



SR5x StellarStudio Pinmap Configurator

SR5x StellarStudio Pinmap Configurator User Guide

Contents

1 Introduction	5
2 Creation of pins configuration	7
2.1 Create a project if needed	7
2.2 Create board configuration	9
2.3 Create application configuration	11
2.4 Modifying a board's predefined pins	13
3 Manipulation of a pinmap configuration	16
3.1 Magic key	16
3.2 Manipulation of the configuration	16
3.3 Code generation	24
3.4 HTML generation	25
3.5 XLS generation	29
3.6 Configuration errors	32
3.7 Application migration	33
3.8 Outline	35
3.9 Pin configuration formatting	37
4 Graphical configuration of pins	40
4.1 Graphical configuration	40
4.2 Open the graphical configuration editor	40
4.3 Using pinmap configuration outline views	43
4.4 Assigning a function to a pin	46
4.5 Locate graphical pins in textual editor	48
5 Integration	50
5.1 Integration of your generated file (C file)	50
5.2 Integration of your gpio file	51
5.3 Adapt project's Makefile	53
6 Disclaimer	56

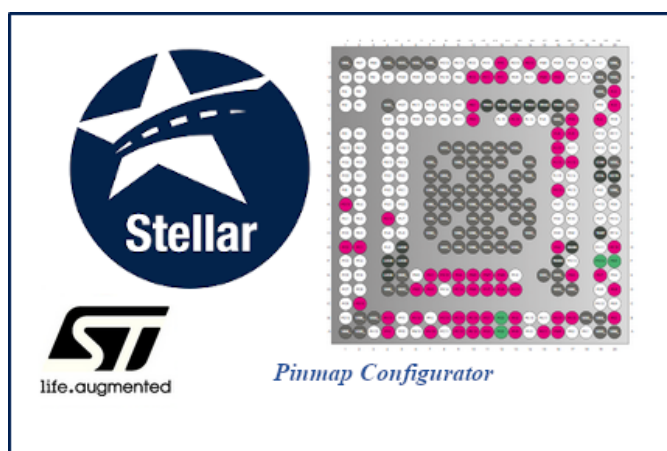
List of figures

Figure 1. StellarStudio Pinmap configurator - Copyright STMicroelectronics (c) 2026	5
Figure 2. Pinmap configuration tool	6
Figure 3. Create a project from an empty project	7
Figure 4. Select Project creation wizard	8
Figure 5. Name the project	9
Figure 6. Launch pinmap board configurator wizard	10
Figure 7. Choose predefined board	10
Figure 8. Start a configuration from a board	12
Figure 9. Copy to current project	14
Figure 10. Changed imported board path	15
Figure 11. Choose a peripheral	17
Figure 12. Choose a direction,	17
Figure 13. ... then a function associated to a pin	18
Figure 14. Pin is chosen	19
Figure 15. Add registers bitfields	20
Figure 16. Resulting configuration	21
Figure 17. Warnings on incomplete pin configurations	21
Figure 18. Add some additional pins	22
Figure 19. Add some comments	23
Figure 20. Comments are added to the generated code	24
Figure 21. Code generated from the configuration	25
Figure 22. Generate the HTML file	26
Figure 23. Open an HTML file	27
Figure 24. HTML generated from the configuration	28
Figure 25. Auto-refresh local changes	29
Figure 26. Generate the Excel file	30
Figure 27. Open an XLS file	31
Figure 28. Example of XLS generated from the configuration	32
Figure 29. Syntax Errors	32
Figure 30. Old applications Migration	33
Figure 31. Launching migration	34
Figure 32. Confirm migration	34
Figure 33. Migration result	35
Figure 34. Unneeded migration	35
Figure 35. Outline for pin configuration	36
Figure 36. Outline for a C generated file	37
Figure 37. Example before formatting	38
Figure 38. Example after formatting	39
Figure 39. Open the graphical editor from project explorer	41
Figure 40. Open the graphical editor from Gpio editor menu	42
Figure 41. Open the graphical editor from the MCU icon in the toolbar	42
Figure 42. Result: opened graphical editor	43
Figure 43. Example of selection a peripheral	44
Figure 44. Example of selection a function	45
Figure 45. Example of searching of a PAD	46
Figure 46. Assign a function to a pin	47
Figure 47. Unassign a function from a pin	47

Figure 48. Show pin in xtext editor	48
Figure 49. Copy your C file	50
Figure 50. Paste your C file in your favorite developer tool	51
Figure 51. Copy your file	52
Figure 52. Paste your <i>gpio</i> file in StellarStudio product	53
Figure 53. Name of the configuration to be used	54
Figure 54. Adapt CONFIG_BOARD and C_INCS	55

1 Introduction

Figure 1: StellarStudio Pinmap configurator - Copyright STMicroelectronics (c) 2026



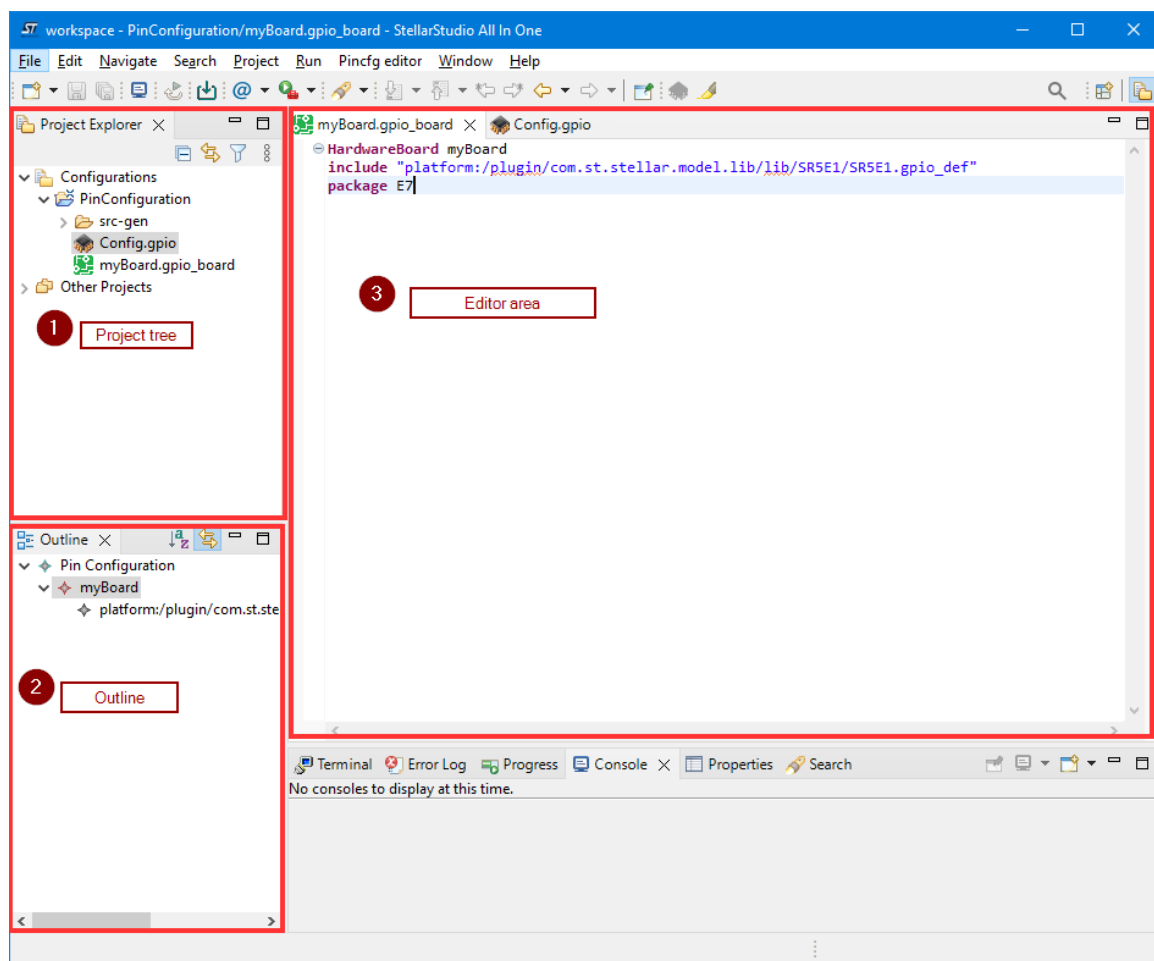
The Pinmap configurator is an intuitive end user graphical assistant dedicated to configure the pinmap of Stellar products.

The configuration is done using a textual edition of the peripherals. For each peripheral, you need to set the associated registers.

An application's pins configuration is based on a board on which the application is supposed to be executed. There is a need to describe the pinmap configuration for this board before being able to add pin configurations dedicated to the application itself.

There are some predefined boards (included inside StellarStudio) that you can start from. Whenever a predefined board needs some modifications, you can import this board into your project's workspace. You can even start from scratch, creating a new board predefined pin configuration from a blank board.

Figure 2: Pinmap configuration tool



When a configuration is 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. The outline will show the hierarchical structure of the *gpio* file shown in the editor area.
3. Editor area: contains the editing part of the tool, where configurations are manipulated, and where generated C, xls and html files can be seen.

2 Creation of pins configuration

As described earlier there are two types of pin configuration : board configuration (identified with the extension *.gpio_board*) and application configuration (identified with the extension *.gpio*).

Both configurations need to be hosted inside an Eclipse project.

Board definitions are created using a dedicated wizard page, accessible using the File/New menu.

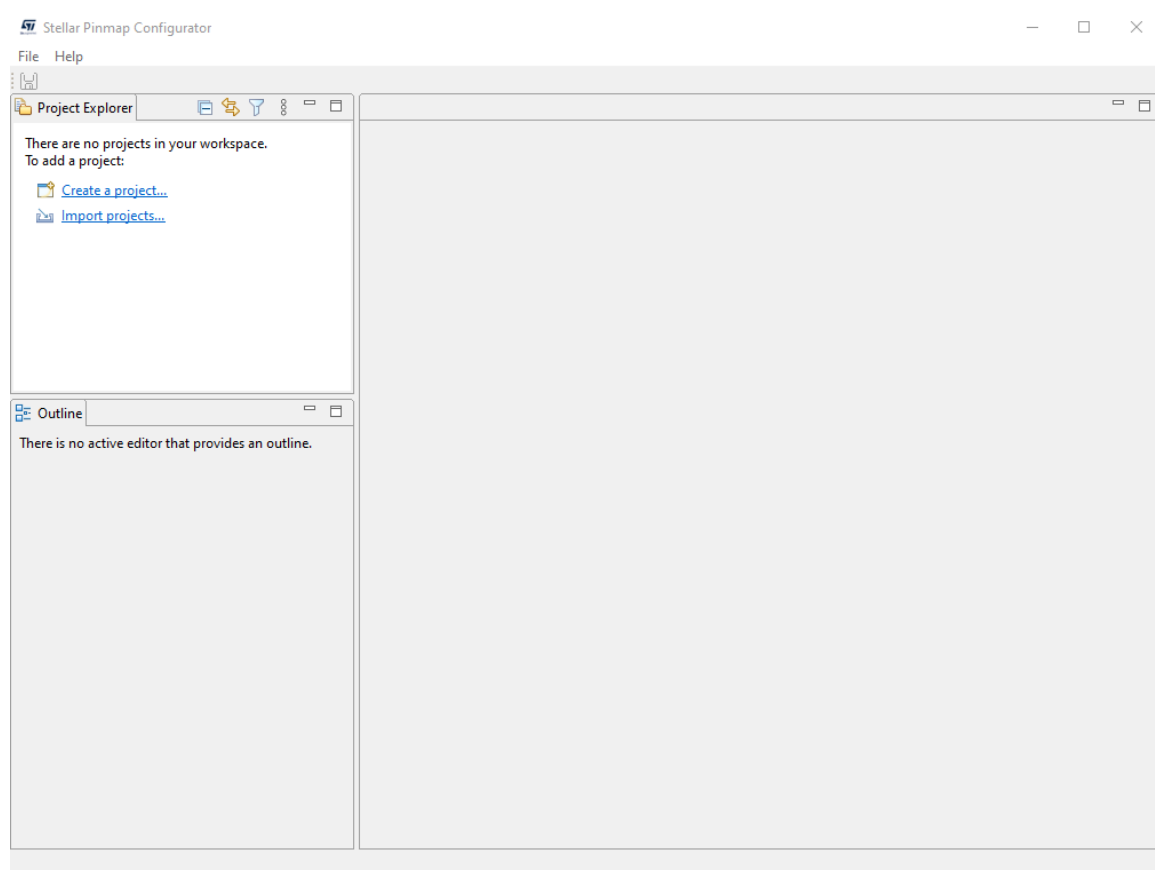
Application configurations are also created using another dedicated wizard page, accessible using the File/New menu as well.

The following sections describe how to create a project and how to create a configuration (either board or application configuration).

2.1 Create a project if needed

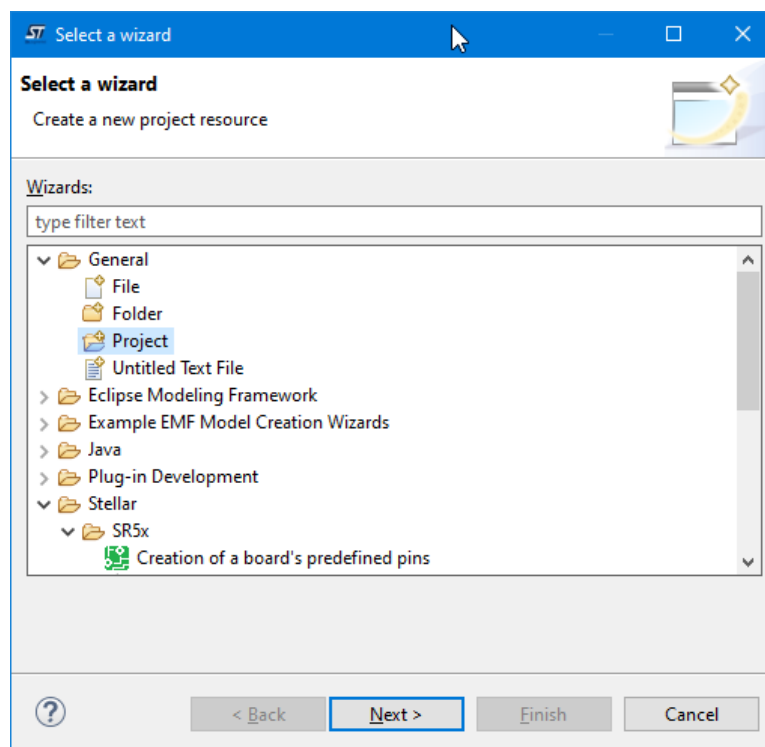
The pin configuration is hosted in a file, that needs a project as container.

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

Project
Create a new project resource.

Project name:

☒ Use default location

Location:

Working sets

☐ Add project to working sets

Working sets:

2.2 Create board configuration

A configuration is based on a board, which contains predefined pins. You can decide to start using a predefined board, or create your own board.

As soon as you have a project you can create a configuration inside.

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

You'll have to choose the wizard named "Creation of a board's predefined pins" in order to create your own board pins definitions. You will be asked to enter a file name (with extension set to `.gpio_board`), then you'll need to choose a MCU pinmap model and a package.

Figure 6: Launch pinmap board configurator wizard

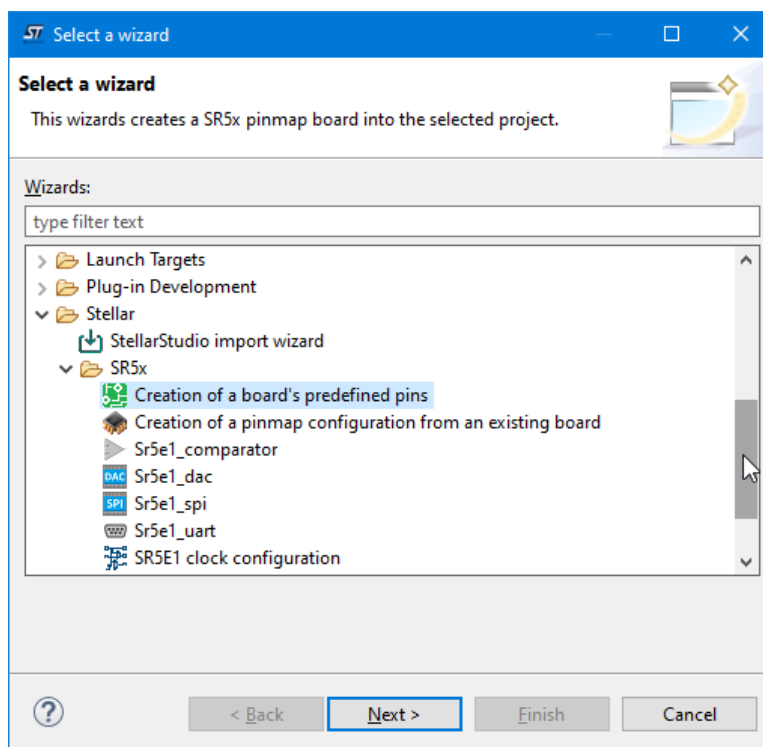
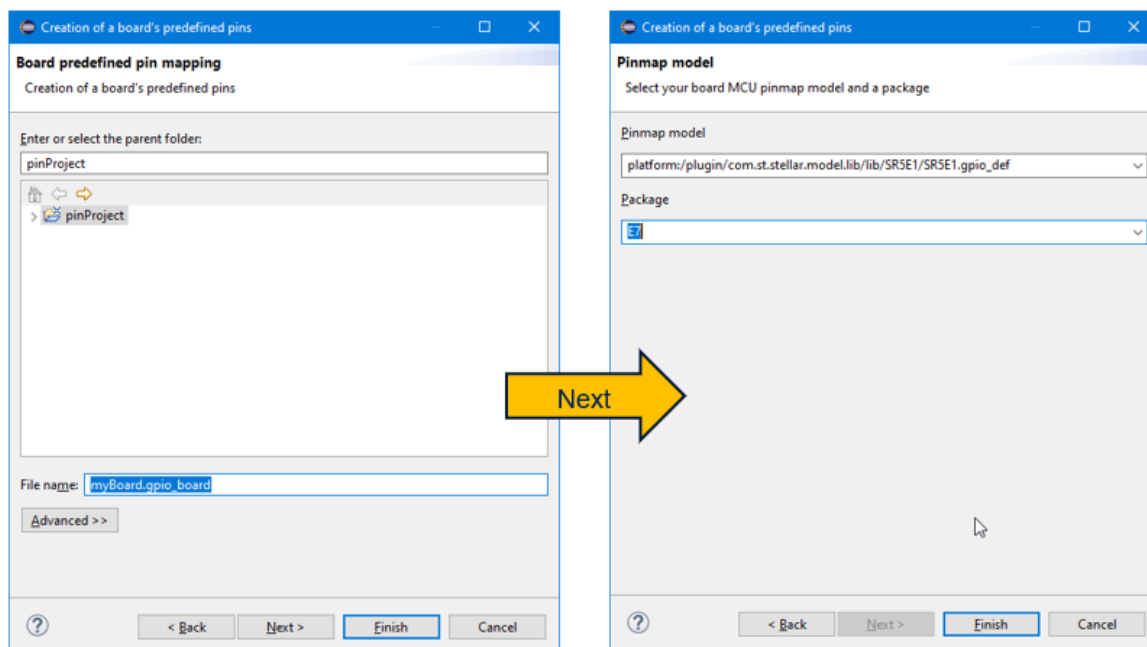


