



# StellarStudio NOC Configurator

---

## StellarStudio NOC Configurator User Guide

---

# Contents

<b>1 Introduction</b>	<b>4</b>
<b>2 Creation of a NoC configuration</b>	<b>6</b>
2.1 Create a project if needed	6
2.2 Create configuration file	8
<b>3 Manipulation of a NoC configuration</b>	<b>12</b>
3.1 Manipulation of the configuration file	12
3.2 Using alternate model files	15
3.3 Code generation	16
3.4 Configuration errors	18
3.5 Outline	19
3.6 NoC configuration formatting	21
<b>4 Integration</b>	<b>24</b>
4.1 Adapt project's Makefile	24
<b>5 Disclaimer</b>	<b>27</b>

## List of figures

Figure 1. StellarStudio NoC configurator - Copyrights STMicroelectronics (c) 2021-2025 .....	4
Figure 2. NoC configuration tool .....	4
Figure 3. Create a project from an empty project .....	6
Figure 4. Select Project creation wizard .....	7
Figure 5. Name the project .....	8
Figure 6. Launch NoC configurator wizard .....	9
Figure 7. Give the configuration a name (with noccfg extension) .....	10
Figure 8. Choose the NoC model you want to work with .....	11
Figure 9. Convert the project to Xtext project .....	11
Figure 10. Choose an initiator .....	12
Figure 11. Initiator has been chosen .....	13
Figure 12. Start initiator configuration .....	13
Figure 13. Start target configuration (first option) .....	14
Figure 14. Start target configuration (first option) - result .....	14
Figure 15. Start target configuration (first option) - result .....	14
Figure 16. Start target configuration (alternate option) .....	15
Figure 17. Start target configuration (alternate option) .....	15
Figure 18. Give the configuration a name (with noccfg extension) .....	16
Figure 19. Code generated from the configuration .....	16
Figure 20. Reset code generation .....	17
Figure 21. Reset code generated .....	18
Figure 22. Syntax Errors .....	19
Figure 23. Semantic errors and quickfixes .....	19
Figure 24. Outline for configuration file .....	20
Figure 25. Outline for generated files .....	21
Figure 26. Example before formatting .....	22
Figure 27. Example after formatting .....	23
Figure 28. Name of the folder to be used .....	24
Figure 29. Adapt C_SRCS and C_INCS .....	25
Figure 30. One function to initialize the Noc .....	25
Figure 31. Call NoC init function from main.c .....	26

# 1 Introduction

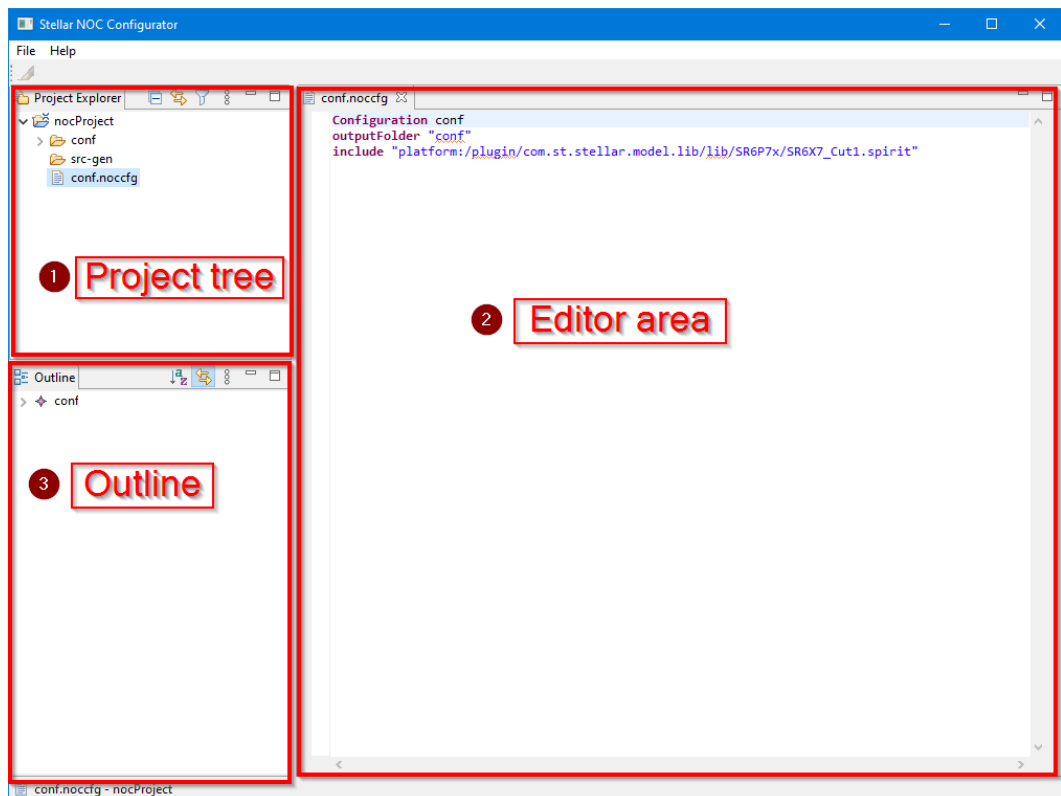
Figure 1: StellarStudio NoC configurator - Copyrights STMicroelectronics (c) 2021-2025



The NoC configurator is an intuitive end user graphical assistant dedicated to configure the Network On Chip of Stellar products.

The configuration is done using a textual edition of the initiators and their targets. For each target in an Initiator group you need to program for example the various QoS generators or Initiator NIU error generation...

Figure 2: NoC configuration tool



After a configuration has been created, the tool looks like in the above picture, where three main areas are present.

1. Project tree: contains the various projects of the workspace and their corresponding files.
2. Editor area: contains the editing part of the tool, where configuration files are manipulated, and where generated C files can be seen.
3. The outline will show the hierarchical structure of the files shown in the editor area.

## 2 Creation of a NoC configuration

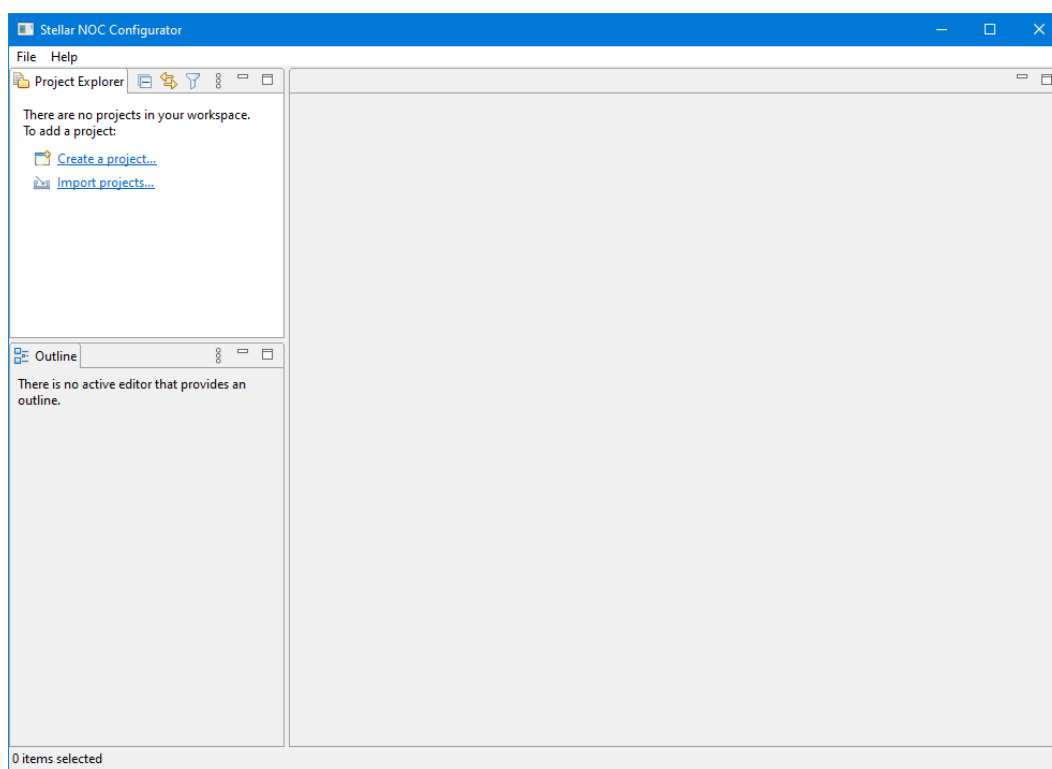
A NoC configuration is represented by a configuration file, with the ".noccfg" extension. This file must be present in a project, which host it and allows for its manipulation.

The following sections describe how to create a project and how to create a configuration file.

