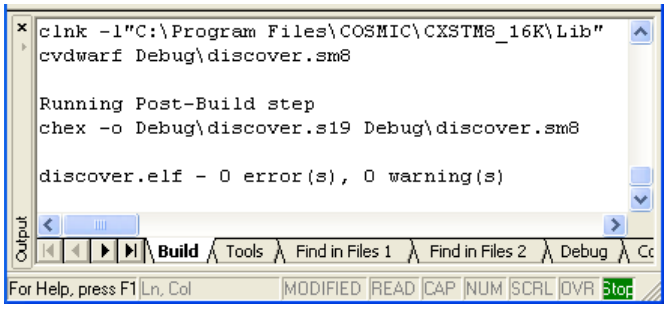


Figure 41. Building successful message



6.2 Debugging your application

Once your application has been successfully built, access the **Debug context** from the STVD main menu to access the debugging features that are supported by your debugging instrument. The debug instrument must be selected before starting debugging.

1. Select ***Debug Instrument > Target settings*** from the STVD main menu toolbar.
2. Select **Swim ST-Link**, which is your debug instrument (see *Figure 42*).
3. Click **Apply** to confirm and **OK** to close the window.
4. Select ***Debug > Start Debugging*** to start your debug session and access the debug context. STVD then connects to your debug instrument, loads the code into the microcontroller Flash memory, and provides access to the debug commands and menus. You can now start debugging your application (see *Figure 43*).

Figure 42. Selecting the debug instrument

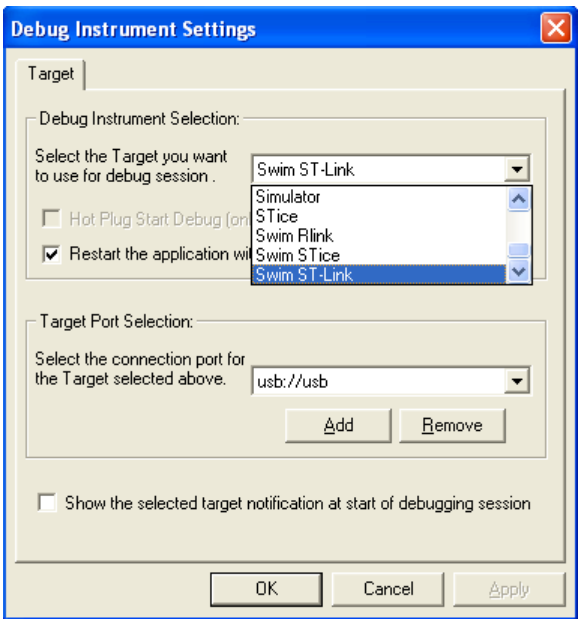
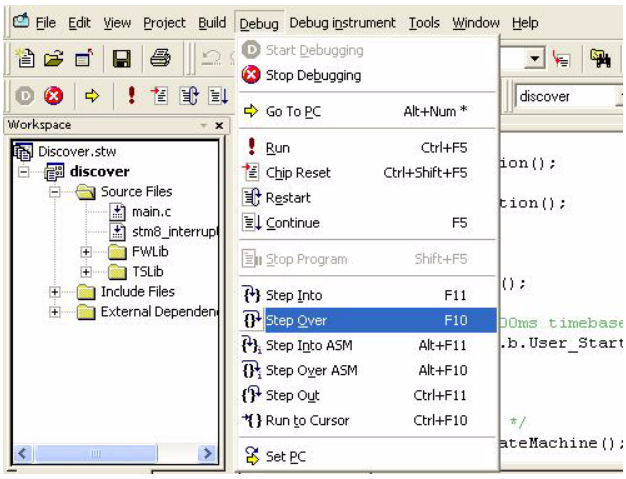


Figure 43. Degugging your application



Several debug windows are available in STVD. You can access them by selecting **View menu** when a debug session is ongoing.

One of the most useful windows is the **Peripheral registers window.** It can be used to display the content of the STM8S105C6 peripheral registers during the debug session. (see *Figure 44*).

The **Watch window** displays the values the code variables. Just drag and drop a variable from your code into the **Watch window**.

The information displayed in the debug windows are refreshed when the program is stopped (for example by a break point). (see *Figure 45*)

Figure 44. Peripheral registers window

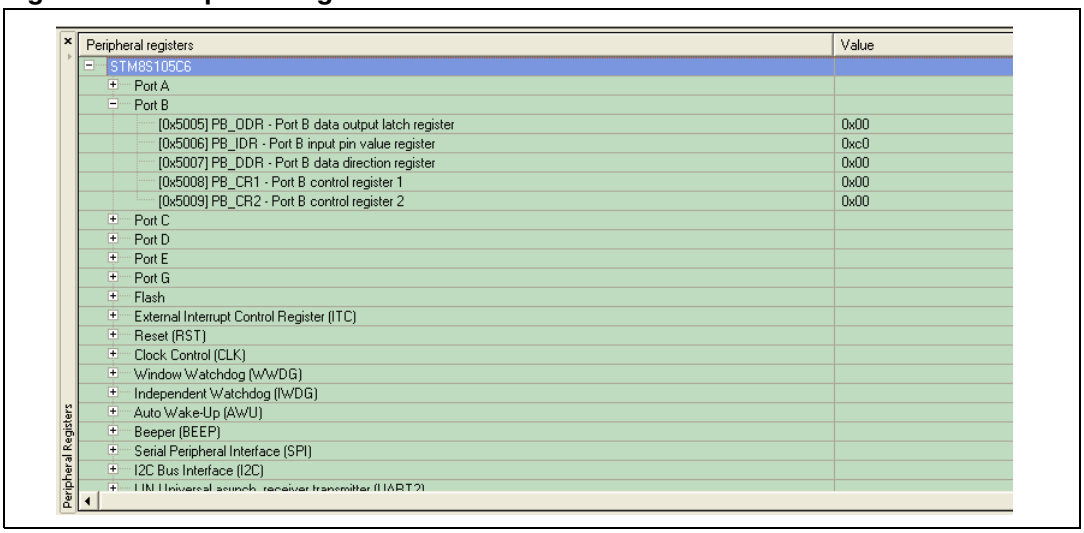
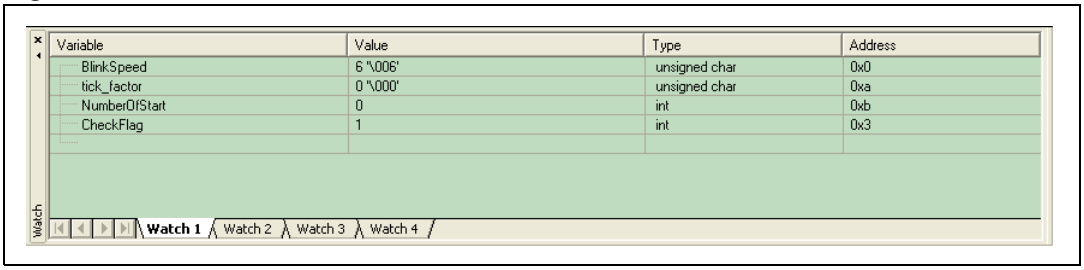


Figure 45. Watch window



6.3 Online help

For more information on building and debugging features, refer to embedded STVD online help page (**Help>Help Home Page)**.

Figure 46. STVD online help

7 Revision history

Table 2. Document revision history

Date	Revision	Changes
04-Nov-2009	1	Initial release.