Figure 7: Choose predefined board

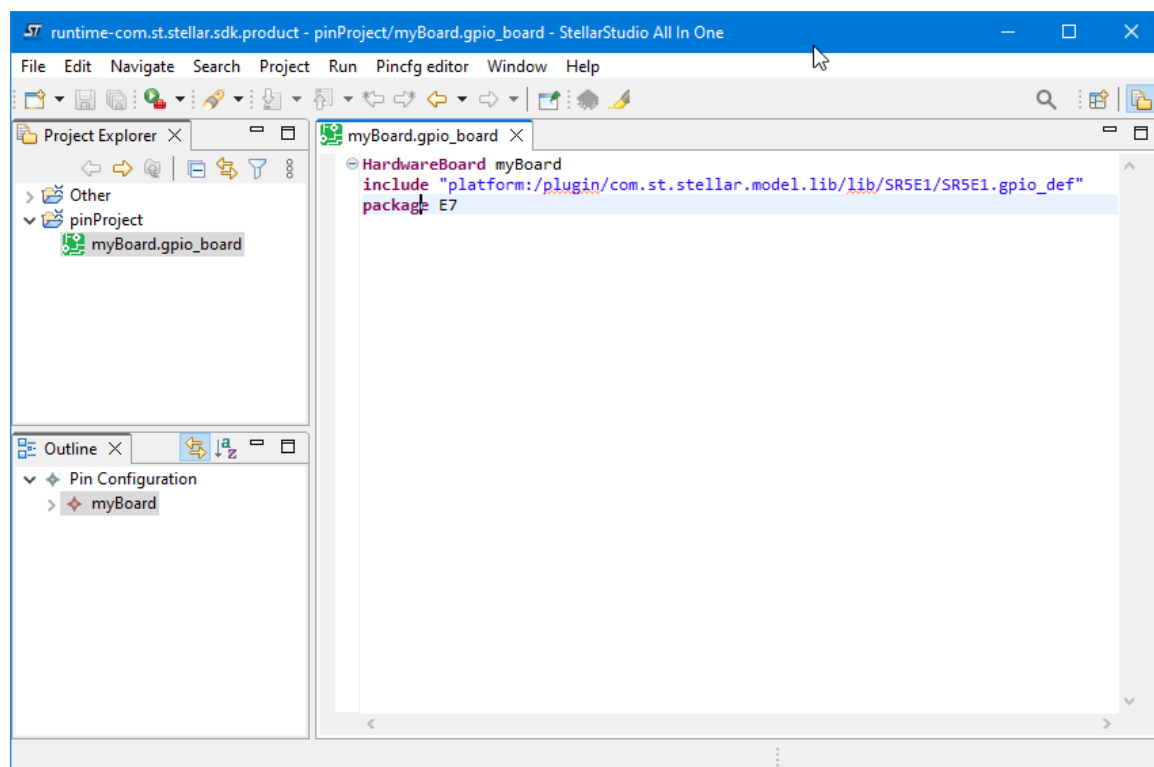


Give it a name, then click on "Next".

Then choose the pinmap model and the package. The pinmap model is provided by the StellarStudio team and contains the needed pinmap definitions that will be used for a particular Stellar platform. This model contains some package definitions. You must choose one of them to start your configuration.

Now click on "Finish".

Now your project is ready and open in the editor area.



Please note that the file name must have the extension ".gpio_board". This file would be part of the available boards you can choose from when creating your pinmap configuration.

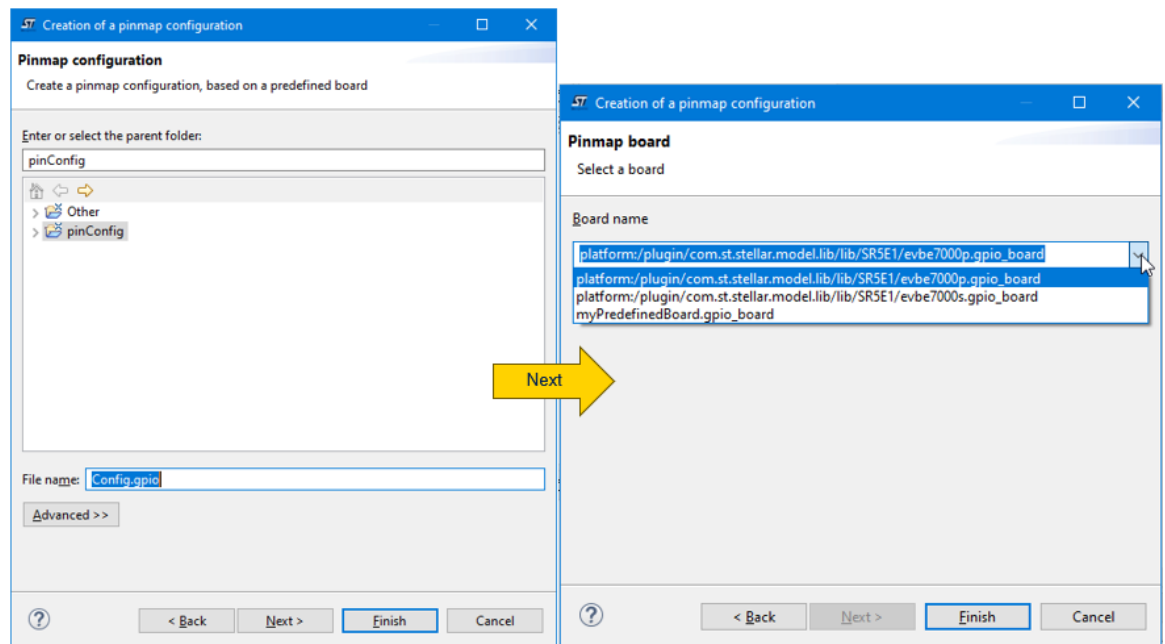
Now you can start adding pin configurations to your board.

2.3 Create application configuration

Application pin configuration are based on the pins already predefined inside a board configuration.

Using the <CTRL>+N, or the File/New menu option, you'll be able to access the "Creation of a pinmap configuration from an existing board" wizard. You will be asked to enter a file name (with extension set to .gpio), then choose the board you want to use, from the list of predefined boards.

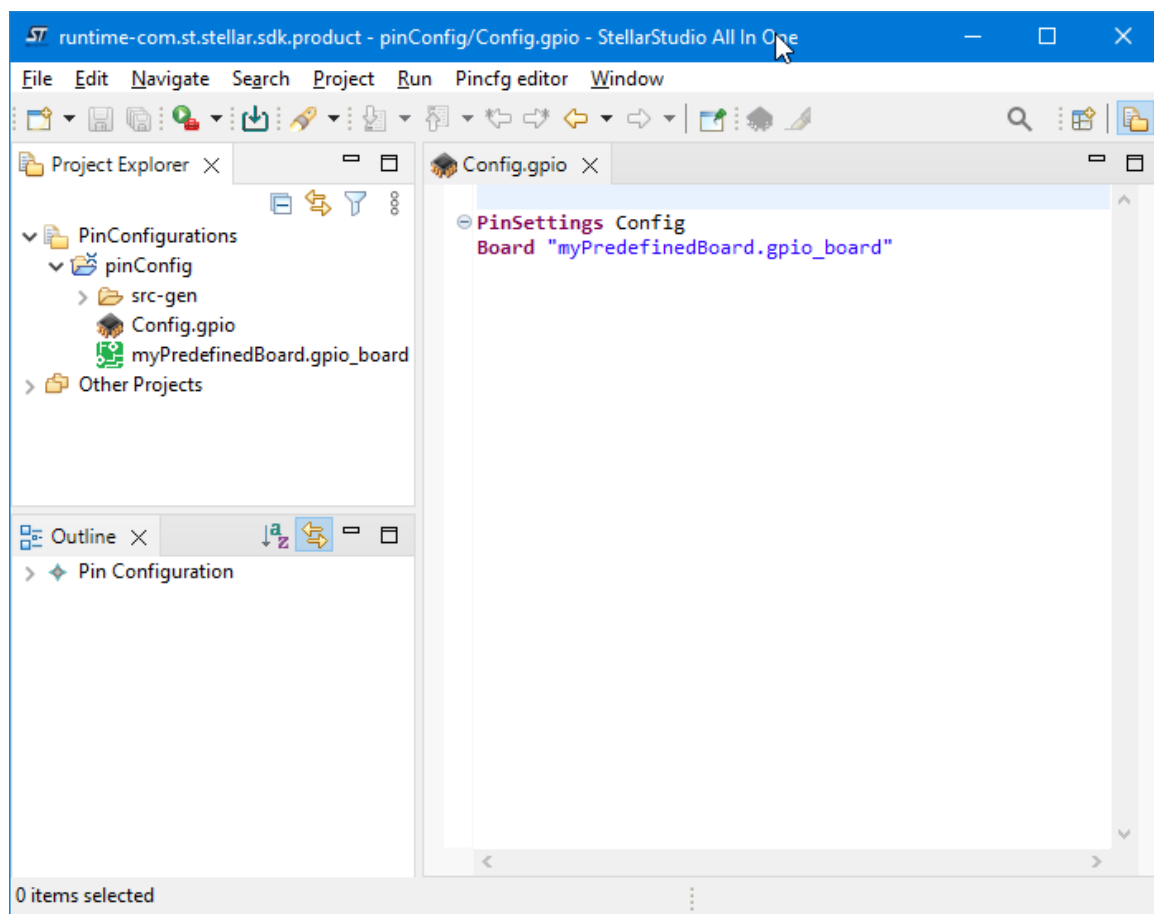
Figure 8: Start a configuration from a board



Choose the board to start from and click on "Finish".

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

Here is the content of the pin configuration, whenever you've chosen the *myPredefinedBoard.gpio_board* file in list of available boards in the previous step.



Now you can start adding pin configurations to your application.

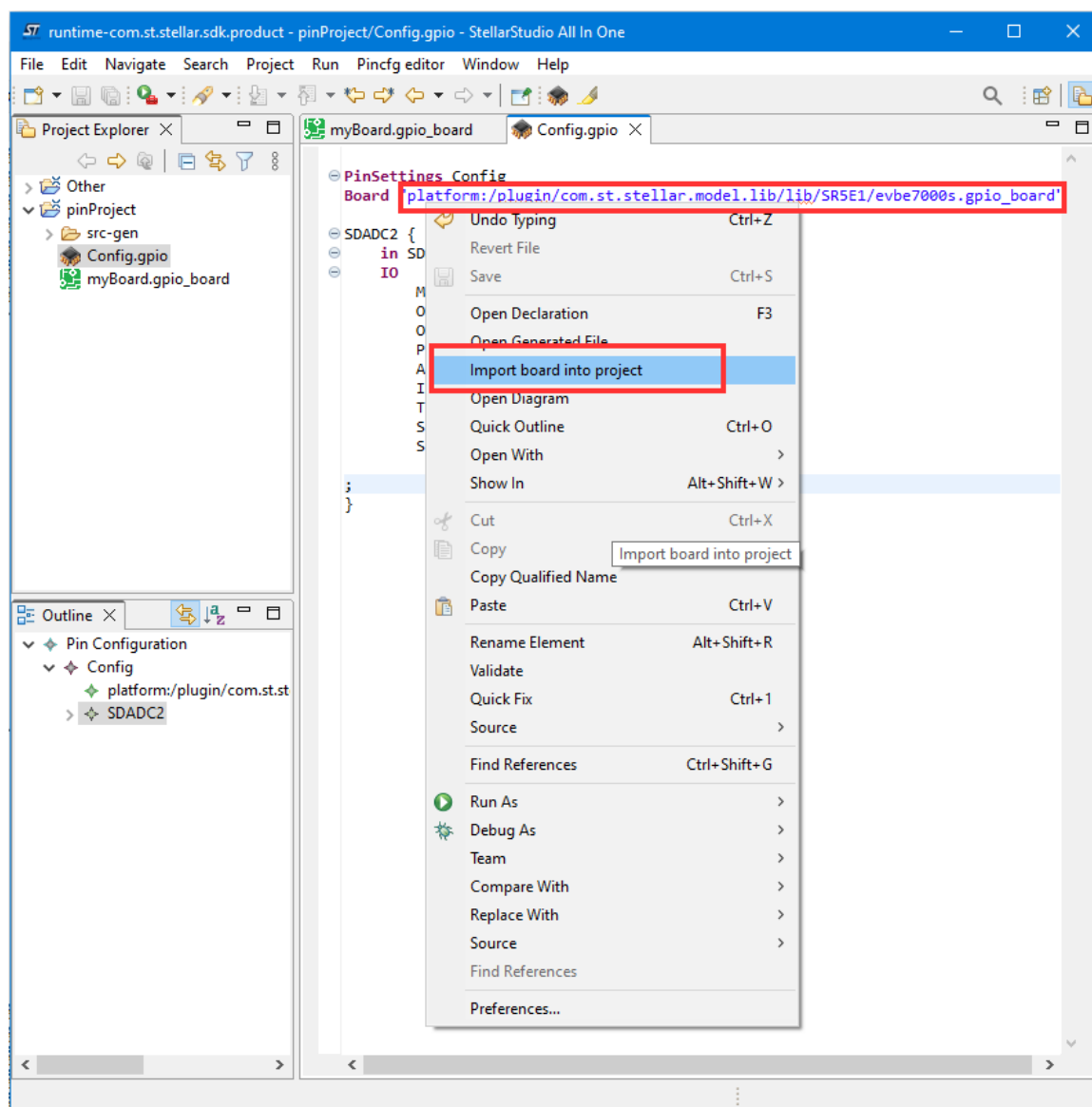
2.4 Modifying a board's predefined pins

The previous section allows you to start a pin configuration from a predefined board. A predefined board, as provided to you by the StellarStudio team, allows to use predefined pins, but contains pin configurations that you cannot modify. Whenever you need to modify or adapt a pin already configured you need to change the predefined definition.

This section describes you how to copy a board definition to your current project.

From an open configuration, you can right click on the editor text and choose the "Import board into project" menu option.

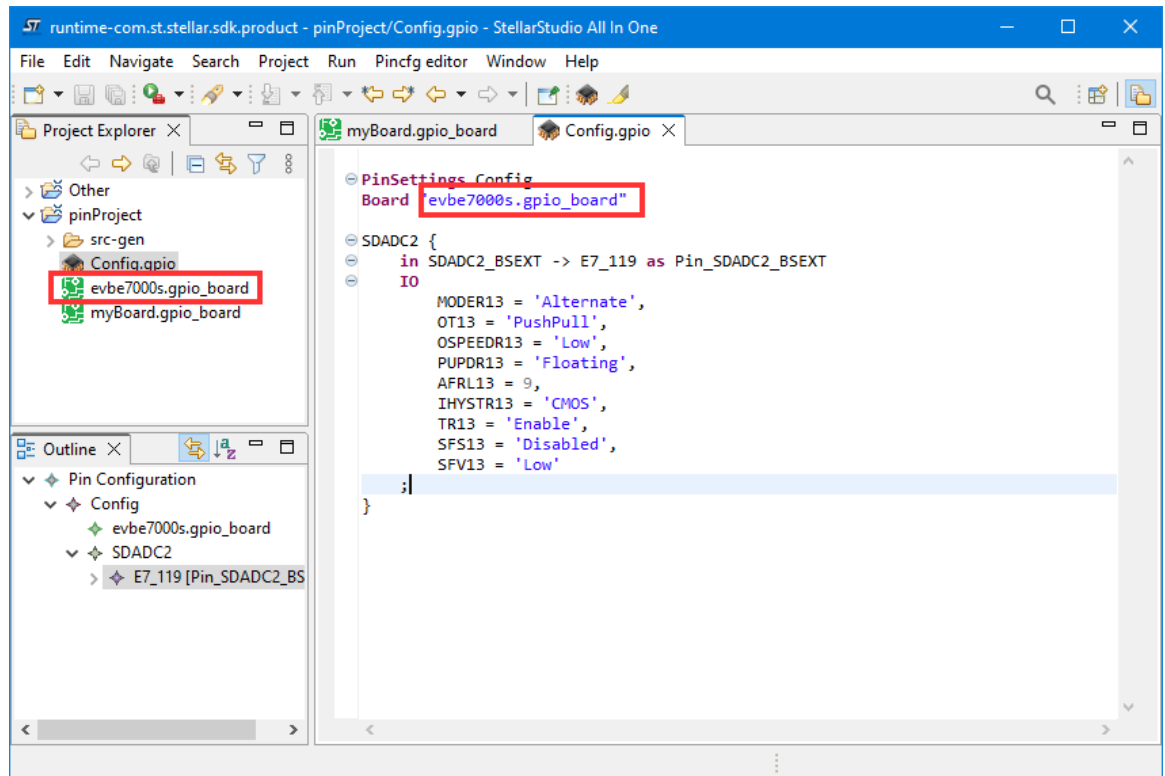
Figure 9: Copy to current project



Please note that this option would be disabled (grayed out) if the included board is already included into your project.

The tool will propose you to copy the file locally to your project (with a dedicated file dialog) and will modify the configuration to import the board from the new chosen location, as in the below example (please note the change in the board imported board path).

Figure 10: Changed imported board path



3 Manipulation of a pinmap configuration

The following section describes how to manipulate a pinmap configuration.

3.1 Magic key

As pinmap configuration manipulation is done using a textual file, the tool provides a so called "*magic key*" used to provide contextual help to the user, depending on the context (i.e. the location of the textual cursor inside the editor).

There are two key combinations available, which offer the same content assist function.

- <CTRL><Space bar>
- <CTRL><TAB>

In the following sections, the



Will be used to remind you to use the *magic key* combination

3.2 Manipulation of the configuration

The pin settings needs a name, a pinmap model and an associated package.

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

The "include" keyword introduces the URI to the pinmap model... This file is provided by the "StellarStudio" team.

Now you can edit the file, add the peripherals and their corresponding configurations.



Hint: remember that at any time you can use the *magic key* combination so that the editor will help you with a contextual content assist helper menu.