### 2.1 Create a project if needed

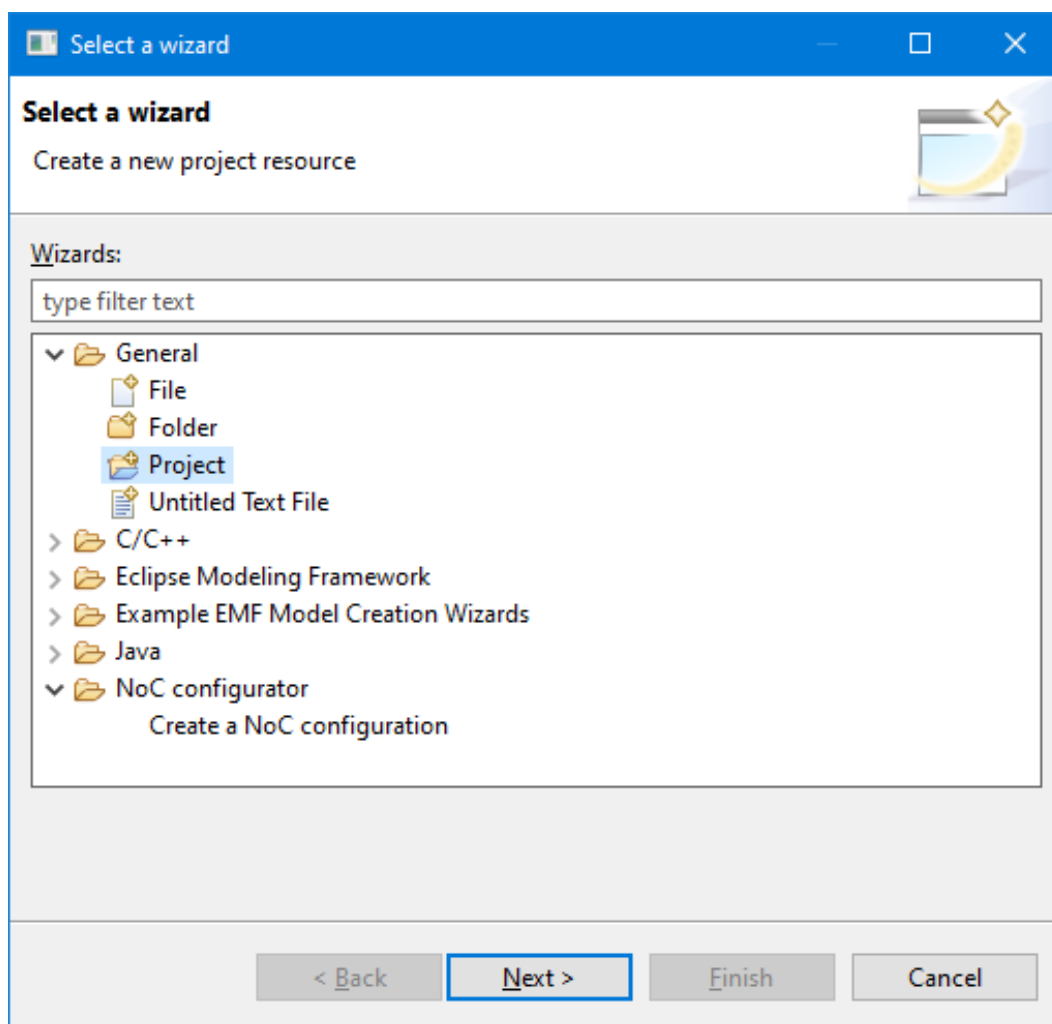
The NoC configuration is hosted in a file, that needs a project host.

**Figure 3: Create a project from an empty project**



If you do not have yet a project, you might use the project creation wizard, using the <CTRL>N shortcut.

Figure 4: Select Project creation wizard



Then click "Next" and then enter the name you want to give to your project, then click "Finish":

Figure 5: Name the project

**New Project**

**Project**  
Create a new project resource.

Project name:

☒ Use default location

Location:

**Working sets**

☐ Add project to working sets

Working sets:

## 2.2 Create configuration file

As soon as you have a project to host your configuration, you create the configuration file.

This is done using the same key combination: <CTRL>N, then select the NoC configurator creation wizard.



Figure 6: Launch NoC configurator wizard

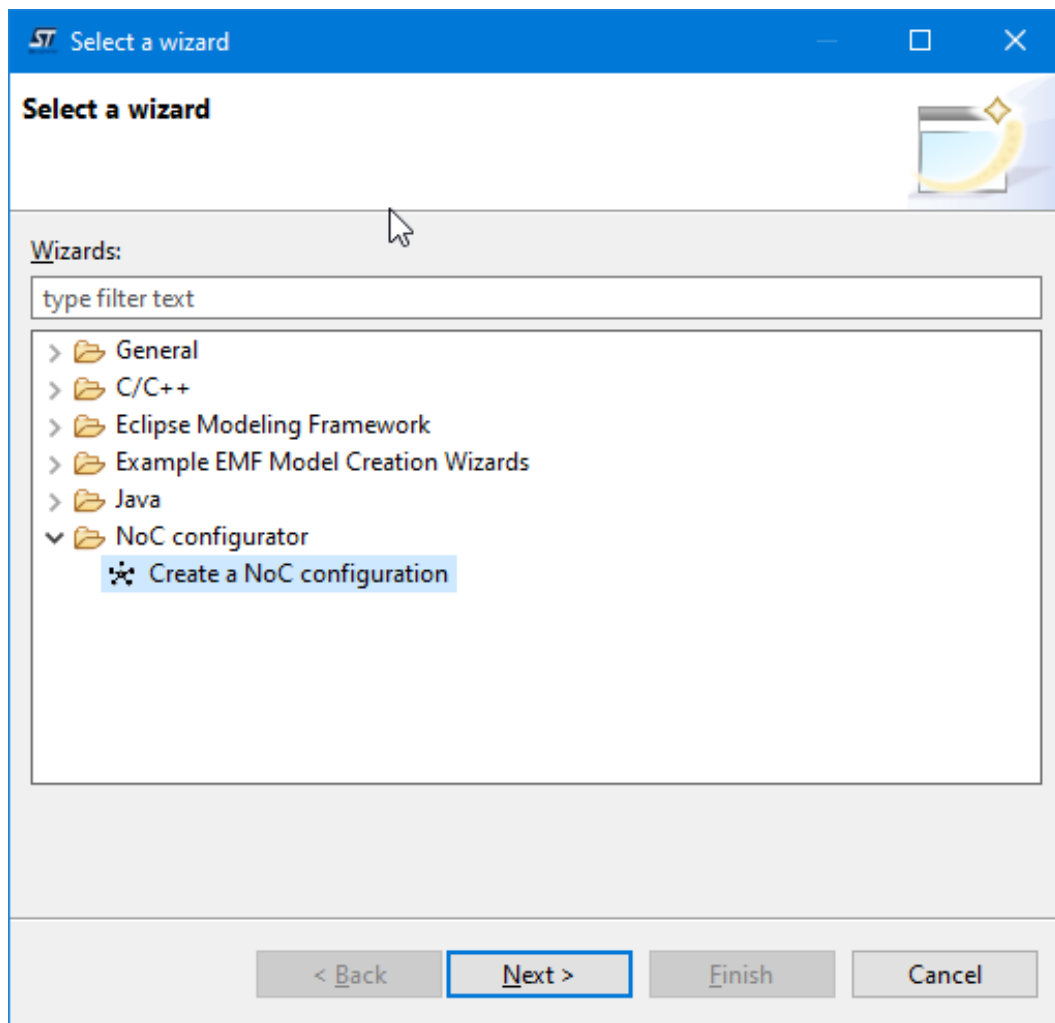
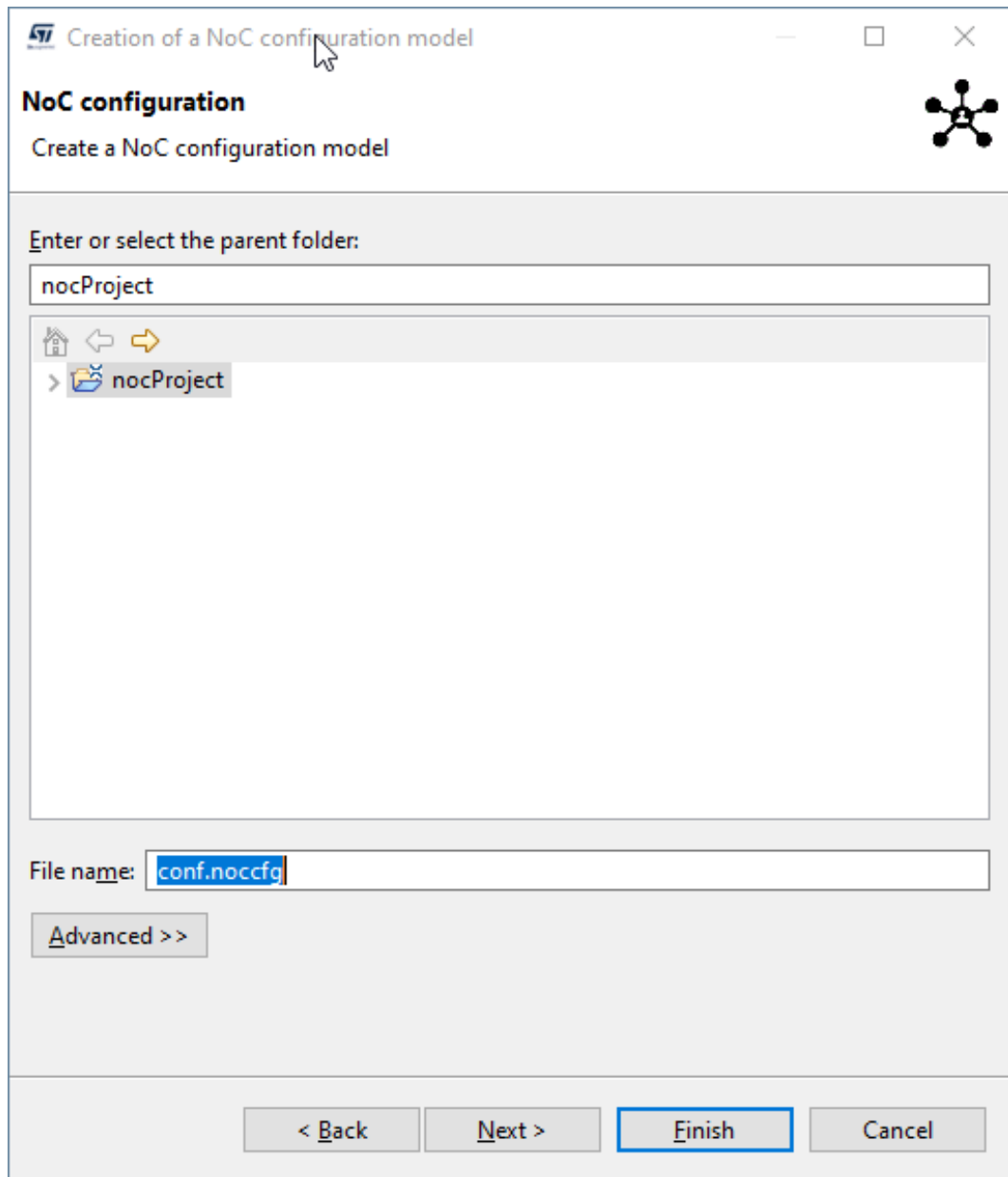