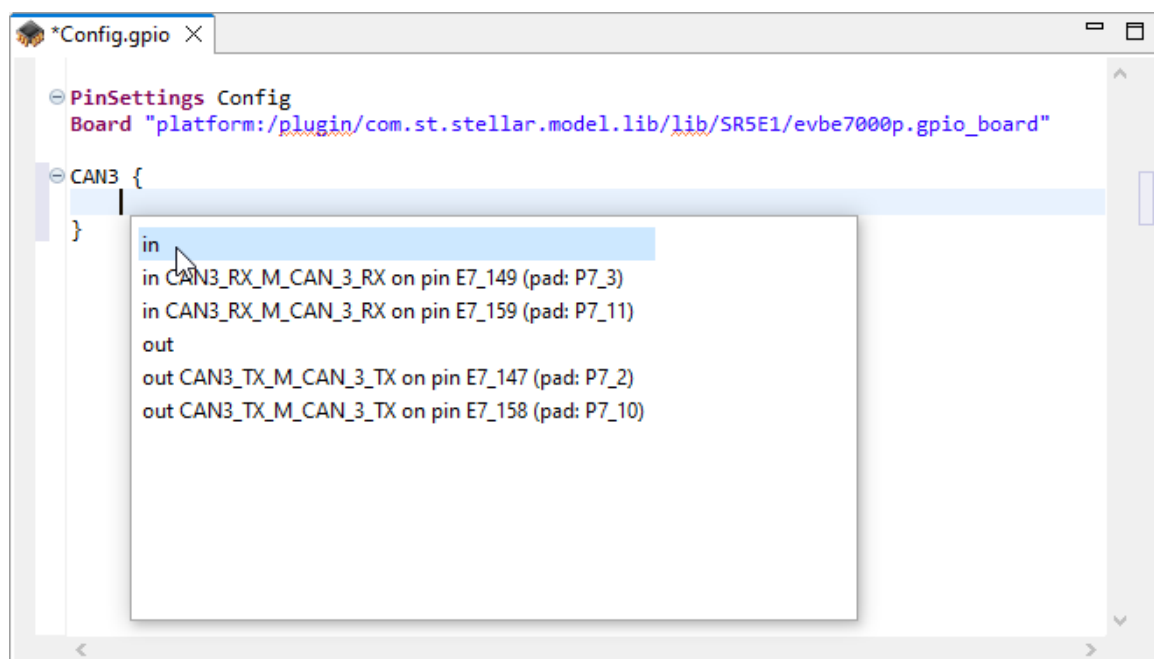
Figure 11: Choose a peripheral



add an opening bracket,

Now choose a pin to configure,

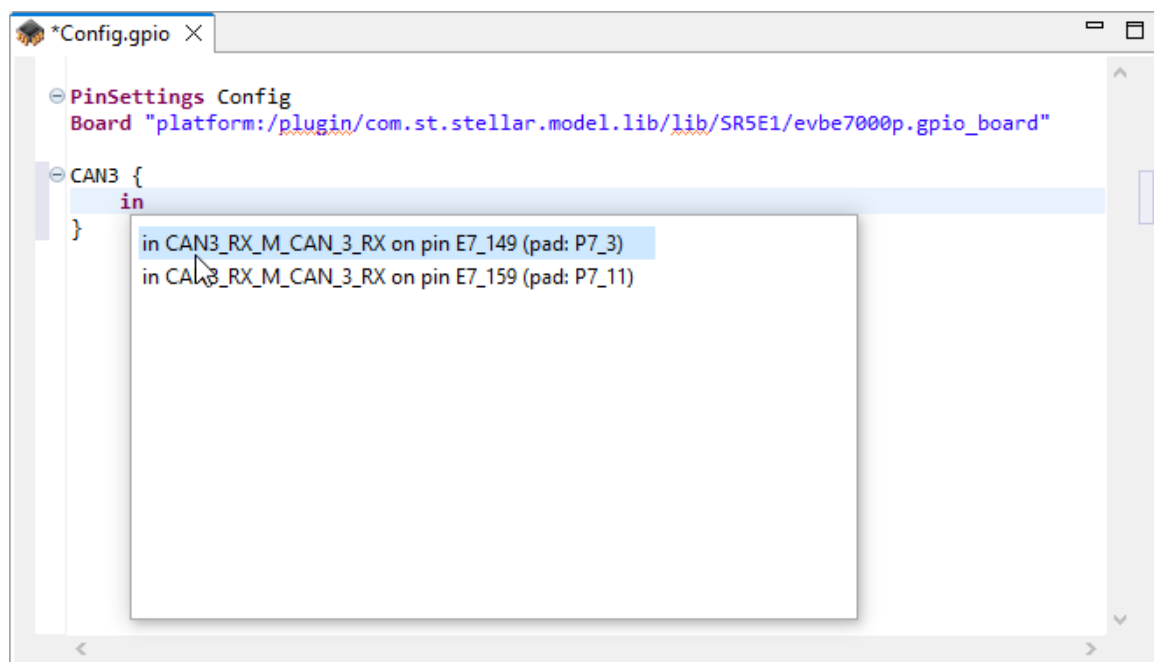
Figure 12: Choose a direction, ...





magic key again

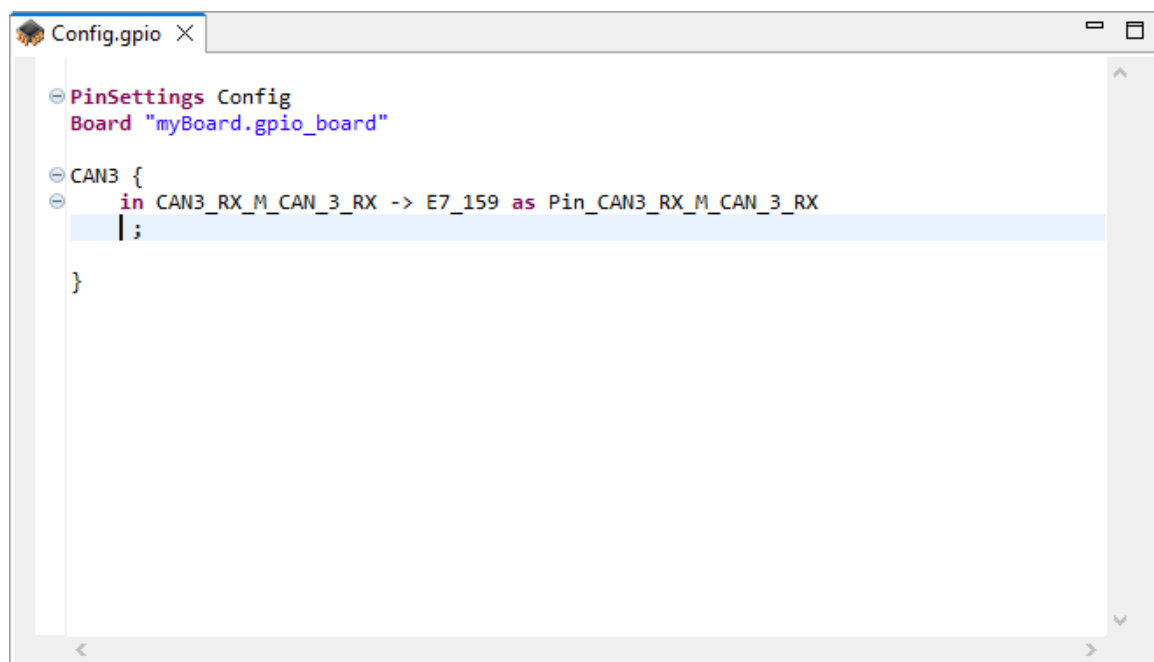
Figure 13: ... then a function associated to a pin



The pin is assigned a default name, that the user can change (the keyword introducing the pin name is 'as'. By default the pin name is named after the associated function, with the 'Pin_' prefix.

The syntax here says that the input function *CAN3_RX_M_CAN_3_RX* is assigned to the pin *E7_159* and is named *Pin_CAN3_RX_M_CAN_3_RX*.

Figure 14: Pin is chosen

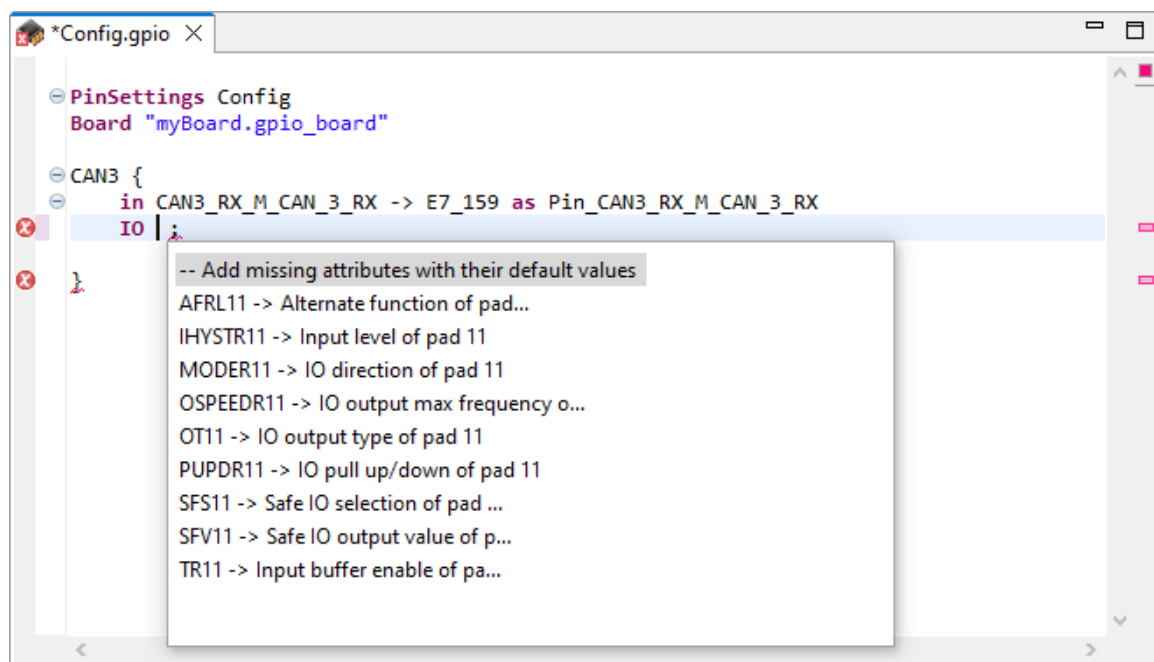


Now you must add IO settings (registers configuration for the created pin). You need to add the 'IO' keyword, then use <CTRL><Space bar> again to add registers values for your pin configuration.



magic key again

Figure 15: Add registers bitfields

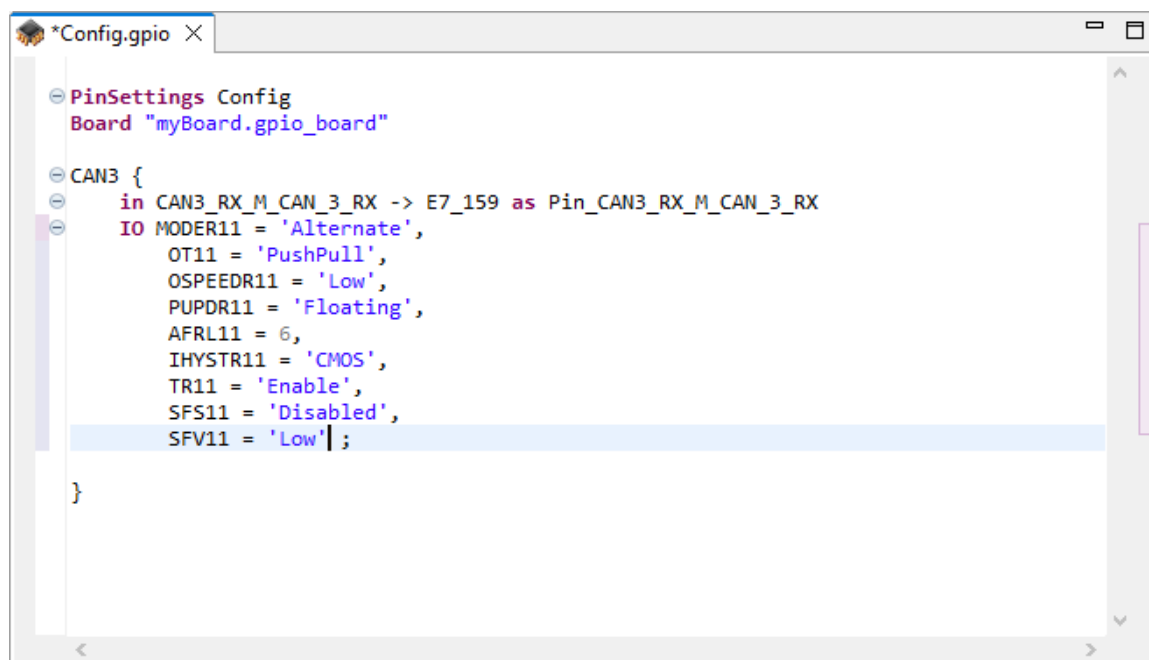


You can assign either all default bit fields, choosing the first option in the menu *add missing attributes with their default values* or add individual bit fields at a time.

Please refer to the appropriate MCU reference manual whenever you need some details on the bit fields possible configurations.

For the chosen register, you need to assign a value. There are some bit fields for which an enumerated value is provided. In such a case, the content assist helps you once more, providing you with the different available values.

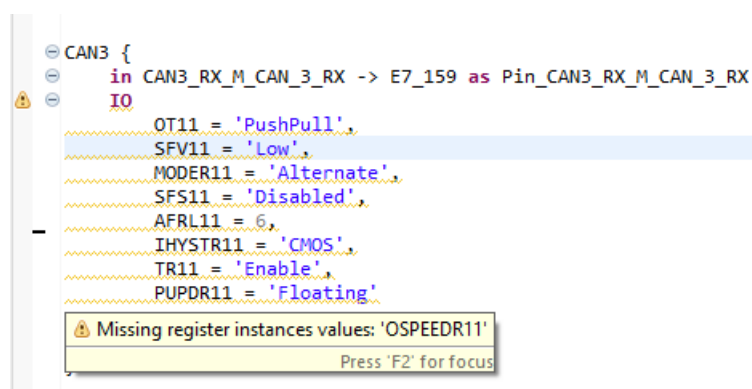
Figure 16: Resulting configuration



Please note that even if the configuration has been completed with default values, you can choose to change attributes' values, or remove some of them, nevertheless, missing values will be considered as warnings.

--> as long as the pin configuration does not contain the whole set of register attributes, the configuration will appear with a warning.

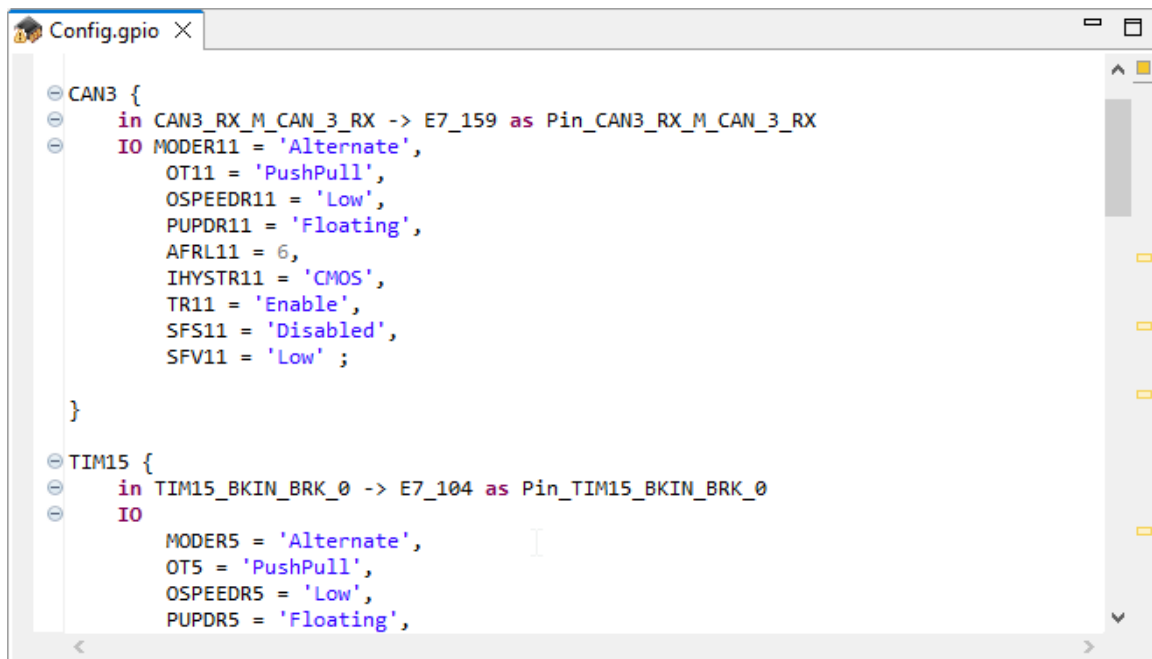
Figure 17: Warnings on incomplete pin configurations



Now, you can add some additional pins inside the peripheral by using <CTRL><Space bar>.

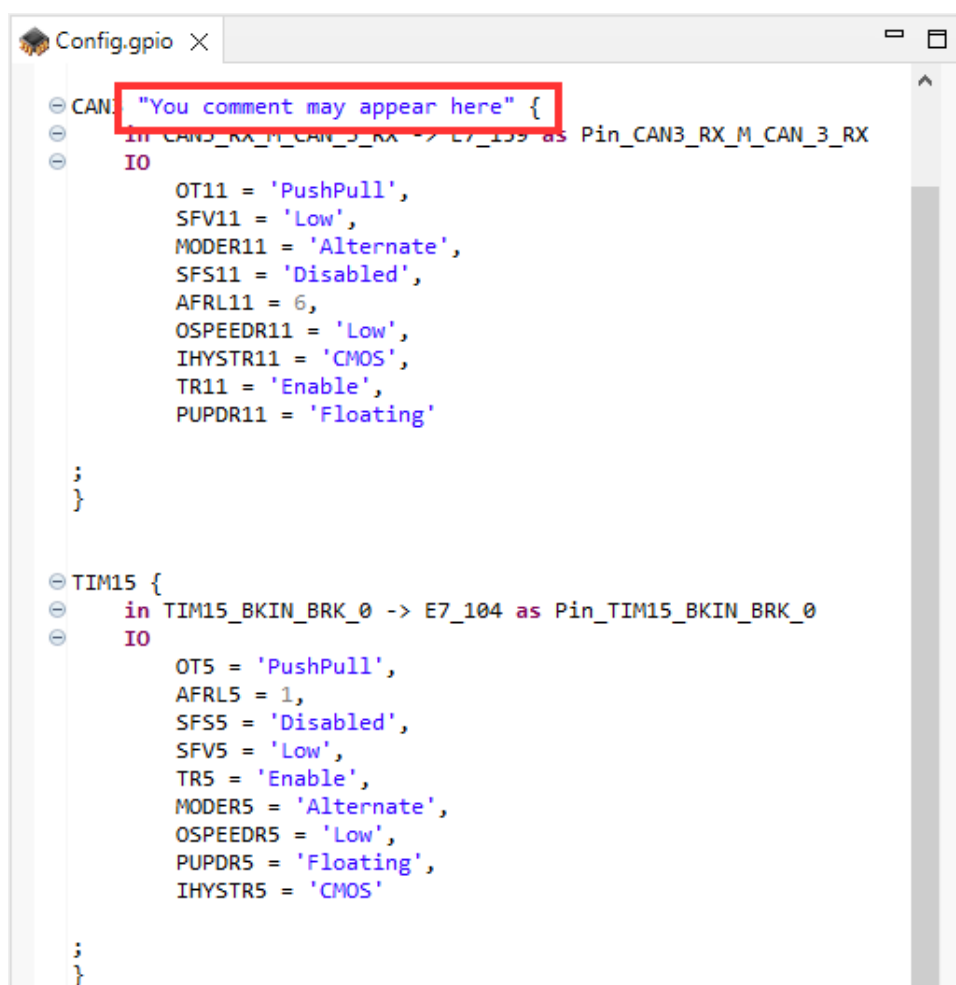
You can notice that functions already used by a previous configuration are not displayed in the contextual menu.

Figure 18: Add some additional pins



You can add a comment with your peripheral. You must add this comment after the peripheral name, and before the { character.

Figure 19: Add some comments



Inside the generated file, the comment will appear as below:

Figure 20: Comments are added to the generated code

```
/*=====
/* Module constants.
/*=====

#define EVBE7000P 1

/*
 * You comment may appear here
 */

#define PIN_CAN3_RX_M_CAN_3_RX gpio_iopack(GPIO_PORT_H, GPIO_PIN_11) /*
#define PIN_CAN3_RX_M_CAN_3_RX_CFG \
(GPIO_MODE_OTYPER_PUSH_PULL | \
 GPIO_MODE_SAFEVALR_LOW | \
 GPIO_MODE_MODER_ALTERNATE | \
 GPIO_MODE_SAFESELR_DISABLED | \
 GPIO_MODE_AFR(6U) | \
 GPIO_MODE_OSPEEDR_LOW | \
 GPIO_MODE_IHYSTR_CMOS | \
 GPIO_MODE_TRIGENR_IN_ENABLED | \
 GPIO_MODE_PUPDR_FLOATING)

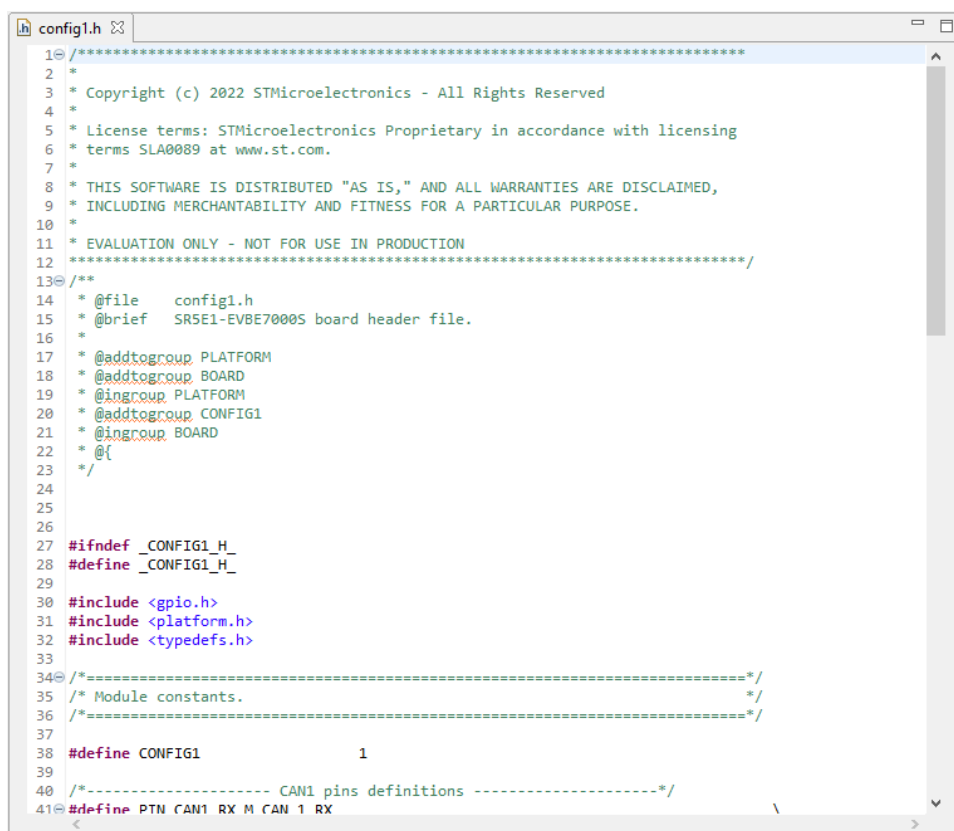
/*----- TTM15 pins definitions -----*/
```

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 "src-gen".

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

Figure 21: Code generated from the configuration



```

1  /*****
2  *
3  * Copyright (c) 2022 STMicroelectronics - All Rights Reserved
4  *
5  * License terms: STMicroelectronics Proprietary in accordance with licensing
6  * terms SLA0089 at www.st.com.
7  *
8  * THIS SOFTWARE IS DISTRIBUTED "AS IS," AND ALL WARRANTIES ARE DISCLAIMED,
9  * INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
10 *
11 * EVALUATION ONLY - NOT FOR USE IN PRODUCTION
12 *****/
13 /**
14  * @file    config1.h
15  * @brief    SR5E1-EVBE7000S board header file.
16  *
17  * @addtogroup PLATFORM
18  * @addtogroup BOARD
19  * @ingroup  PLATFORM
20  * @addtogroup CONFIG1
21  * @ingroup  BOARD
22  * @{
23  */
24
25
26
27 #ifndef _CONFIG1_H_
28 #define _CONFIG1_H_
29
30 #include <gpio.h>
31 #include <platform.h>
32 #include <typedefs.h>
33
34 /*=====*/
35 /* Module constants. */
36 /*=====*/
37
38 #define CONFIG1                1
39
40 /*----- CAN1 pins definitions -----*/
41 #define PTN CAN1_RX M CAN1_RX

```

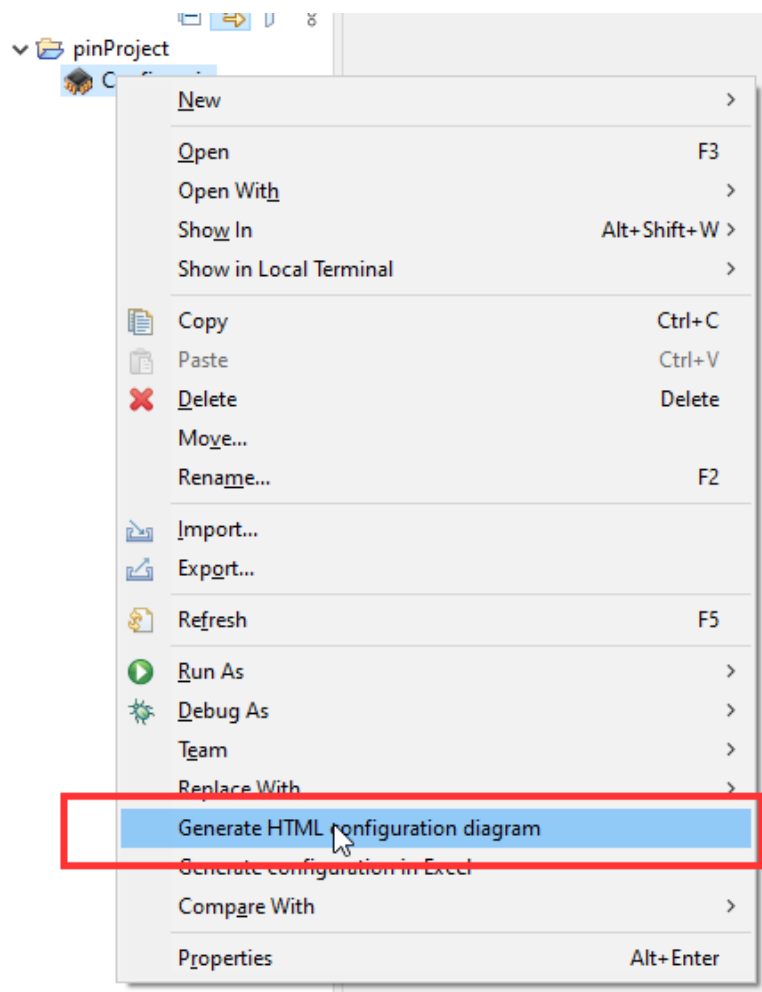
The pin settings are generated with a comment indicating an explanation of the corresponding values, instead of just the hexadecimal values.

3.4 HTML generation

The HTML generation is **not** automatic, but must be done from the project explorer, right clicking on the .gpio file, asking for HTML generation.

The generated file will be located inside the project's 'src-gen' folder.

Figure 22: Generate the HTML file



The developer can open the *.html* file by the contextual menu on *.gpio* file editor area.

Remember to regenerate the html file each time a modification is made into the gpio, as it will not be re-generated by default.

Figure 23: Open an HTML file

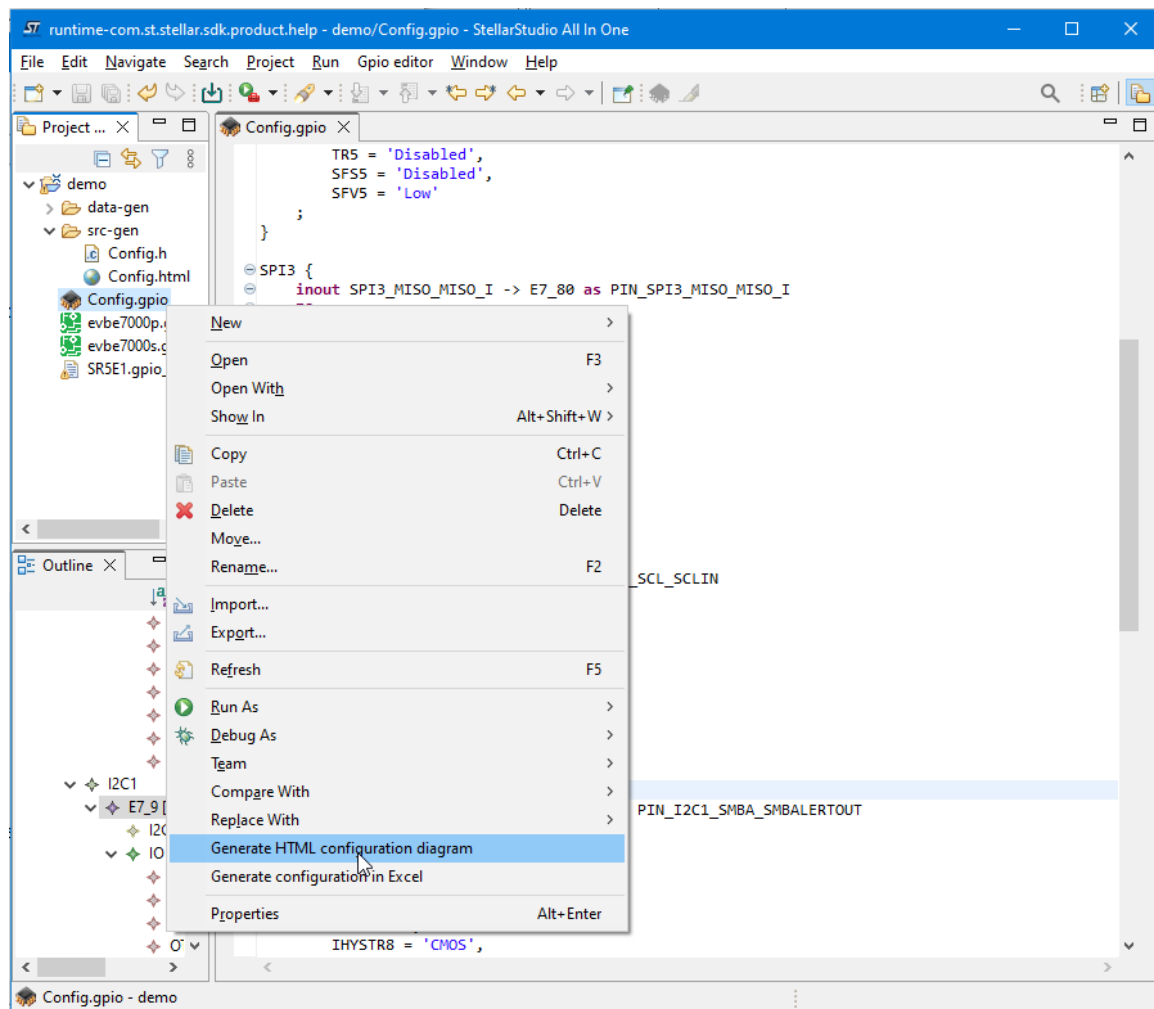
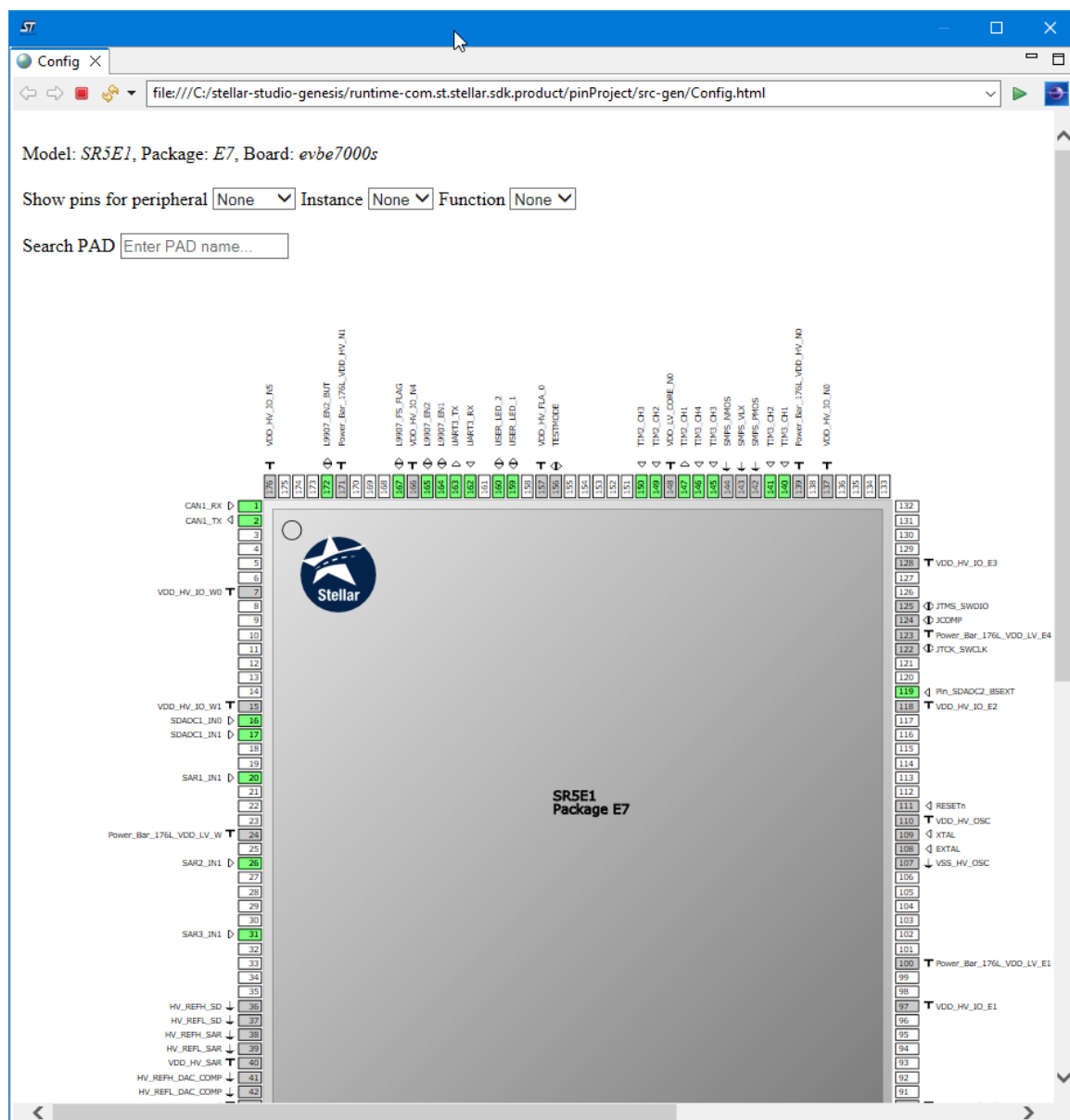


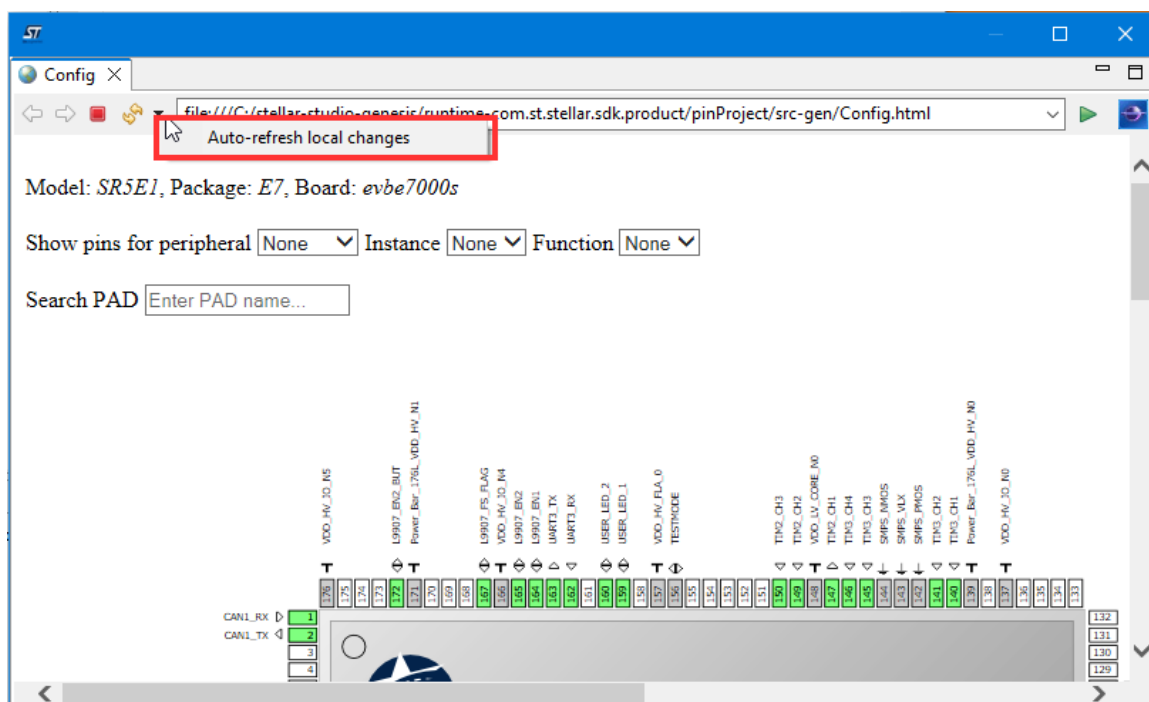
Figure 24: HTML generated from the configuration



It is possible to filter and display by peripheral and subpart of the peripheral.