Figure 7: Give the configuration a name (with noccfg extension)



Creation of a NoC configuration model

**NoC configuration**

Create a NoC configuration model

Enter or select the parent folder:

nocProject

> nocProject

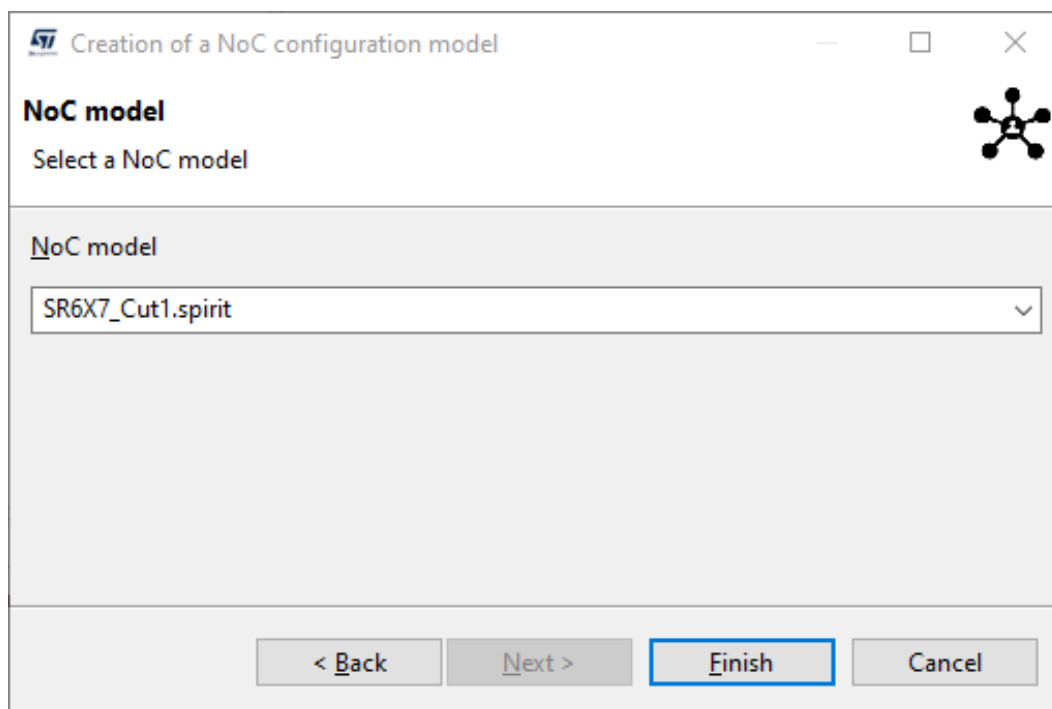
File name: conf.noccfg

Advanced >>

< Back Next > Finish Cancel

Give it a name, then click on "Next"

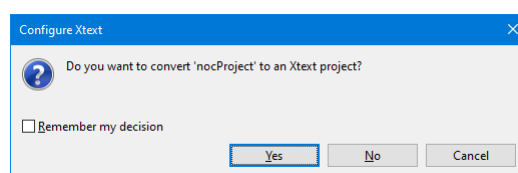
Please note that the file name must have the extension ".noccfg"

**Figure 8: Choose the NoC model you want to work with**

Choose the NoC model you want to work with. This file is provided by the hardware design team and contains the needed NoC definitions that will be used for a particular Stellar platform.

Now click on "Finish", one dialog window should appear, asking you to convert your project into a Xtext project. Select Yes.

This step is mandatory, as the configuration file is based on the Xtext technology, declaring the project as an Xtext project allows for launching the appropriate editor when double-clicking on the configuration file.

**Figure 9: Convert the project to Xtext project**

Now your project is ready and is opened in the editor area of the tool.

## 3 Manipulation of a NoC configuration

Now that you have created a NoC configuration file, the following section describe how to manipulate a NoC configuration.

### 3.1 Manipulation of the configuration file

The configuration needs a name, an output folder, a NoC model and a collection of Initiators and their corresponding configurations.

The "Configuration" keyword precedes the name to be given to the configuration.

The output folder, following the "outputFolder" keyword, is the project relative path to the folder where the C code corresponding to the configuration will be generated.

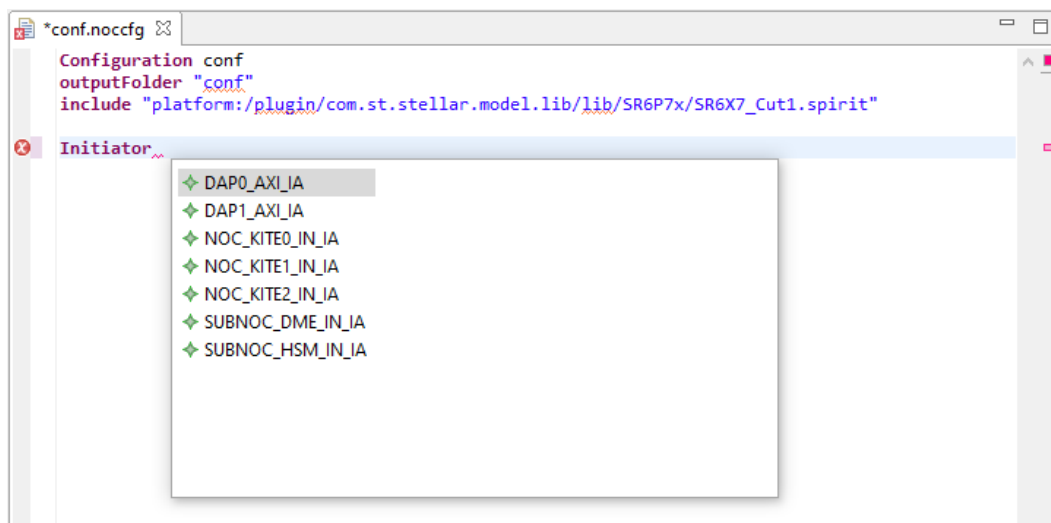
The "include" keyword introduces the URI to the IP-XACT file containing the definitions of the needed NoC registers, address blocks, initiators... This file is provided by the "hardware design" team.

Now you can edit the file, adding the initiators and their corresponding configurations.