The dynamic list and the tooltip comment are very useful to know your current configuration.

Figure 25: Auto-refresh local changes



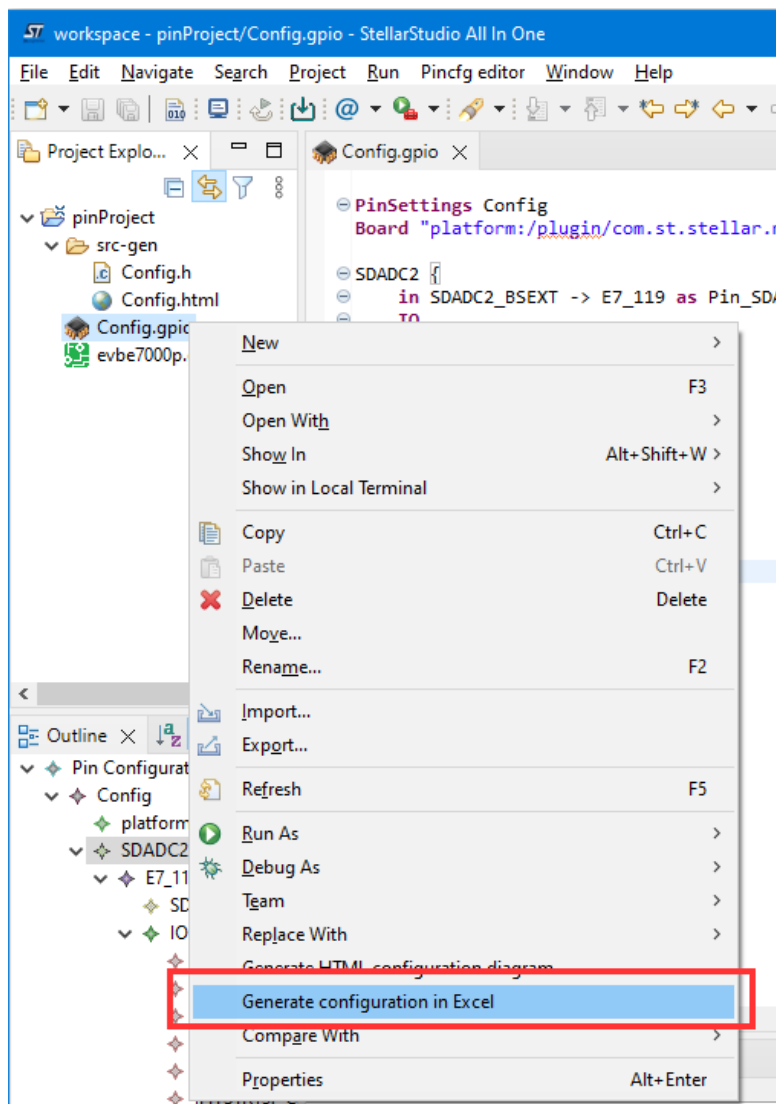
in order to have the HTML file synchronized with your current pin configuration, the developer can enable *Auto-refresh local changes*

3.5 XLS generation

The XLS generation is **not** automatic, but must be done from the project explorer, right clicking on the .gpio file, asking for Excel generation.

The generated file will be located inside the project's 'src-gen' folder.

Figure 26: Generate the Excel file



The developer can then open the XLS file by Microsoft Excel® in the *src-gen* folder, using right-click + open With.../System Editor menu option.

WARNING : double-clicking is not supported because no OLE editor is supported

Figure 27: Open an XLS file

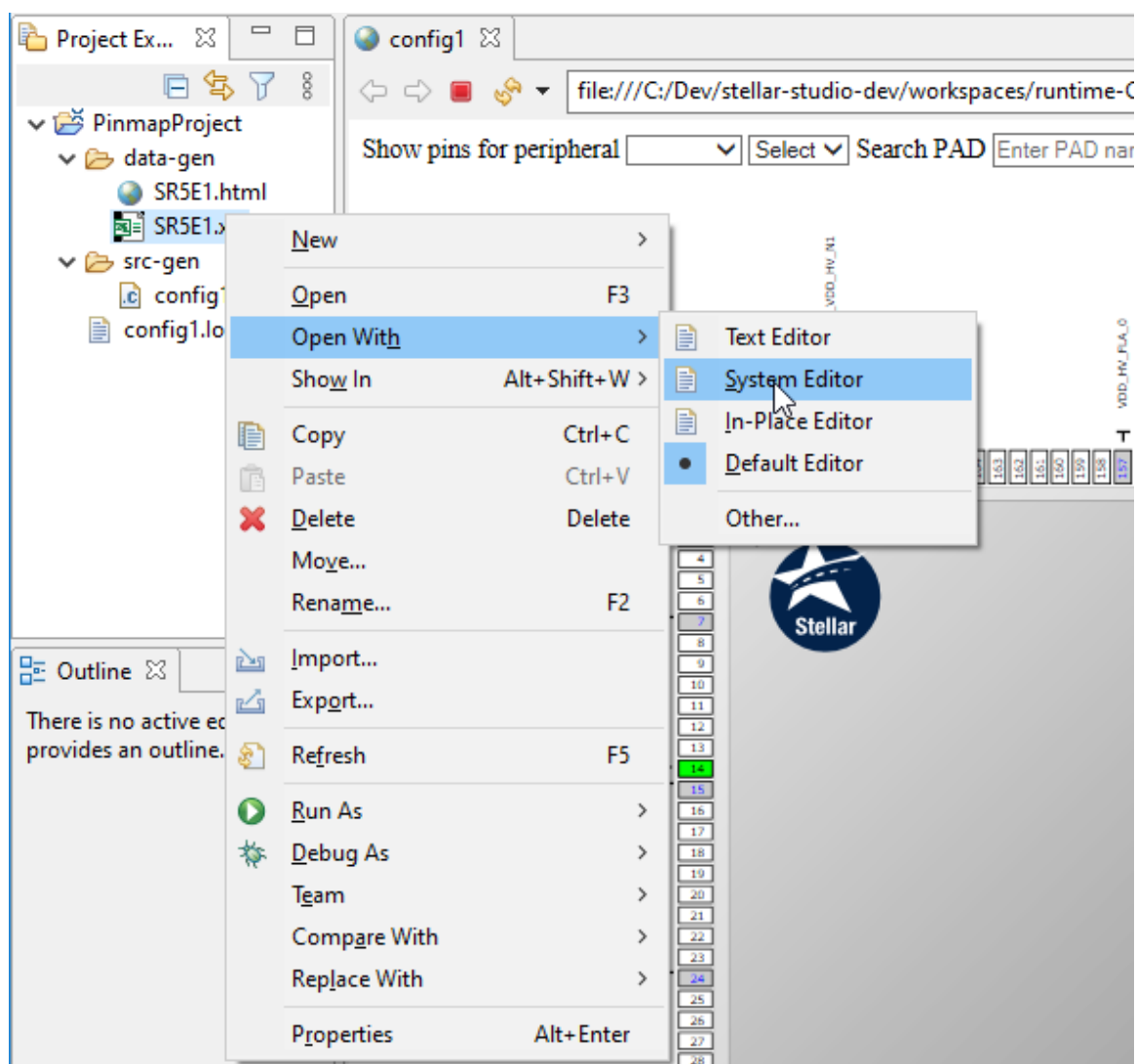
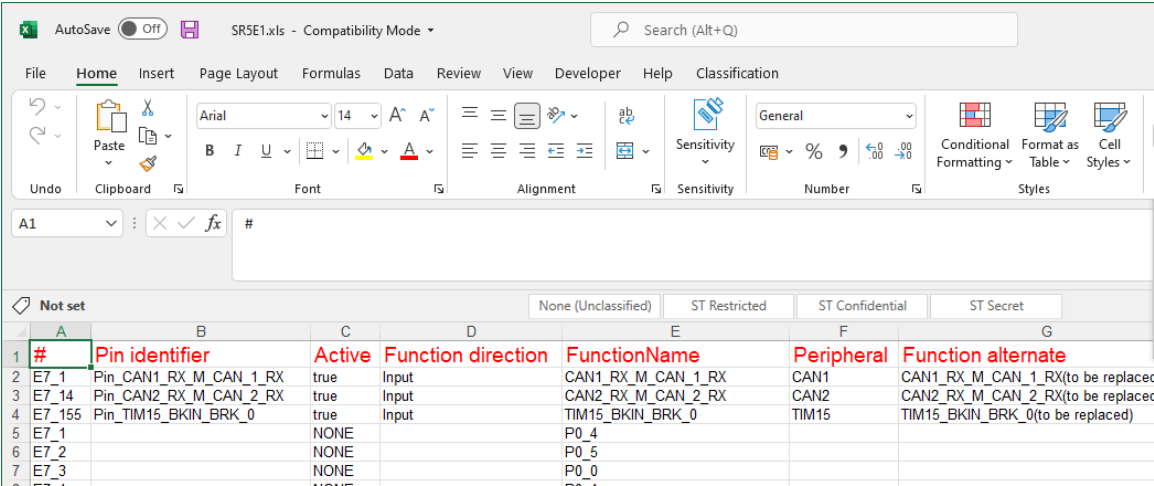


Figure 28: Example of XLS generated from the configuration



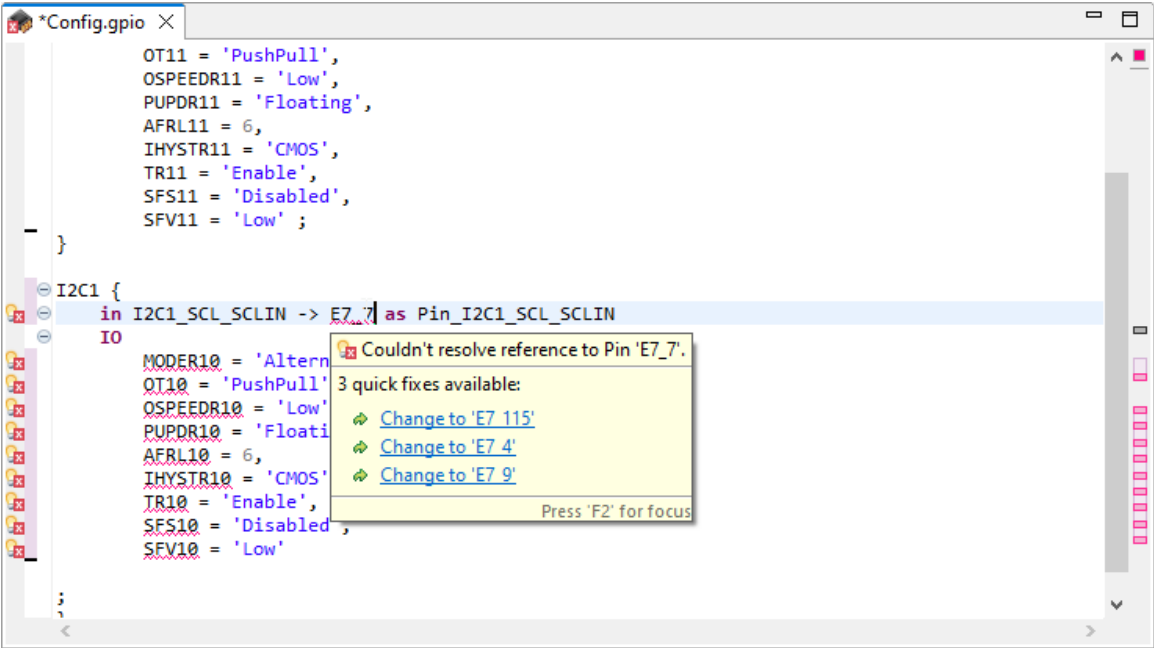
The screenshot shows an Excel spreadsheet titled 'SR5E1.xls - Compatibility Mode'. The spreadsheet contains a table with the following data:

#	Pin identifier	Active	Function direction	FunctionName	Peripheral	Function alternate
E7_1	Pin_CAN1_RX_M_CAN_1_RX	true	Input	CAN1_RX_M_CAN_1_RX	CAN1	CAN1_RX_M_CAN_1_RX(to be replaced)
E7_14	Pin_CAN2_RX_M_CAN_2_RX	true	Input	CAN2_RX_M_CAN_2_RX	CAN2	CAN2_RX_M_CAN_2_RX(to be replaced)
E7_155	Pin_TIM15_BKIN_BRK_0	true	Input	TIM15_BKIN_BRK_0	TIM15	TIM15_BKIN_BRK_0(to be replaced)
E7_1		NONE		P0_4		
E7_2		NONE		P0_5		
E7_3		NONE		P0_0		

3.6 Configuration errors

Errors in naming peripherals, pin configuration, as well as syntax errors in the expected keywords, are identified with some markers in the configuration editor.

Figure 29: Syntax Errors



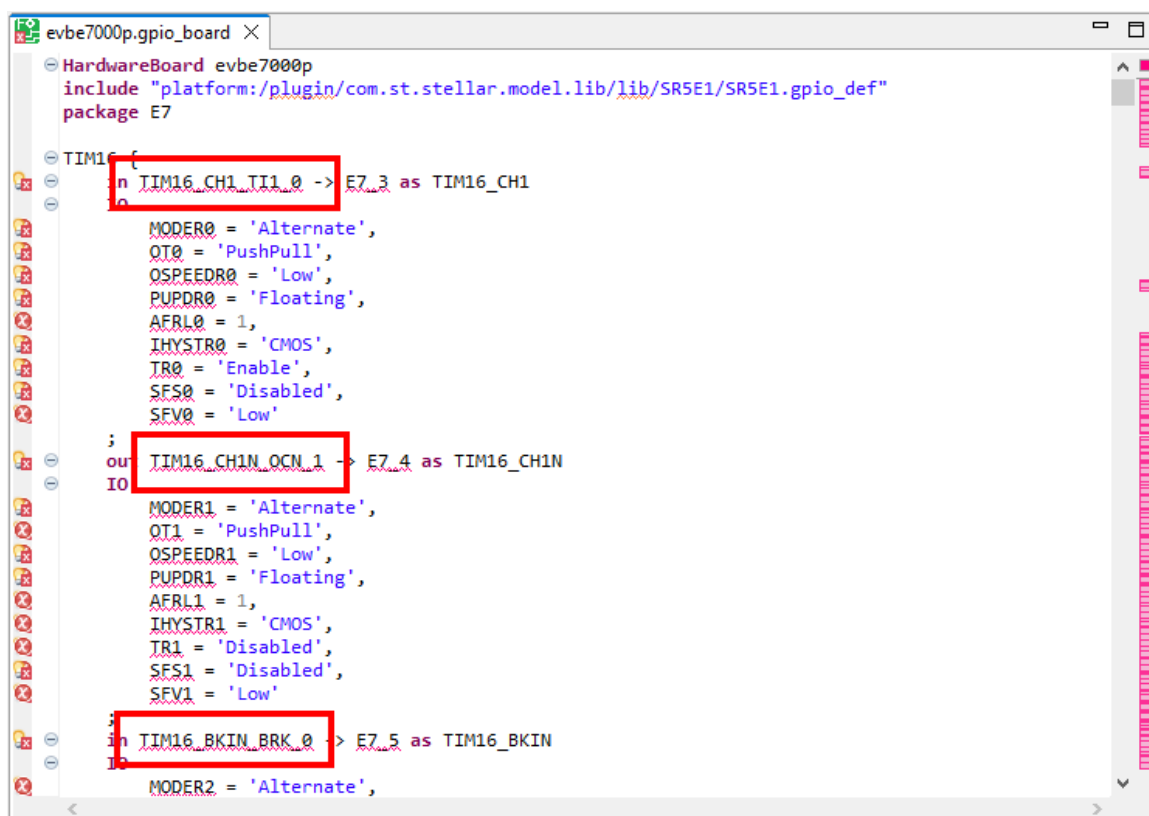
For some of these errors, a quick fix is available.

3.7 Application migration

Between successive versions of the tool the underlying model of pinmap definitions may change.

In such a case, errors will appear inside the configurations when launching the tool after the update.

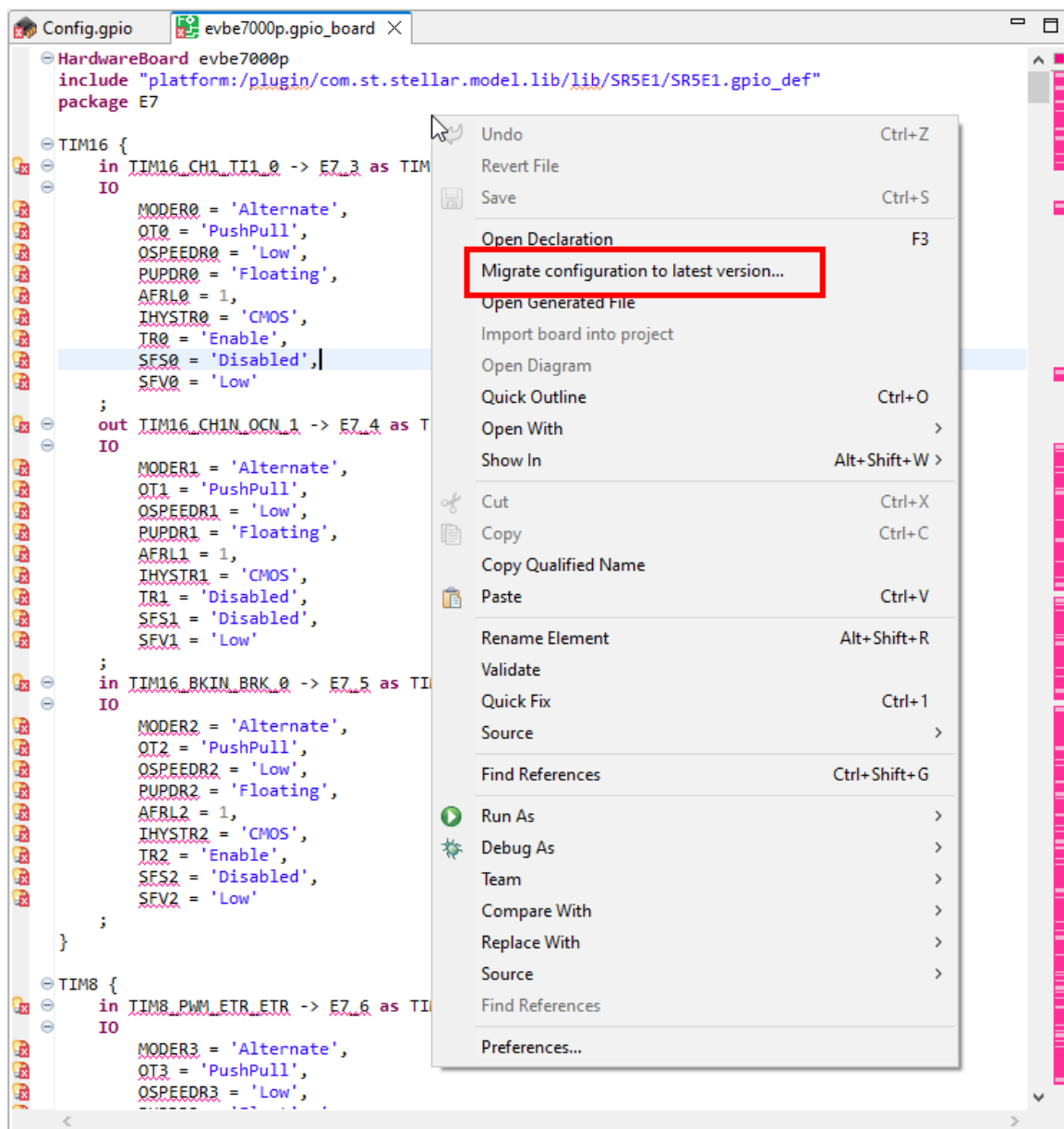
Figure 30: Old applications Migration



In the above picture, some errors have appeared because the underlying pinmap model has changed.