**NB: remember that at any time you can use the key combination <CTRL><Space bar>, and the editor will help you with a contextual content assist helper menu.**

You can add one Initiator using the "Initiator" keyword, and type <CTRL><Space bar> to choose amongst available initiators :

Figure 10: Choose an initiator



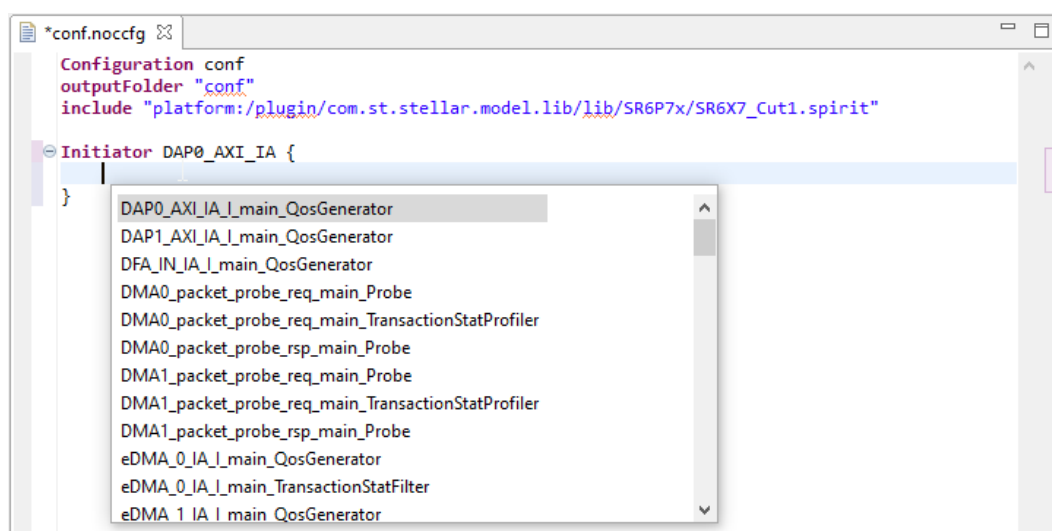
Now choose an initiator and type <ENTER>.

Figure 11: Initiator has been chosen



then type { end <ENTER> to specify the start of the configuration of the initiator.

Figure 12: Start initiator configuration

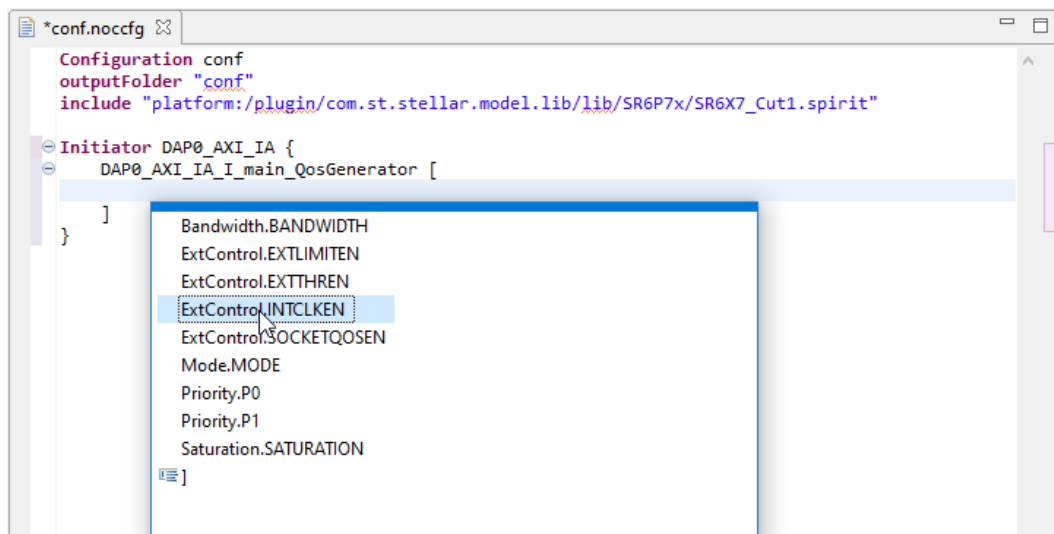


Now you need to add some configurations, here again, <CTRL><Space bar> will propose you the available possibilities. Chose one of them, its name will be added. Now enter [ then <ENTER> so that you start configuration of the target, the configuration will look like:

Here you have two choices, either enter the target register that you want to configure (again using the content assist <CTRL><Space bar> as usual), or ask the tool to add all registers with their default values, right clicking where the cursor is and choosing the "Add default register fields" contextual menu option.

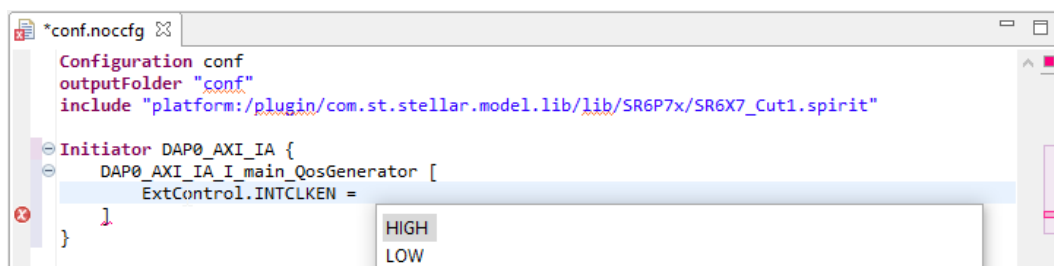
Here are the two options:

Figure 13: Start target configuration (first option)



which will result in the following configuration:

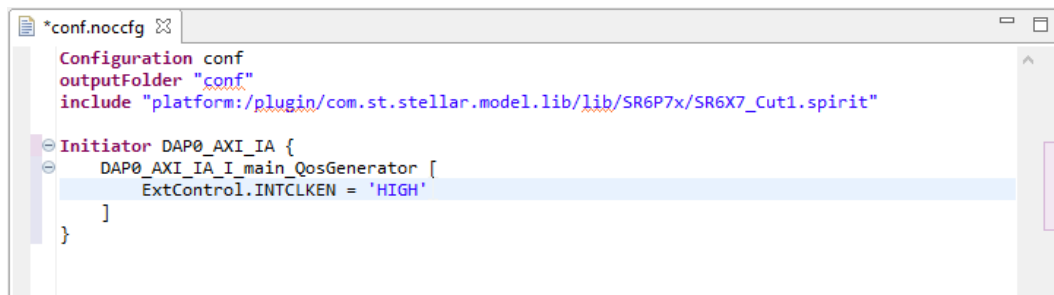
Figure 14: Start target configuration (first option) - result



for the chosen register you can assign a value between available options when possible), here only HIGH and LOW values are possible.

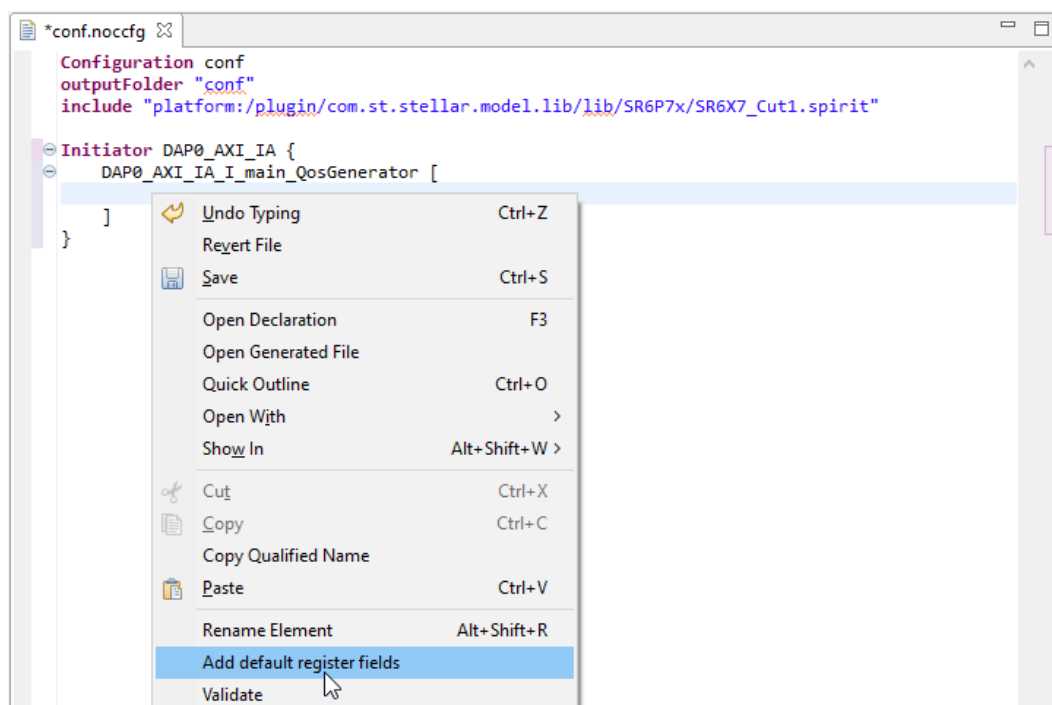
The result will be:

Figure 15: Start target configuration (first option) - result



or

Figure 16: Start target configuration (alternate option)



here the result will be:

Figure 17: Start target configuration (alternate option)



You can reproduce this procedure for all the needed configurations.

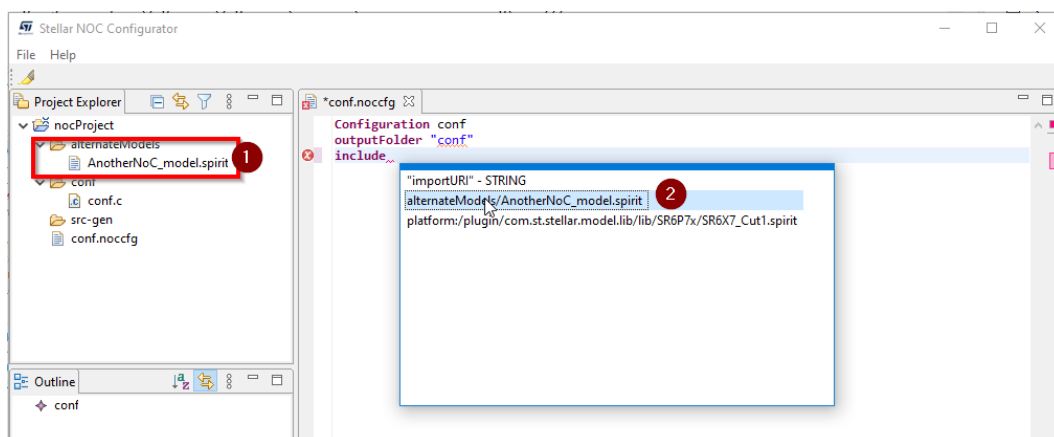
## 3.2 Using alternate model files

NB: The tool contains some predefined NoC models, but you can add other models inside your project as well.

The various models that you can use as an input will be accessible through the <CTRL><Space bar>, after the "include" keyword.

In the following example, you have added a folder with an alternate NoC model file (1). It will be presented to you when choosing the include file to be used in the editor:

**Figure 18: Give the configuration a name (with noccfg extension)**



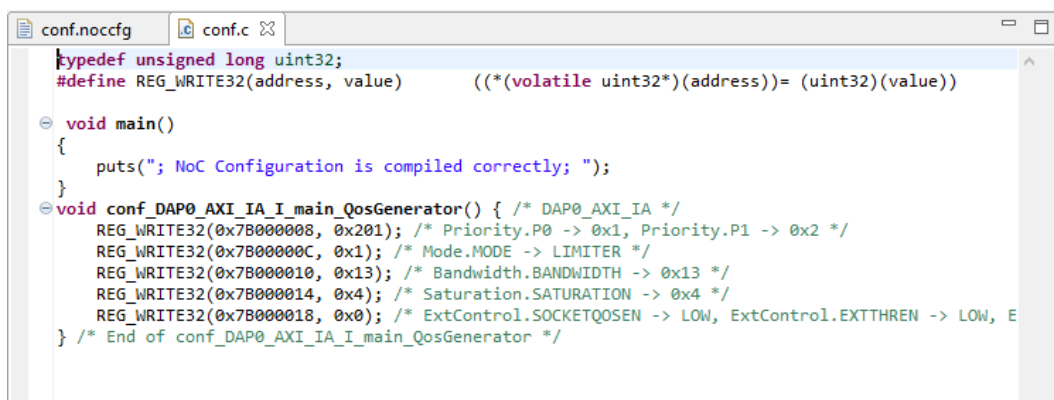
In the following example, you have added a folder with an alternate NoC model file. It will be presented to you when choosing the include file to be used in the editor:

### 3.3 Code generation

As soon as you have a configuration without any error (i.e. without any red markers), the tool will generate the C code corresponding to the current configuration, in the folder identified by the "outputFolder" keyword.

This code generation is automatic, and is done each time the configuration file is saved.

**Figure 19: Code generated from the configuration**



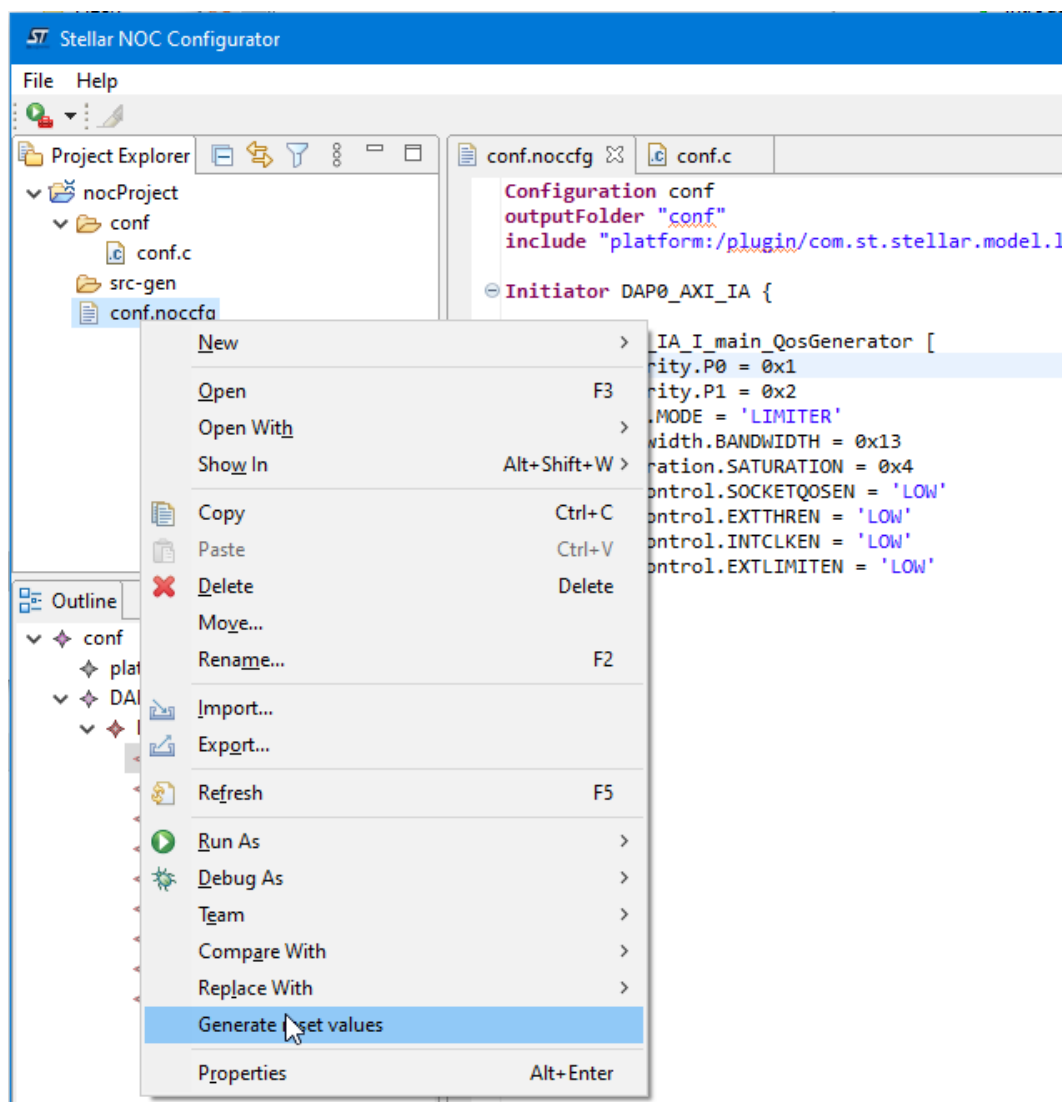
The registers are generated with a comment indicating an explanation of the corresponding values, instead of just the hexadecimal values.



On the contrary, the reset code, i.e. the code corresponding to the POR state of all registers will be generated only on demand, in the "src-gen" folder of the project.

Generation of the reset code is accessible (when a "noccfg" file is selected) through the project explorer contextual menu, with the option "Generate reset values".