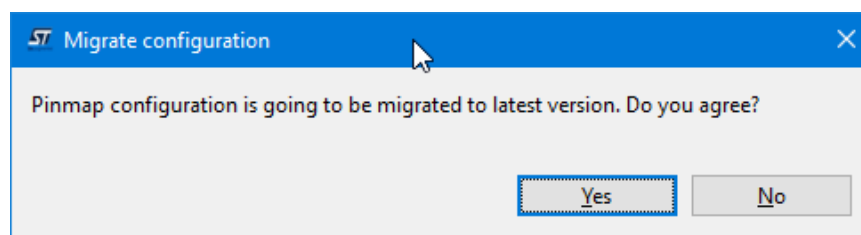
In order to fix these errors, you must use the migration tool that is accessible using right click inside the configuration editor.

Figure 31: Launching migration



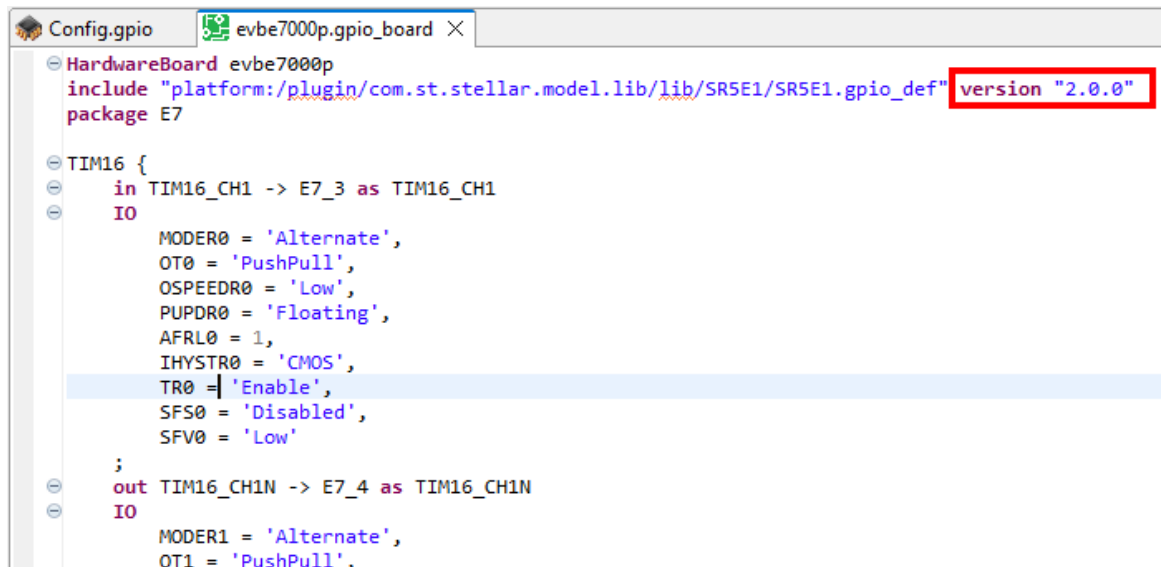
You'll be asked to confirm your choice.

Figure 32: Confirm migration



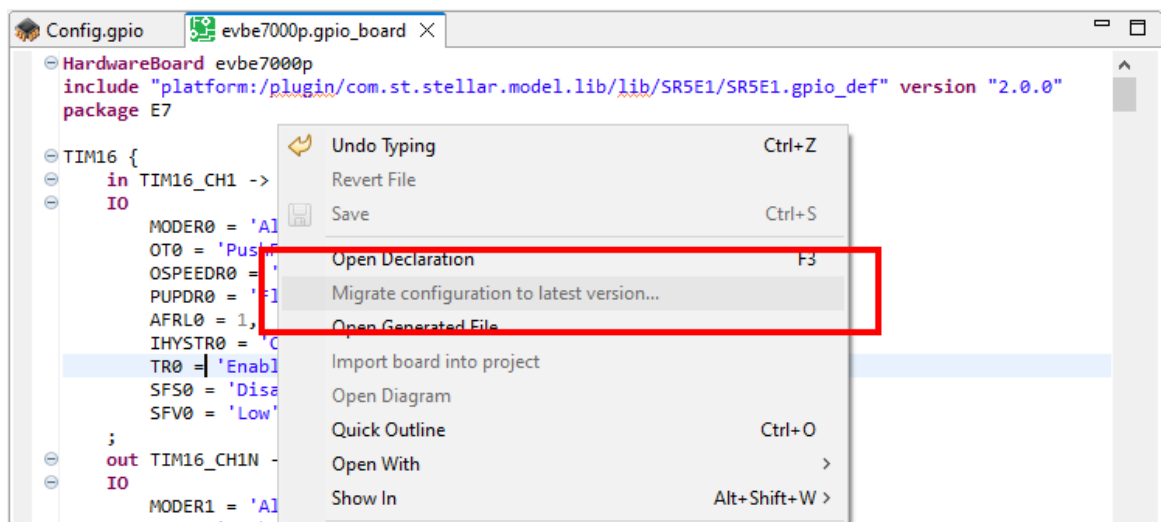
As a result the functions names has been adapted to the new names, and the version of the model is now changed and appears inside the converted file (here 2.0.0).

Figure 33: Migration result



As soon as the migration has been done, it is marked as not needed, and you cannot launch the migration anymore.

Figure 34: Unneeded migration



3.8 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 35: Outline for pin configuration

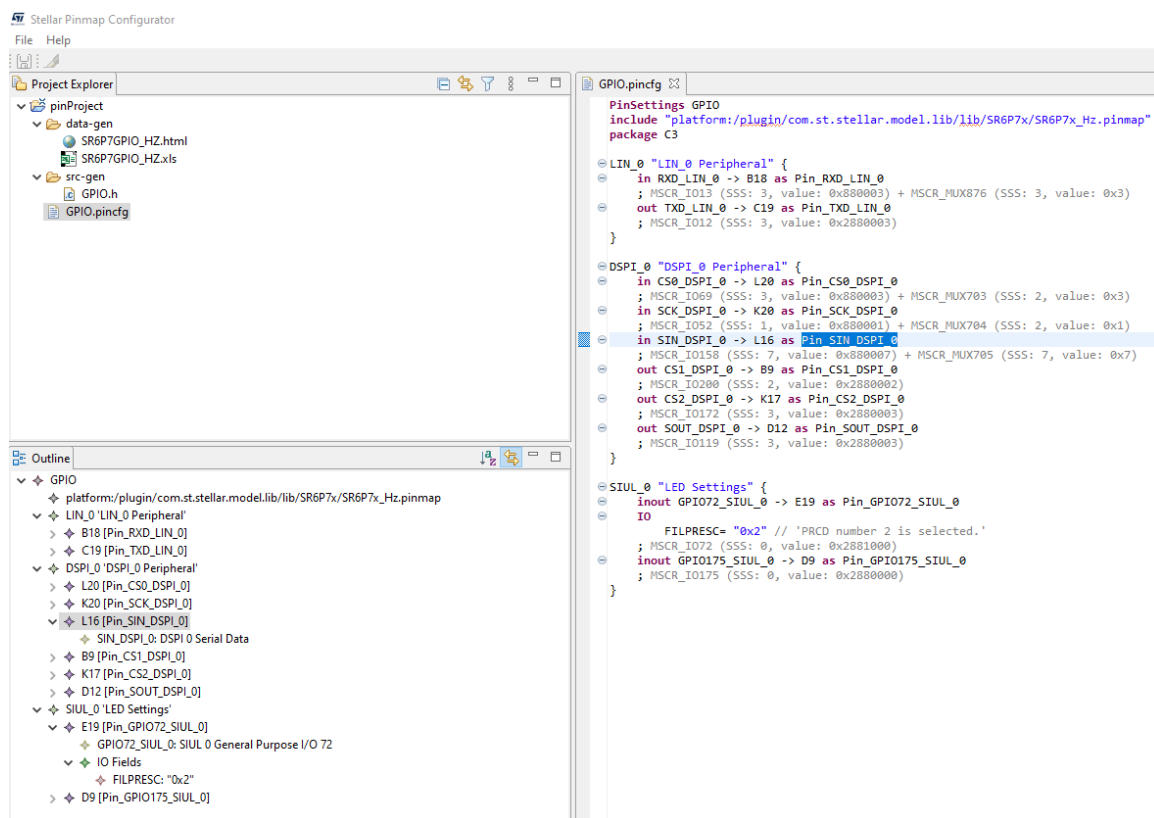
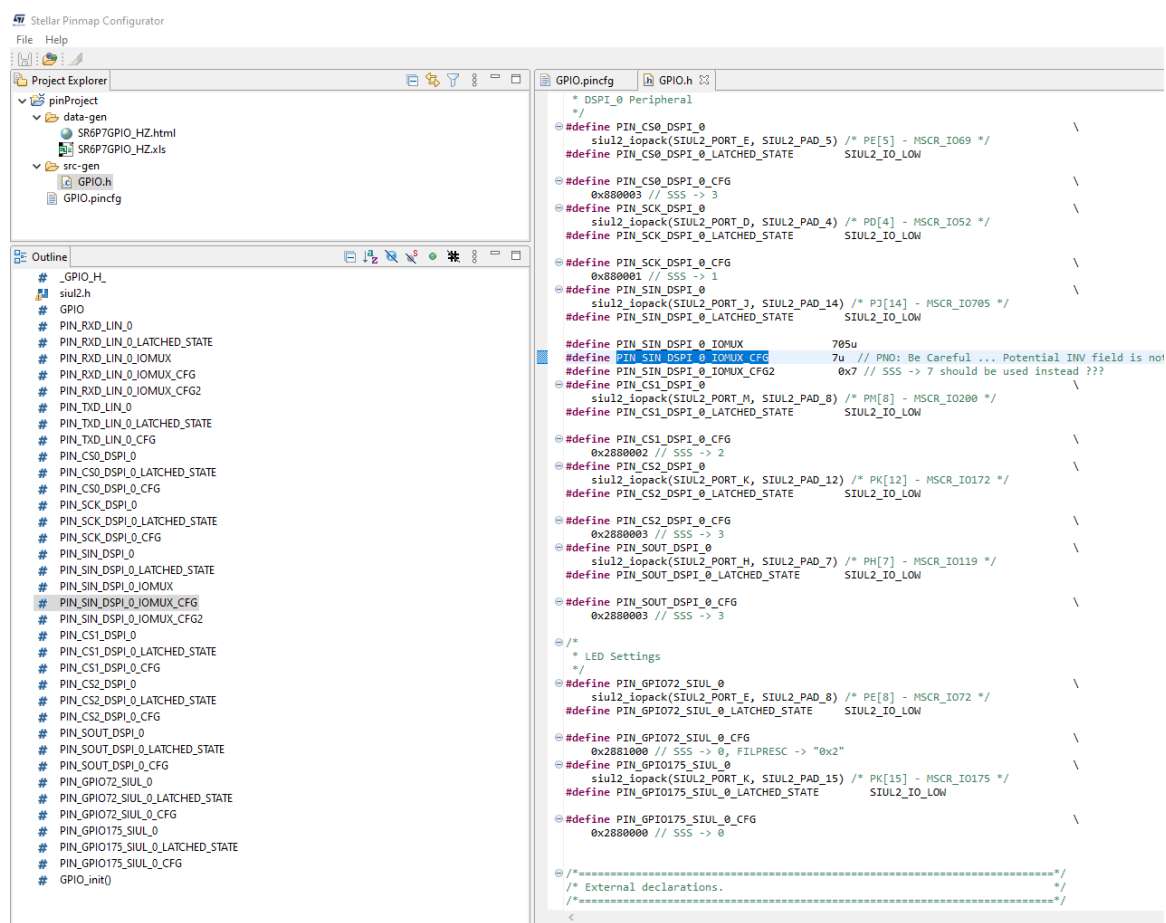


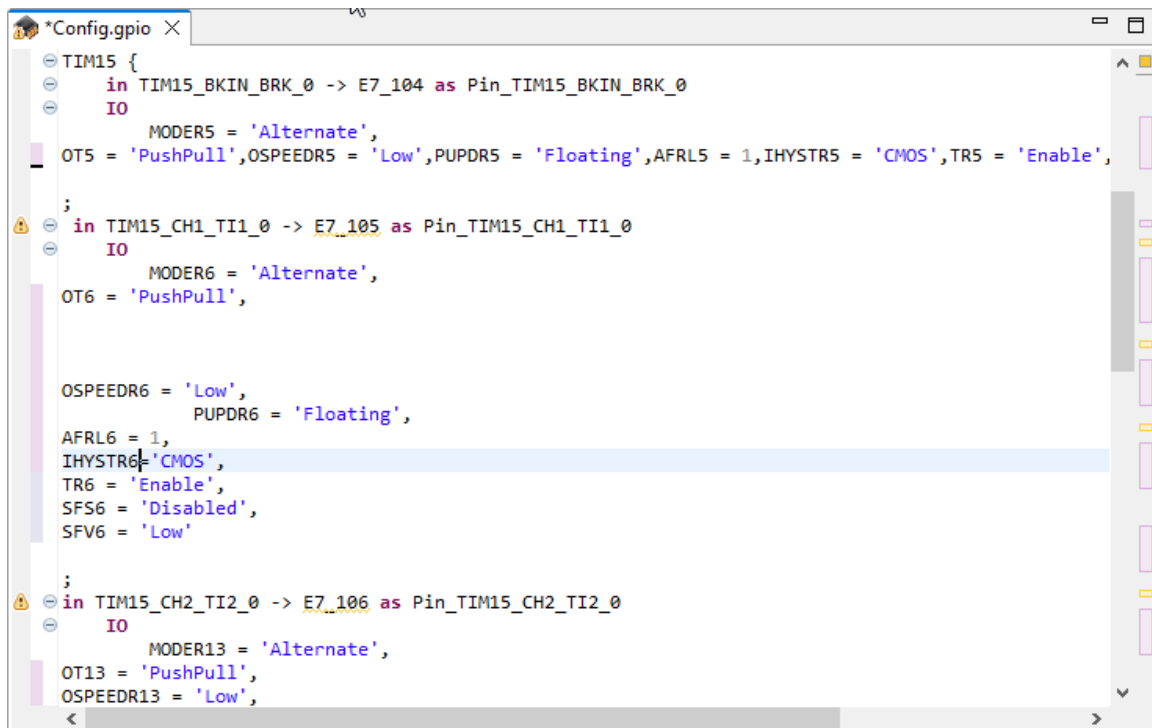
Figure 36: Outline for a C generated file



3.9 Pin 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 37: Example before formatting



And here is how it looks like after formatting:

Figure 38: Example after formatting



4 Graphical configuration of pins

The following section describes how to configure pins graphically.

4.1 Graphical configuration

As described in the previous sections, the fine pin configurations are achieved using the textual domain specific language (DSL) dedicated to the pin configuration. The graphical configuration allows to start configuring a pin, that the user can then fine tune using the textual configurator.

The user will be able to:

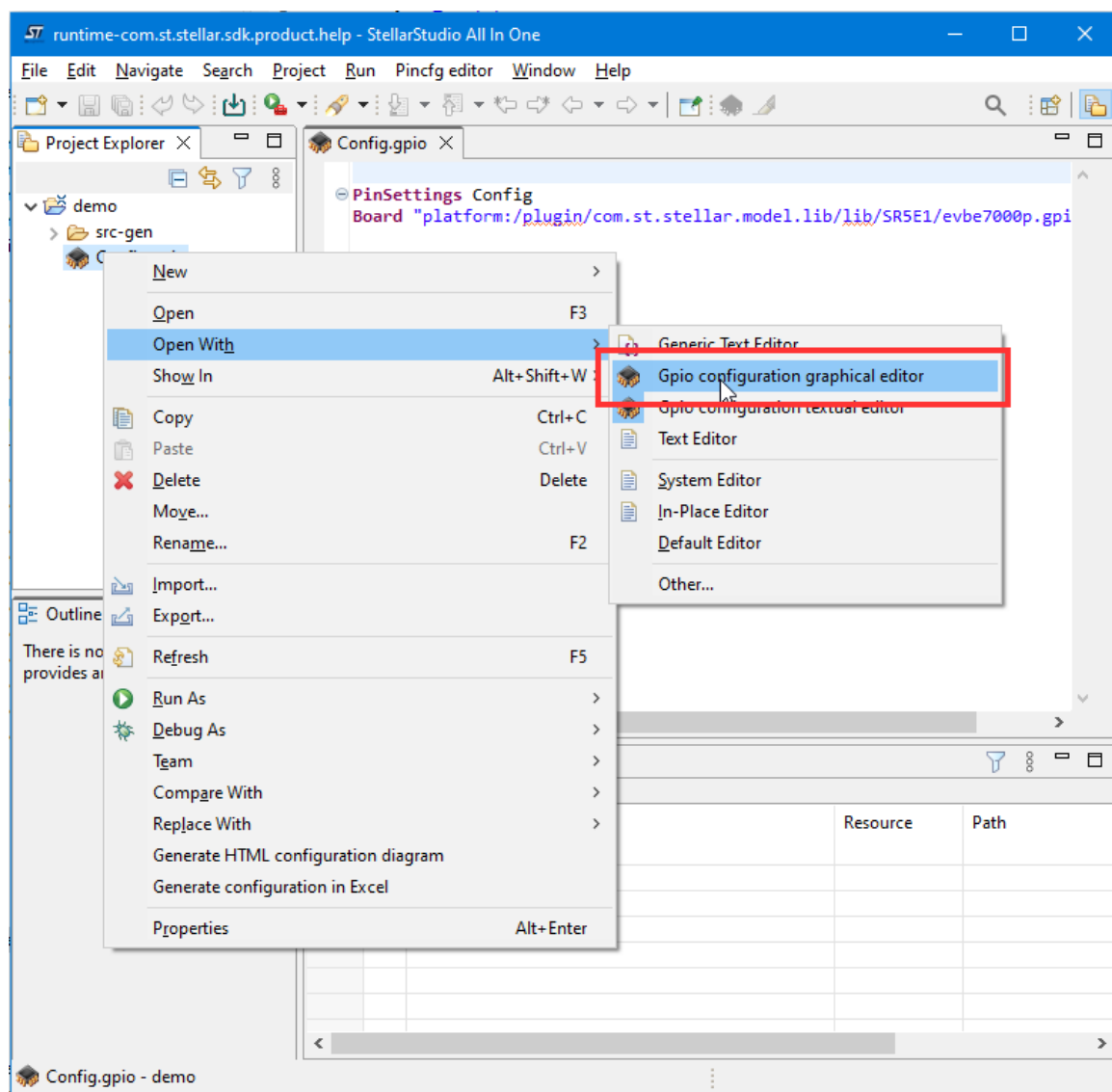
- Open the graphical configuration editor
- Use the outline view to identify where peripherals, functions or PADs are accessible,
- Graphically assign/unassign a function to/from a pin,
- Ask to show a configured pin inside the textual configuration editor,

4.2 Open the graphical configuration editor

There are several ways to open the graphical configuration editor.