Figure 20: Reset code generation



The resulting code looks like following picture:

Figure 21: Reset code generated

```

conf.noccfg  conf.c  resetValues.c
typedef unsigned long uint32;
#define REG_WRITE32(address, value) ((*volatile uint32*)(address))= (uint32)(value))

void main()
{
    puts("; NoC Configuration is compiled correctly; ");
}

void reset_QOS_GENERATOR(){
    /*DAP0_AXI_IA_I_main_QosGenerator*/
    REG_WRITE32(0x7B000008, 0x201); /* Priority.P0 -> 0x1, Priority.P1 -> 0x2 */
    REG_WRITE32(0x7B00000C, 0x1); /* Mode.MODE -> LIMITER */
    REG_WRITE32(0x7B000010, 0x13); /* Bandwidth.BANDWIDTH -> 0x13 */
    REG_WRITE32(0x7B000014, 0x4); /* Saturation.SATURATION -> 0x4 */
    REG_WRITE32(0x7B000018, 0x0); /* ExtControl.SOCKETQSEN -> LOW, ExtControl.EXTTHREN -> LOW, E

    /*DAP1_AXI_IA_I_main_QosGenerator*/
    REG_WRITE32(0x7B000008, 0x201); /* Priority.P0 -> 0x1, Priority.P1 -> 0x2 */
    REG_WRITE32(0x7B00000C, 0x1); /* Mode.MODE -> LIMITER */
    REG_WRITE32(0x7B000009, 0x13); /* Bandwidth.BANDWIDTH -> 0x13 */
    REG_WRITE32(0x7B000094, 0x4); /* Saturation.SATURATION -> 0x4 */
    REG_WRITE32(0x7B000098, 0x0); /* ExtControl.SOCKETQSEN -> LOW, ExtControl.EXTTHREN -> LOW, E

    /*DFA_IN_IA_I_main_QosGenerator*/
    REG_WRITE32(0x7B000108, 0x201); /* Priority.P0 -> 0x1, Priority.P1 -> 0x2 */
    REG_WRITE32(0x7B00010C, 0x1); /* Mode.MODE -> LIMITER */
    REG_WRITE32(0x7B000110, 0x10); /* Bandwidth.BANDWIDTH -> 0x10 */
    REG_WRITE32(0x7B000114, 0x4); /* Saturation.SATURATION -> 0x4 */
    REG_WRITE32(0x7B000118, 0x0); /* ExtControl.SOCKETQSEN -> LOW, ExtControl.EXTTHREN -> LOW, E

    /*Ether0_I_main_QosGenerator*/
    REG_WRITE32(0x7B000188, 0x400); /* Priority.P0 -> 0x0, Priority.P1 -> 0x4 */
    REG_WRITE32(0x7B00018C, 0x3); /* Mode.MODE -> REGULATOR */
    REG_WRITE32(0x7B000190, 0x173); /* Bandwidth.BANDWIDTH -> 0x173 */
    REG_WRITE32(0x7B000194, 0x4); /* Saturation.SATURATION -> 0x4 */
    REG_WRITE32(0x7B000198, 0x0); /* ExtControl.SOCKETQSEN -> LOW, ExtControl.EXTTHREN -> LOW, E

    /*Ether1_I_main_QosGenerator*/
    REG_WRITE32(0x7B000208, 0x400); /* Priority.P0 -> 0x0, Priority.P1 -> 0x4 */
    REG_WRITE32(0x7B00020C, 0x3); /* Mode.MODE -> REGULATOR */

```

### 3.4 Configuration errors

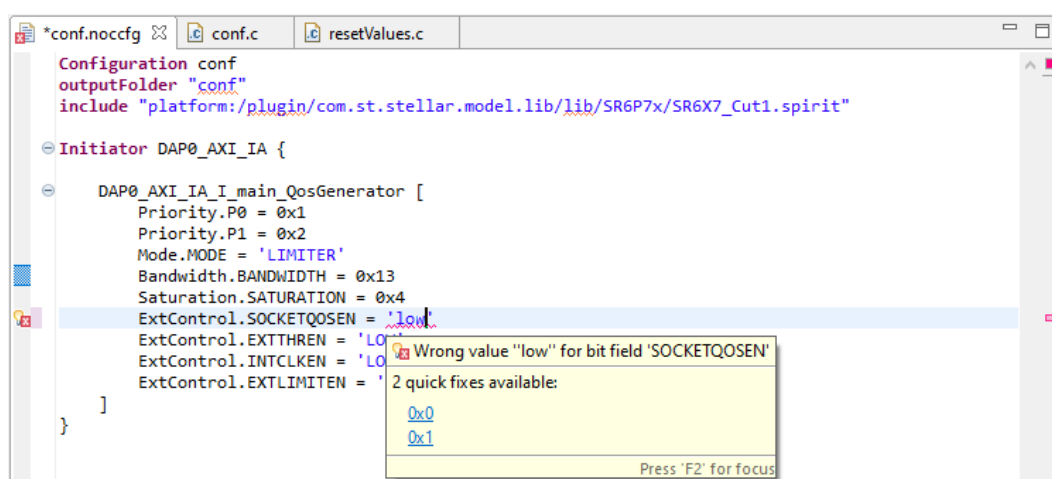
Errors in naming initiators, targets or registers, as well as syntax errors in the expected keywords, are identified with some markers in the configuration editor.

Figure 22: Syntax Errors



For some errors a quick fix is available.

Figure 23: Semantic errors and quickfixes



## 3.5 Outline

The outline view only shows the hierarchical structure of the selected editor, being the configuration editor, or the generated code.

Selecting one item in the outline will make the selected item visible in the corresponding edition area.

Figure 24: Outline for configuration file

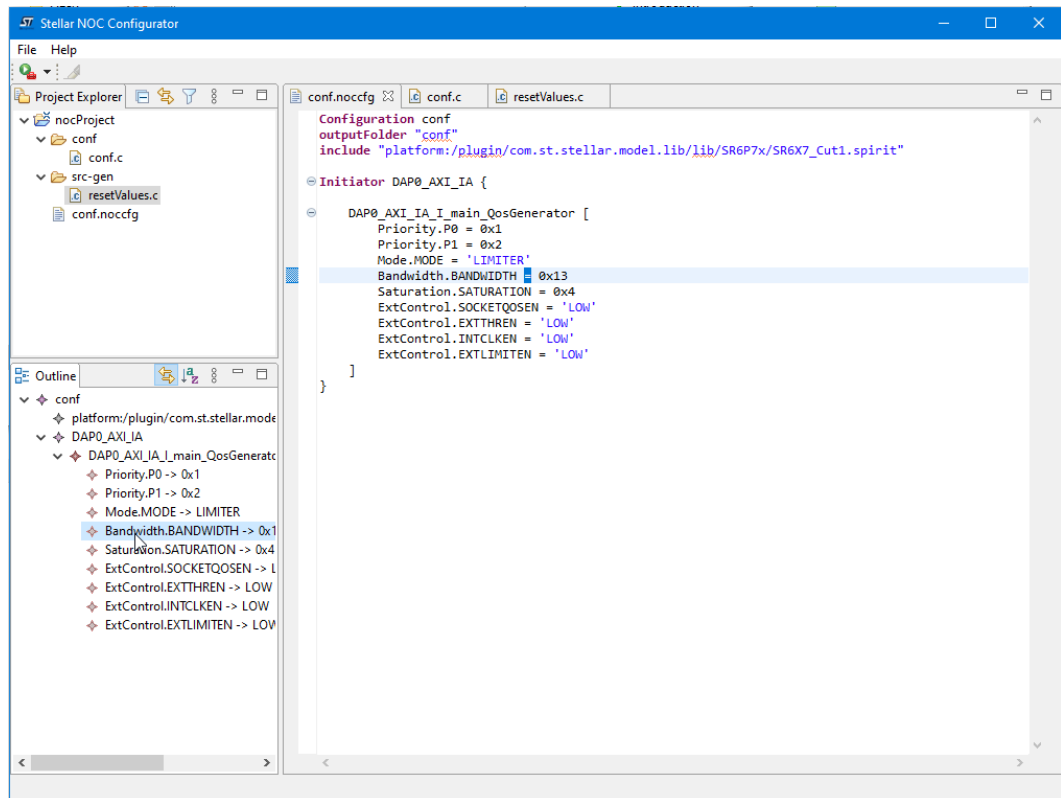
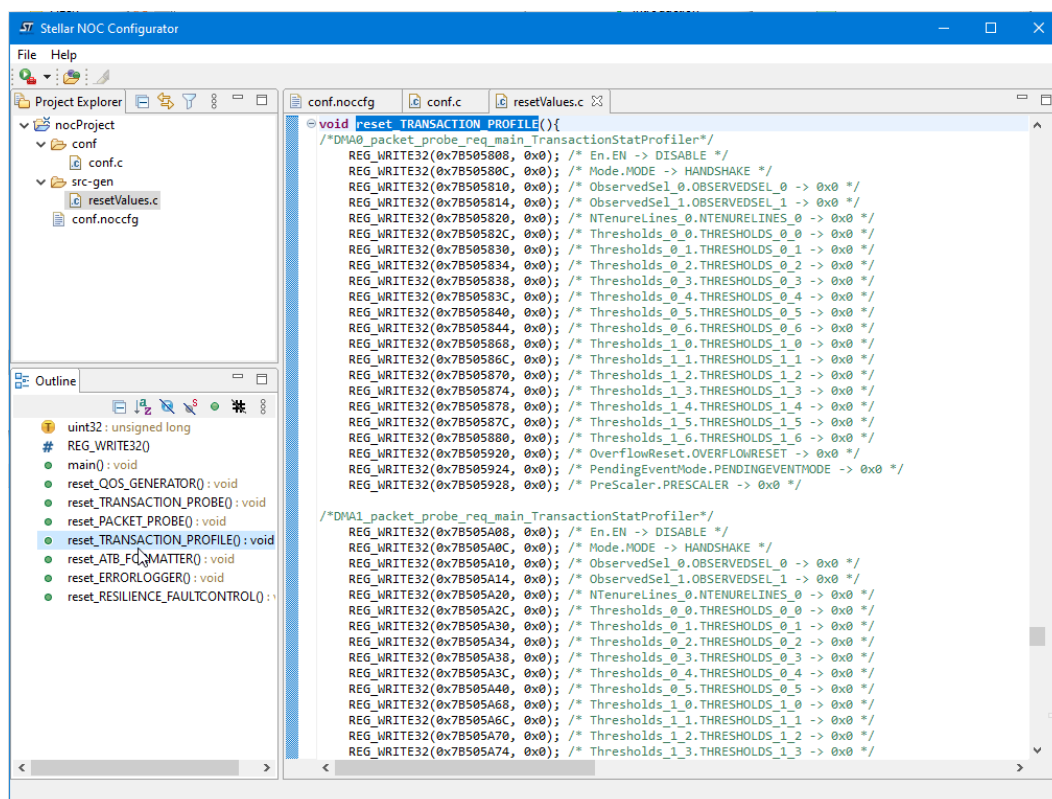