You can open it from the project explorer, using a right click on a *.gpio* file, then choose the Open with/Gpio configuration **graphical** editor.

Figure 39: Open the graphical editor from project explorer



Alternatively you can use the Gpio editor menu, or click on the MCU icon in the toolbar.

Figure 40: Open the graphical editor from Gpio editor menu

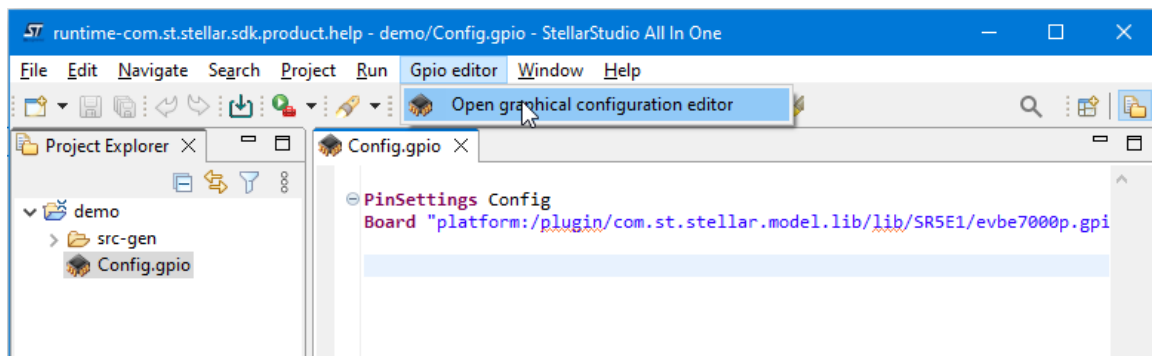
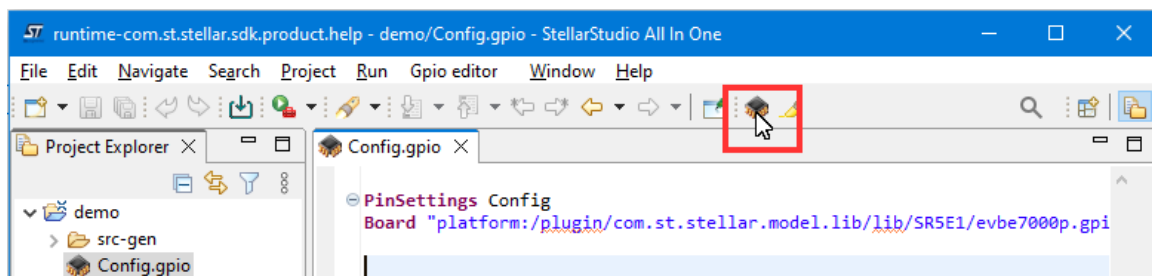
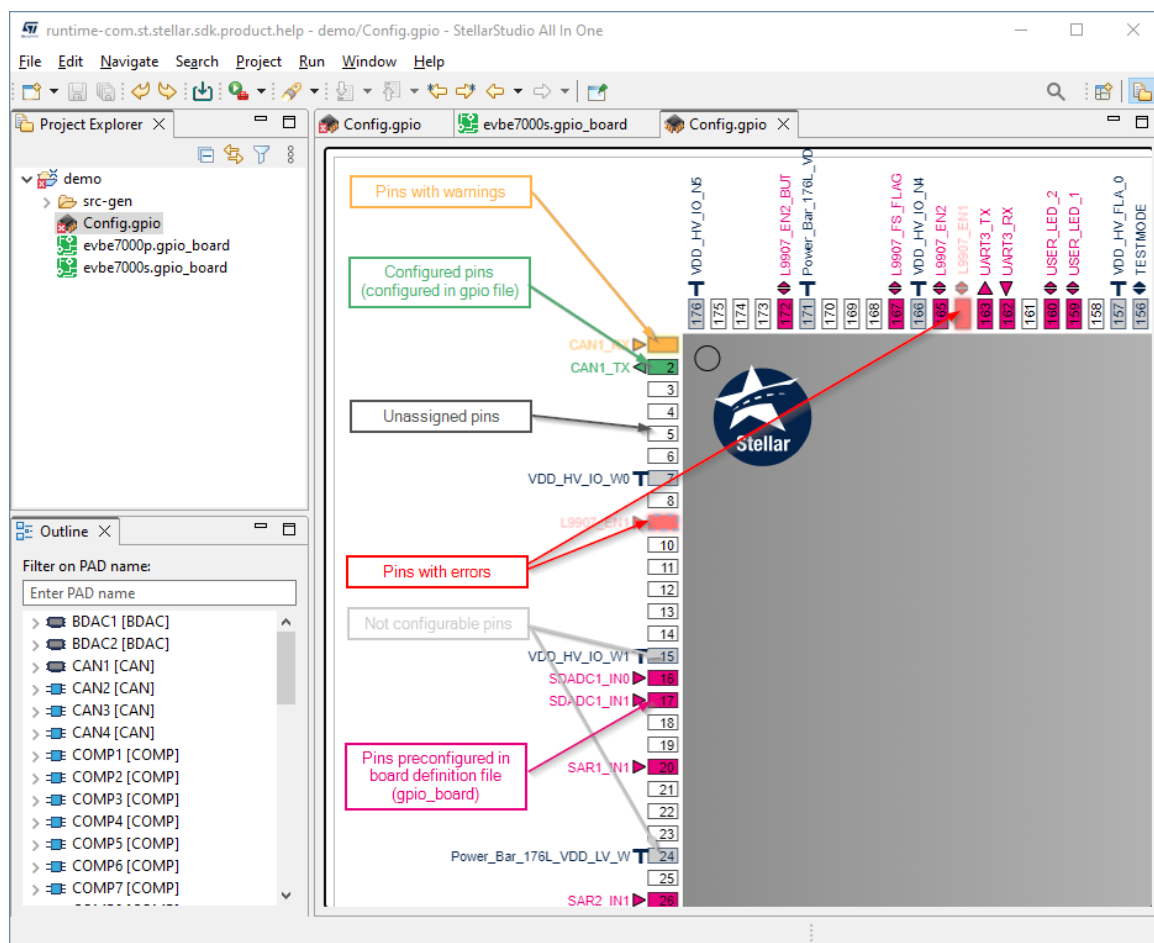


Figure 41: Open the graphical editor from the MCU icon in the toolbar



The result would be similar to the below picture.

Figure 42: Result: opened graphical editor

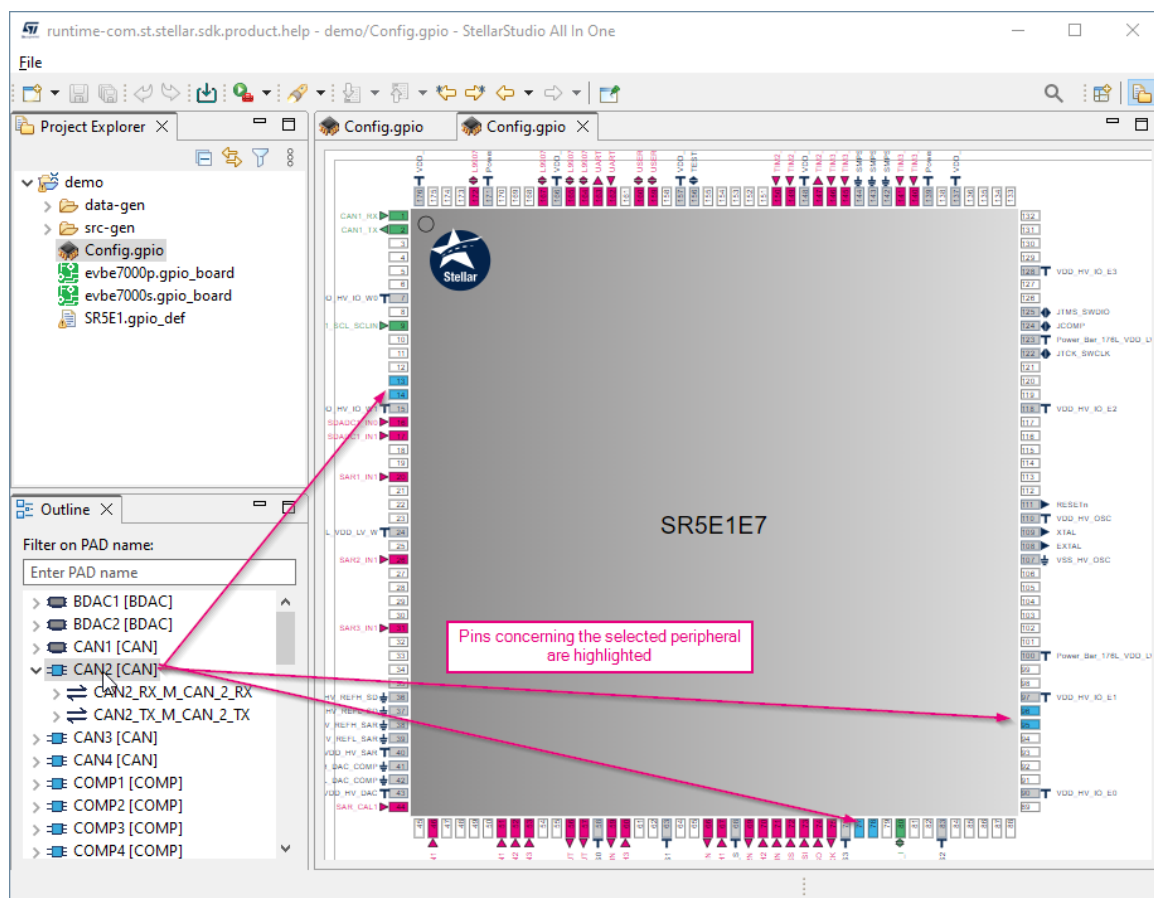


4.3 Using pinmap configuration outline views

Outline views in Eclipse allows to display specific information about the selected object inside an editor.

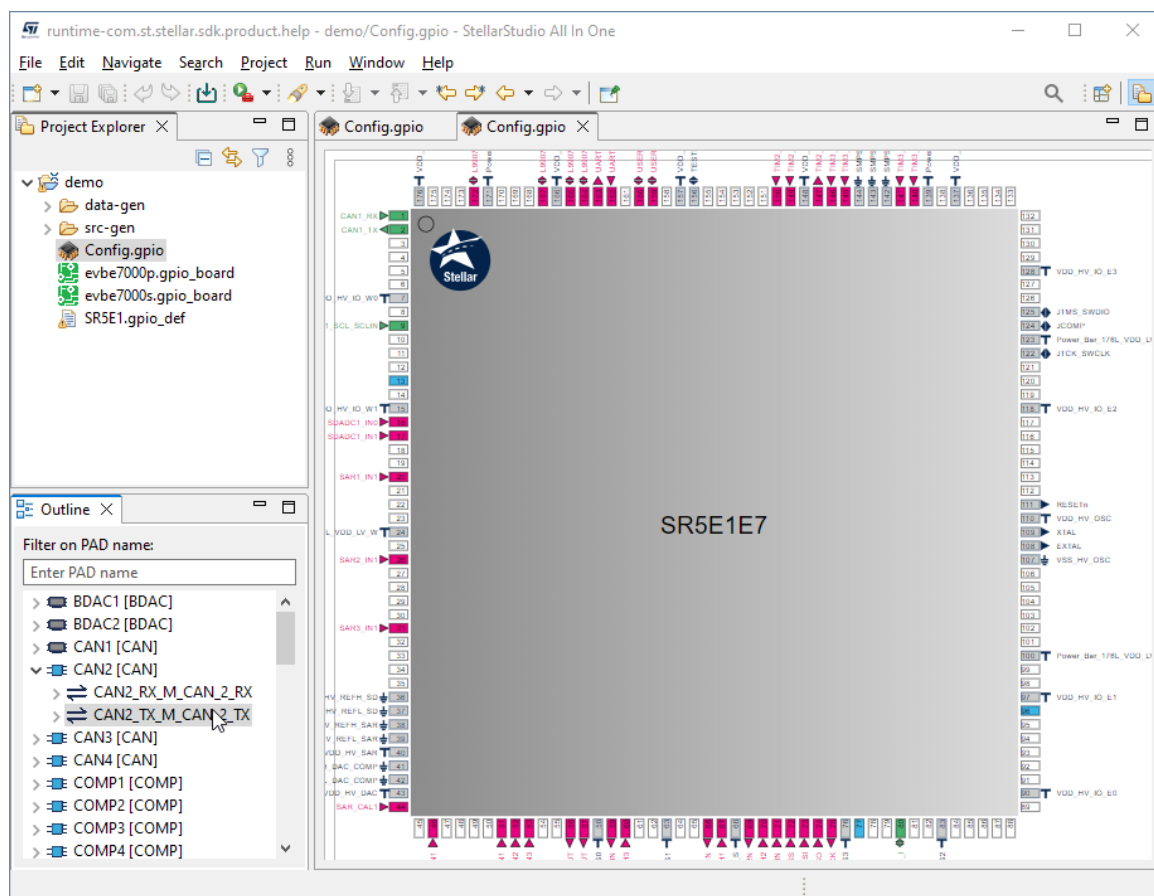
For the Gpio configuration textual editor, it displays the list of peripherals, with their assigned functions.

Figure 43: Example of selection a peripheral



For the Gpio configuration graphical editor, it shows the peripherals, with their assigned functions and the concerned pins.

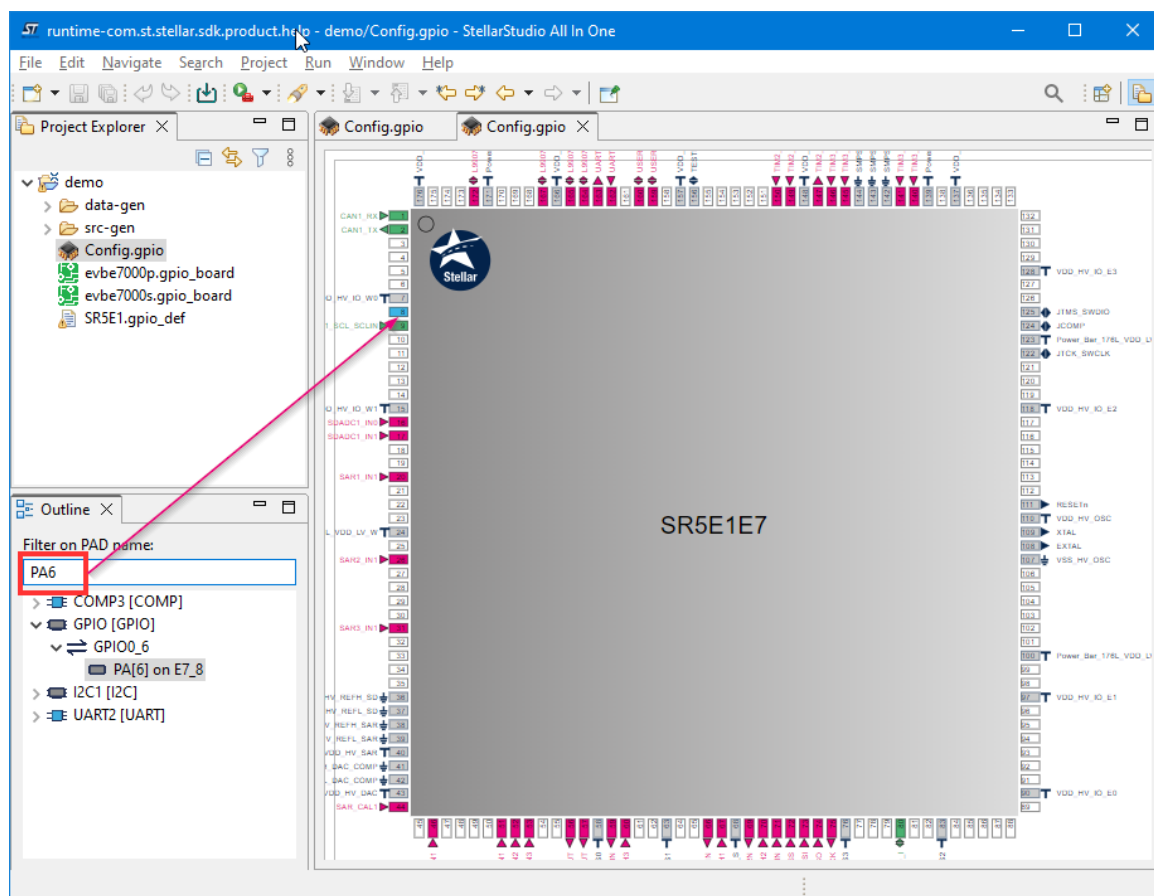
Figure 44: Example of selection a function



In particular in this view, the user can search for a particular PAD, using the filter text field.

All pins that matches the entered pad name will be displayed, so that the user can select them. A selected object in the outline would be displayed inside the graphical editor. For example, if you select a peripheral, all pins related to this peripheral will be highlighted. If you select only a function, then you can see (highlighted in the graphical editor) all the pins where you can assign this function.