Figure 25: Outline for generated files

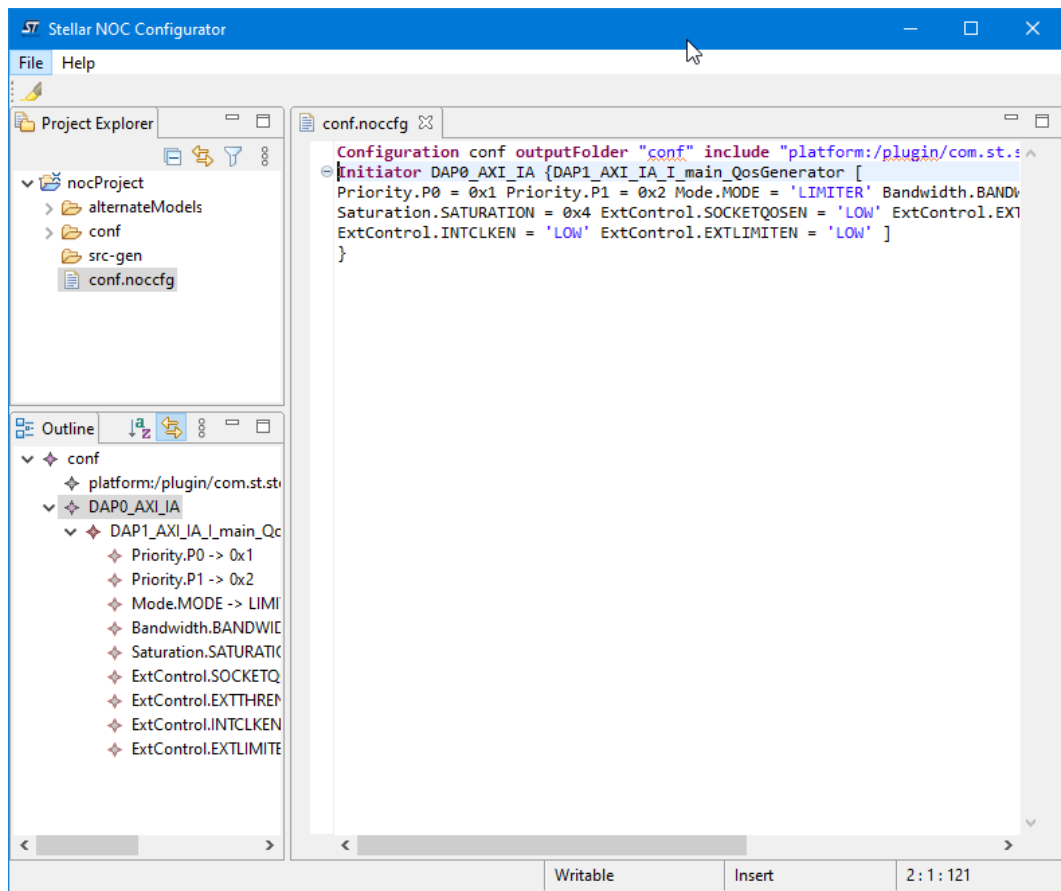


The resulting code looks like following picture:

### 3.6 NoC configuration formatting

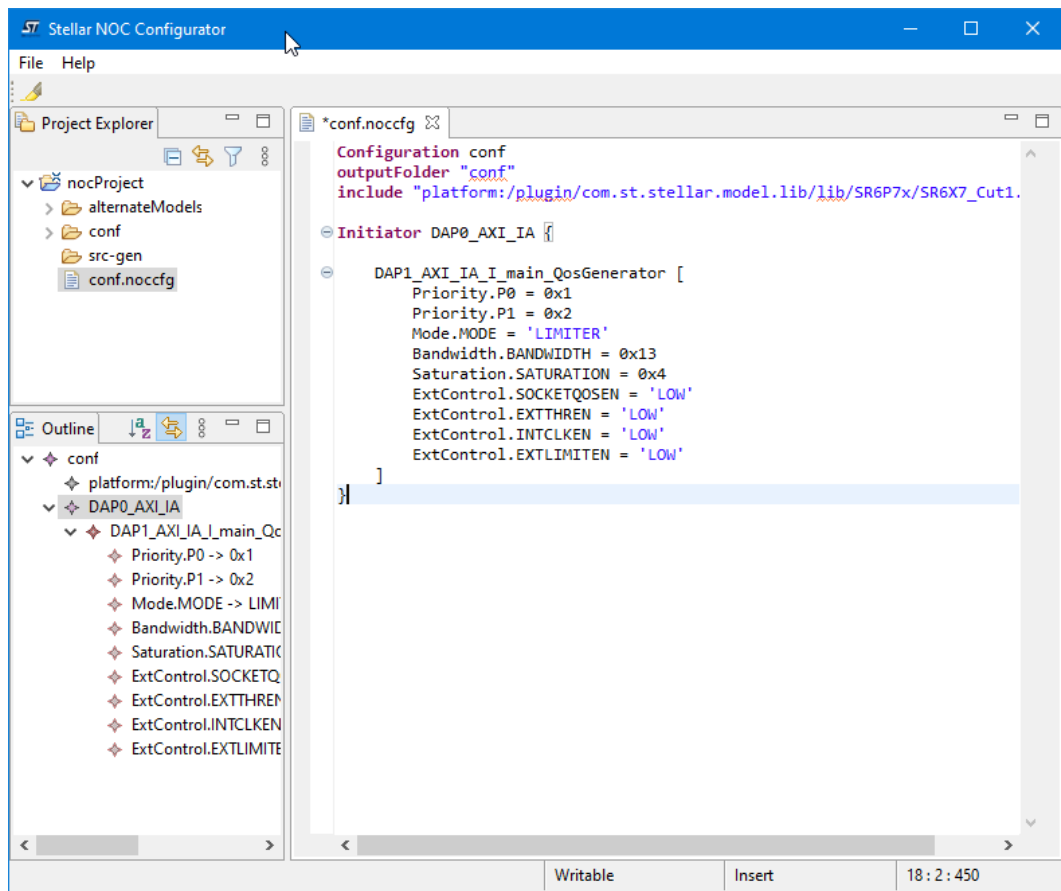
As any other textual editor, you can rearrange its appearance using a default formatting option, accessible with the key combination <CTRL><SHIFT>F

Figure 26: Example before formatting



And here is how it looks like after formatting:

Figure 27: Example after formatting



## 4 Integration

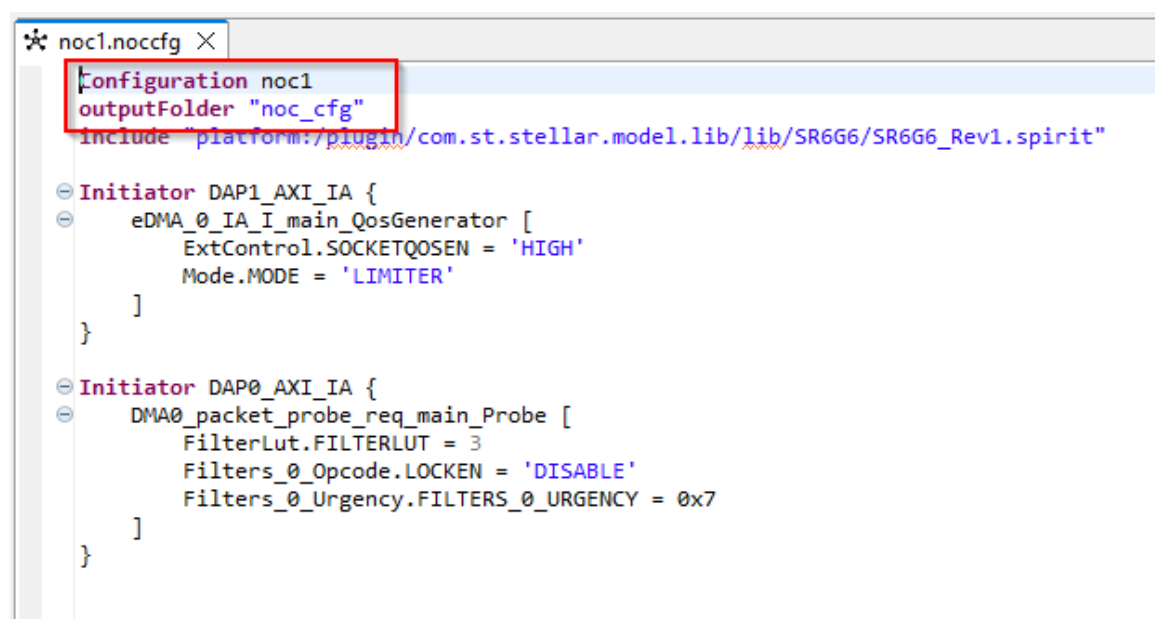
Now that you have manipulated your configuration file, the following section describes how to integrate your generated file in the IDE product.

### 4.1 Adapt project's Makefile

If you want your project to use the NoC configuration that you've created, you'll need to adapt the Makefile to point to the correct generated configuration file, generated from your `.noccfg` file.

The default Makefile uses some source files, listed in the `C_SRC` variable, that you need to adapt, adding the Noc generated file from its output folder.

Figure 28: Name of the folder to be used



Here are the two modifications, illustrated in the window below.



Figure 29: Adapt C\_SRCS and C\_INCS

```
#####
# Project builds
#####
BUILD_OS_OSAL                := 1
BUILD_DRIVERS_DMA            := 1
BUILD_DRIVERS_CAN            := 1

#####
# Add project files
#####

# Application name
APP_NAME := $(PROJECTNAME)

# C sources
C_SRCS += \
    src/main.c \
    noc_cfg/noc1.c \
    cfg/can_cfg.c

# C includes
C_INCS += \
    src-gen/ \
    noc_cfg/ \
    cfg/ \
    src-gen/sr5e1
```

Now that the Makefile is updated, you can compile your project.

Of course you'll need to call the Noc initialization function from the main.c file.

Figure 30: One function to initialize the Noc

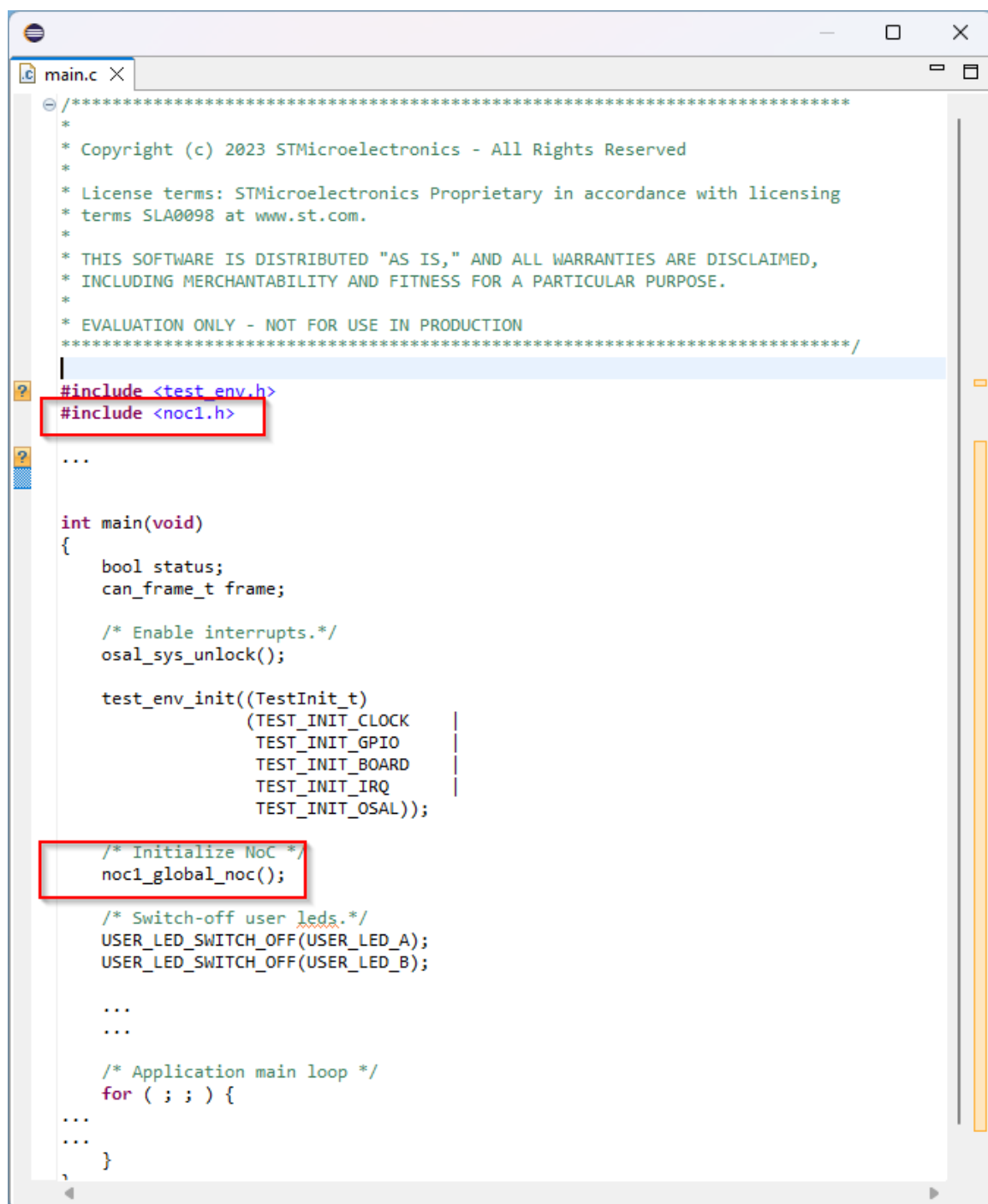
```
typedef unsigned long uint32;
#define REG_WRITE32(address, value) ((*(volatile uint32*)(address))= (uint32)(value))

void noc1_eDMA_0_IA_I_main_QosGenerator() { /* DAP1_AXI_IA */
    REG_WRITE32(0x7B00078C, 0x1); /* Mode.MODE -> LIMITER */
    REG_WRITE32(0x7B000798, 0x1); /* ExtControl.SOCKETQOSEN -> HIGH */
} /* End of noc1_eDMA_0_IA_I_main_QosGenerator */

void noc1_DMA0_packet_probe_req_main_Probe() { /* DAP0_AXI_IA */
    REG_WRITE32(0x7B500014, 0x3); /* FilterLut.FILTERLUT -> 3 */
    REG_WRITE32(0x7B50009C, 0x0); /* Filters_0_Opcode.LOCKEN -> DISABLE */
    REG_WRITE32(0x7B5000A8, 0x7); /* Filters_0_Urgency.FILTERS_0_URGENCY -> 0x7 */
} /* End of noc1_DMA0_packet_probe_req_main_Probe */

/* Global init function for noc1 */
void noc1_global_noc() {
    noc1_eDMA_0_IA_I_main_QosGenerator();
    noc1_DMA0_packet_probe_req_main_Probe();
}
```

Figure 31: Call NoC init function from main.c



```
main.c X
/*****
 * Copyright (c) 2023 STMicroelectronics - All Rights Reserved
 *
 * License terms: STMicroelectronics Proprietary in accordance with licensing
 * terms SLA0098 at www.st.com.
 *
 * THIS SOFTWARE IS DISTRIBUTED "AS IS," AND ALL WARRANTIES ARE DISCLAIMED,
 * INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 * EVALUATION ONLY - NOT FOR USE IN PRODUCTION
 *****/

#include <test_env.h>
#include <noc1.h>

...

int main(void)
{
    bool status;
    can_frame_t frame;

    /* Enable interrupts.*/
    osal_sys_unlock();

    test_env_init((TestInit_t)
                  (TEST_INIT_CLOCK
                   TEST_INIT_GPIO
                   TEST_INIT_BOARD
                   TEST_INIT_IRQ
                   TEST_INIT_OSAL));

    /* Initialize NoC */
    noc1_global_noc();

    /* Switch-off user leds.*/
    USER_LED_SWITCH_OFF(USER_LED_A);
    USER_LED_SWITCH_OFF(USER_LED_B);

    ...
    ...

    /* Application main loop */
    for ( ; ; ) {
    ...
    ...
    }
}
```

## 5 Disclaimer

### Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2014 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany  
- Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

**[www.st.com](http://www.st.com)**