Figure 45: Example of searching of a PAD

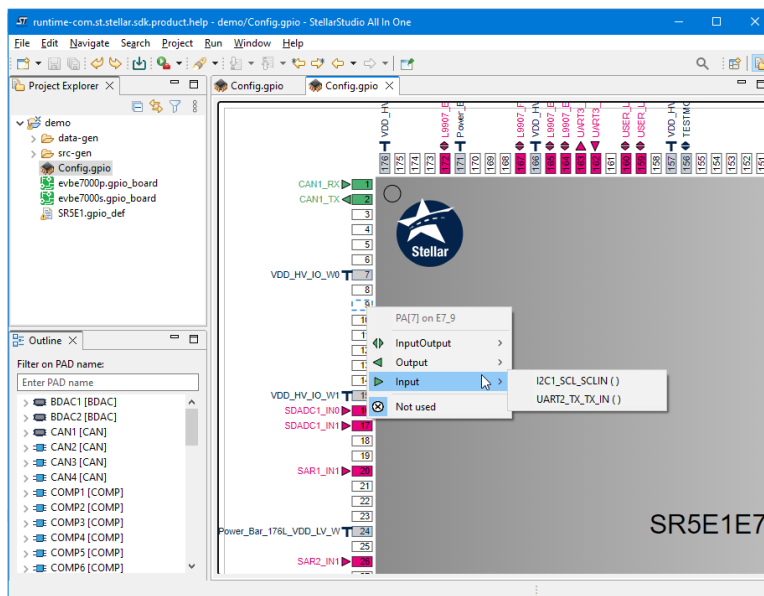


4.4 Assigning a function to a pin

Now that you have chosen the pin on which you would like to assign a function, you'll need to use the contextual menu to choose the proper function to be assigned.

The menu that will be shown contains some direction sub-menus, then the possible functions accessible from the selected pin.

Figure 46: Assign a function to a pin

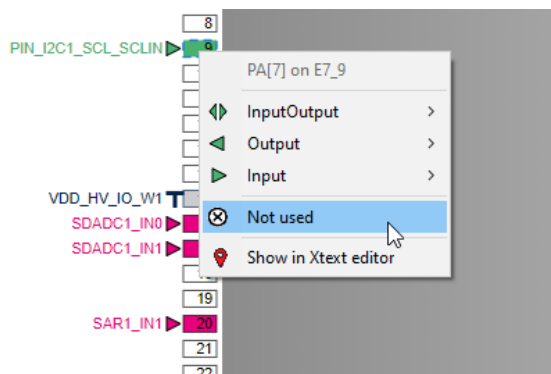


The result will show the assigned pin, with its default name.

If you want to change the pin configuration, you can choose another function, using again the same assignment operation.

Removing the assigned function is done with the same kind of operation, with the 'not used' option.

Figure 47: Unassign a function from a pin



Both textual editor and graphical editor are sharing the same resource (the .gpio file).

Please note that for the assigned function to be visible inside the textual editor, you will need to save the current graphical editor.

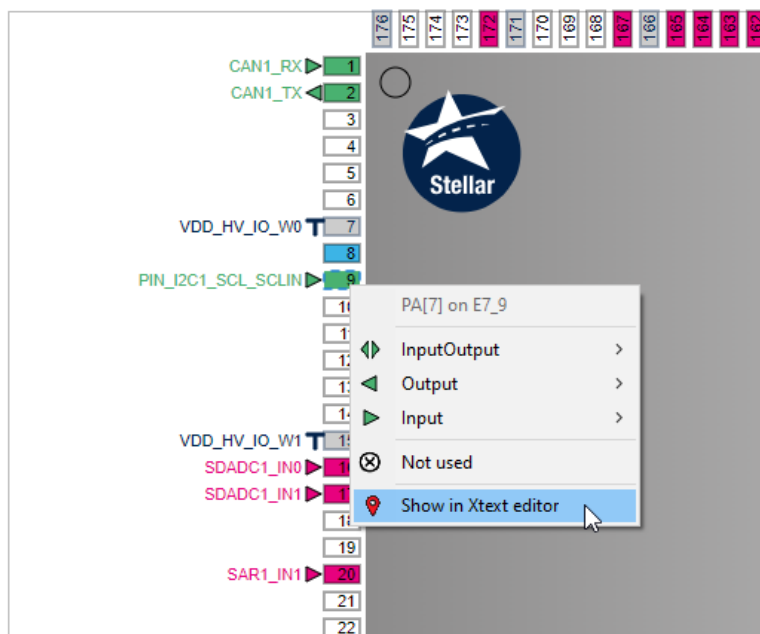
NB: Saving it (using the floppy disk from the toolbar icon) is sometimes not available. If this occurs, you'll need to click on the graphical editor tab in the editor area, then the save icon will be accessible.

4.5 Locate graphical pins in textual editor

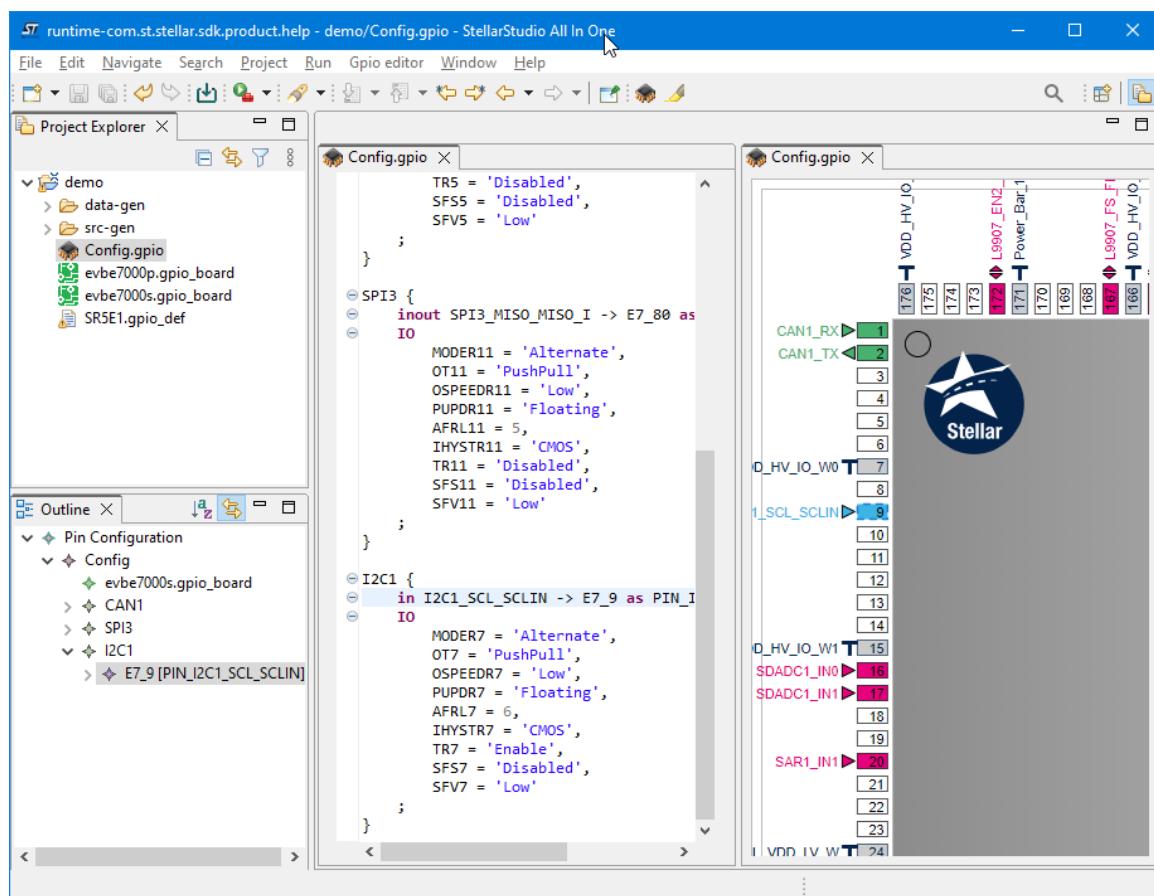
As the graphical configuration editor show a lot a pins, it is not obvious to see where is it present inside the textual editor.

For showing the textual pin configuration of a configured pin from the graphical editor, you must use the 'Show pin in Xtext editor' context menu.

Figure 48: Show pin in xtext editor



This will open the textual editor if not already opened, and will place the cursor location near the concerned pin configuration.



Now that the pin is located inside the textual editor, you can modify the register field instances of this configuration, or change its name, using the textual editor like described in the chapter named *manipulation of a pinmap configuration*.

A similar operation is done whenever you move the edition cursor inside the textual editor, if the graphical editor is opened as well (as in the picture above). The selected pin will be shown and centered in the graphical editor.

5 Integration

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

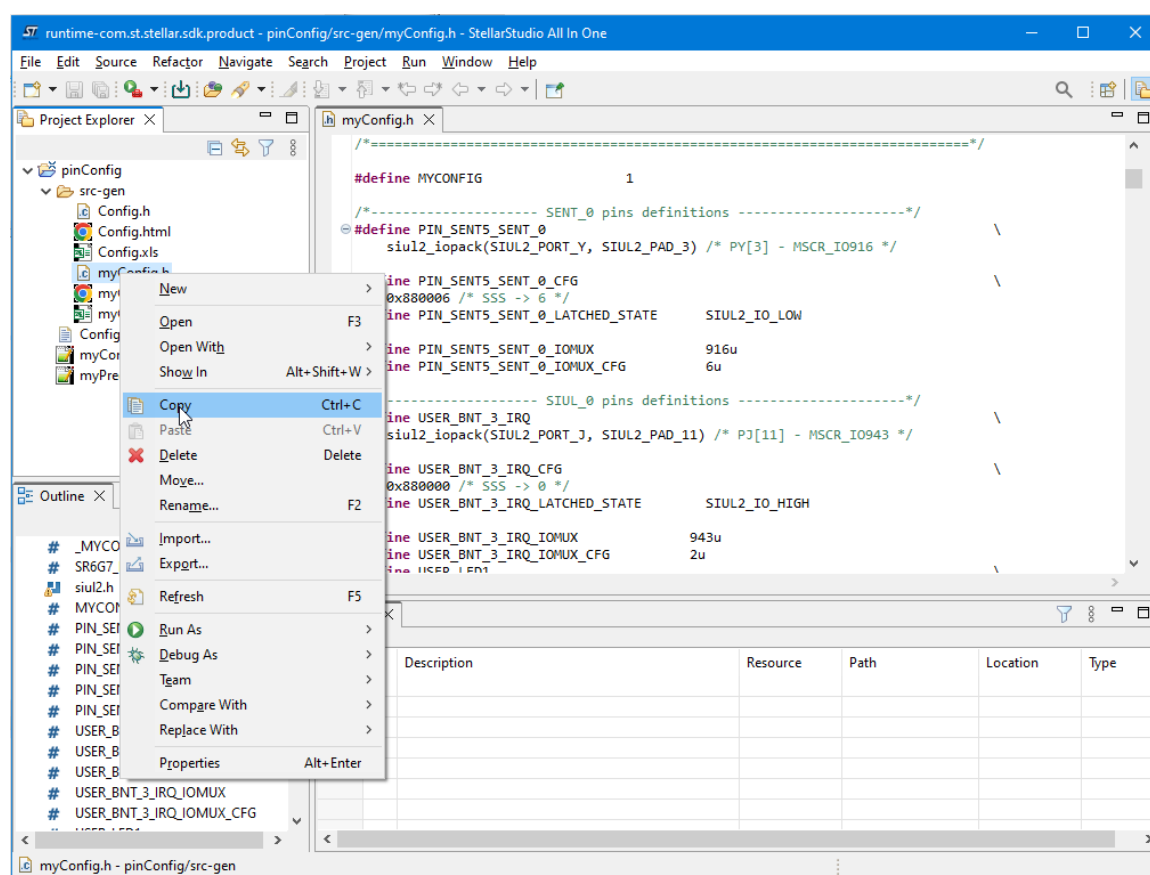
5.1 Integration of your generated file (C file)

Your generated file can be integrated from standalone product in any developer tools.

Select your C file

With contextual menu, copy your file

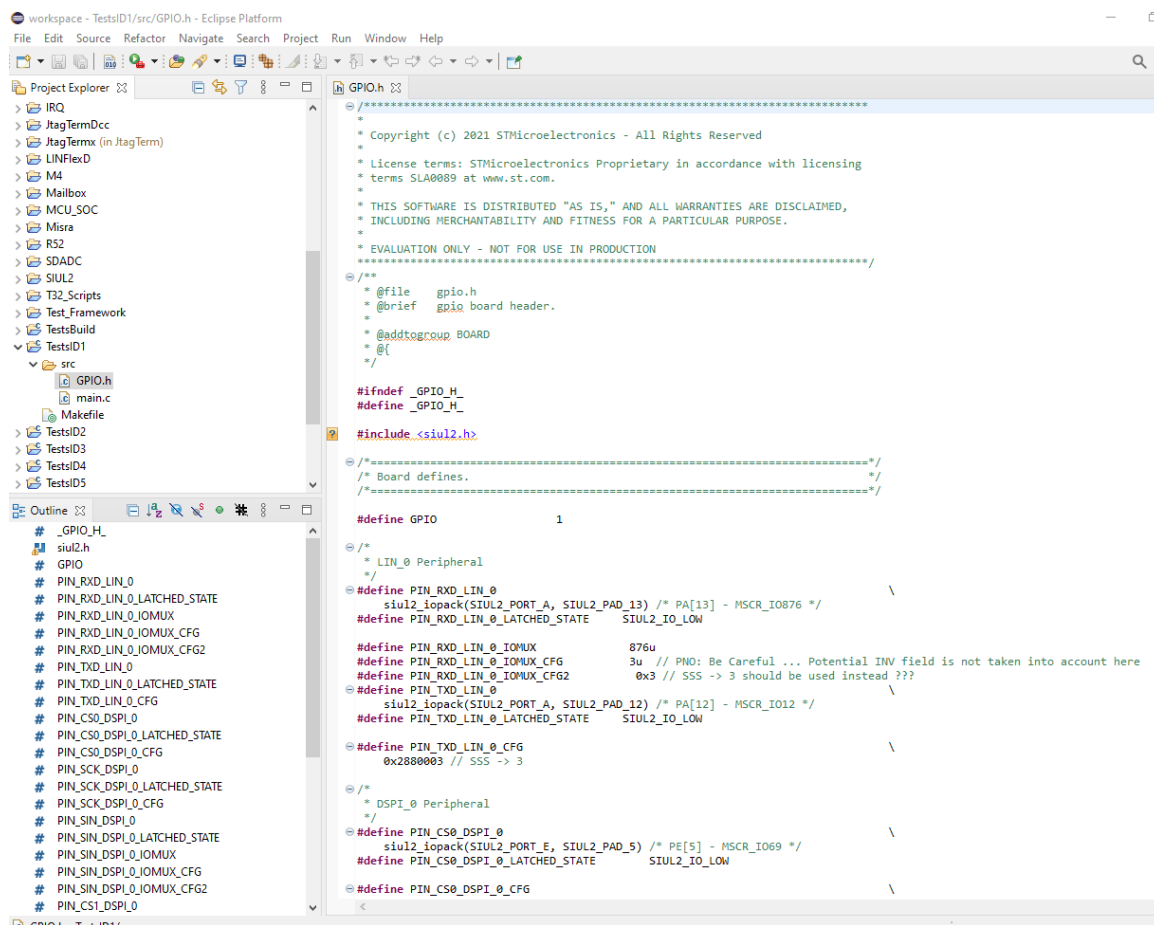
Figure 49: Copy your C file



Integrate it, in your favorite developer tool (Eclipse, Visual Studio Code and so on ..).

With contextual menu, paste your file.

Figure 50: Paste your C file in your favorite developer tool



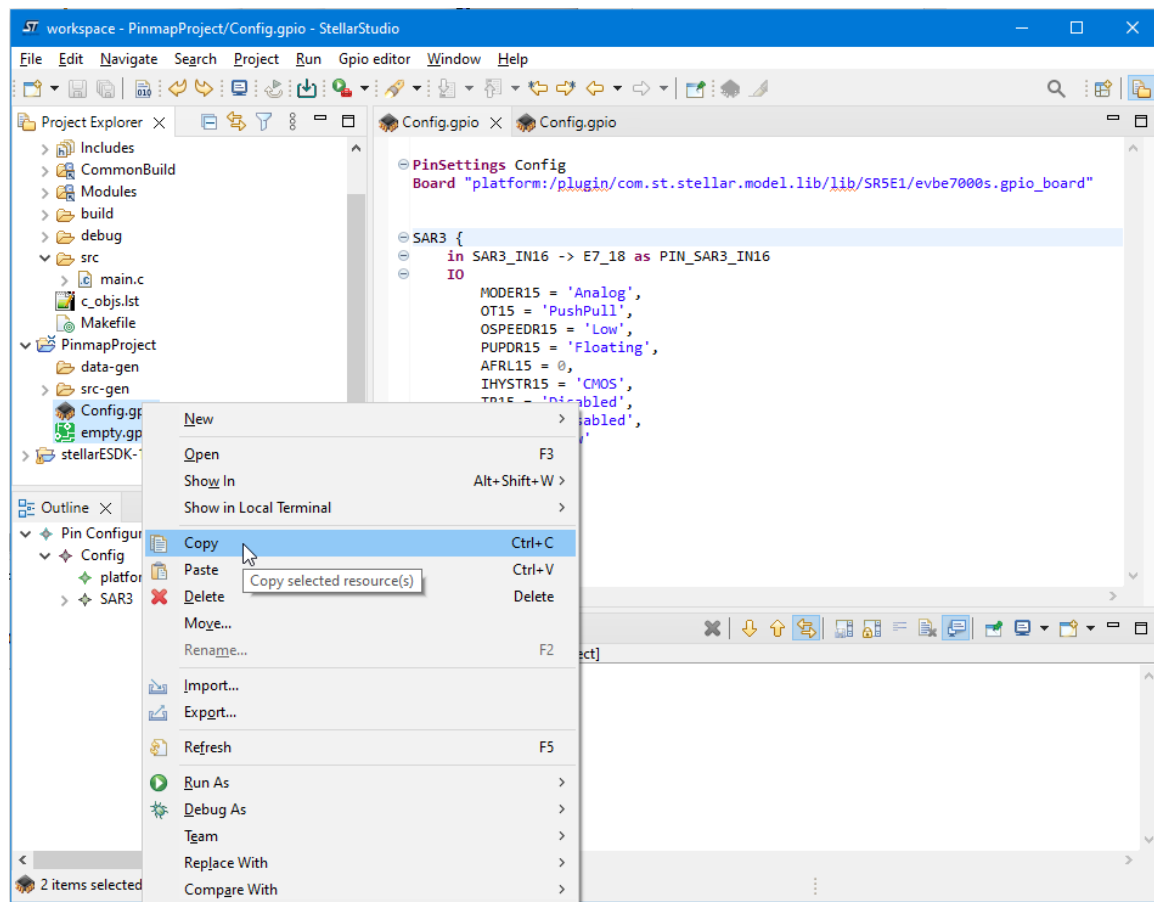
5.2 Integration of your gpio file

you can integrate your pinmap configuration in StellarStudio product

From StellarStudio pinmap configurator, select your *gpio* file

With contextual menu, copy your file

Figure 51: Copy your file

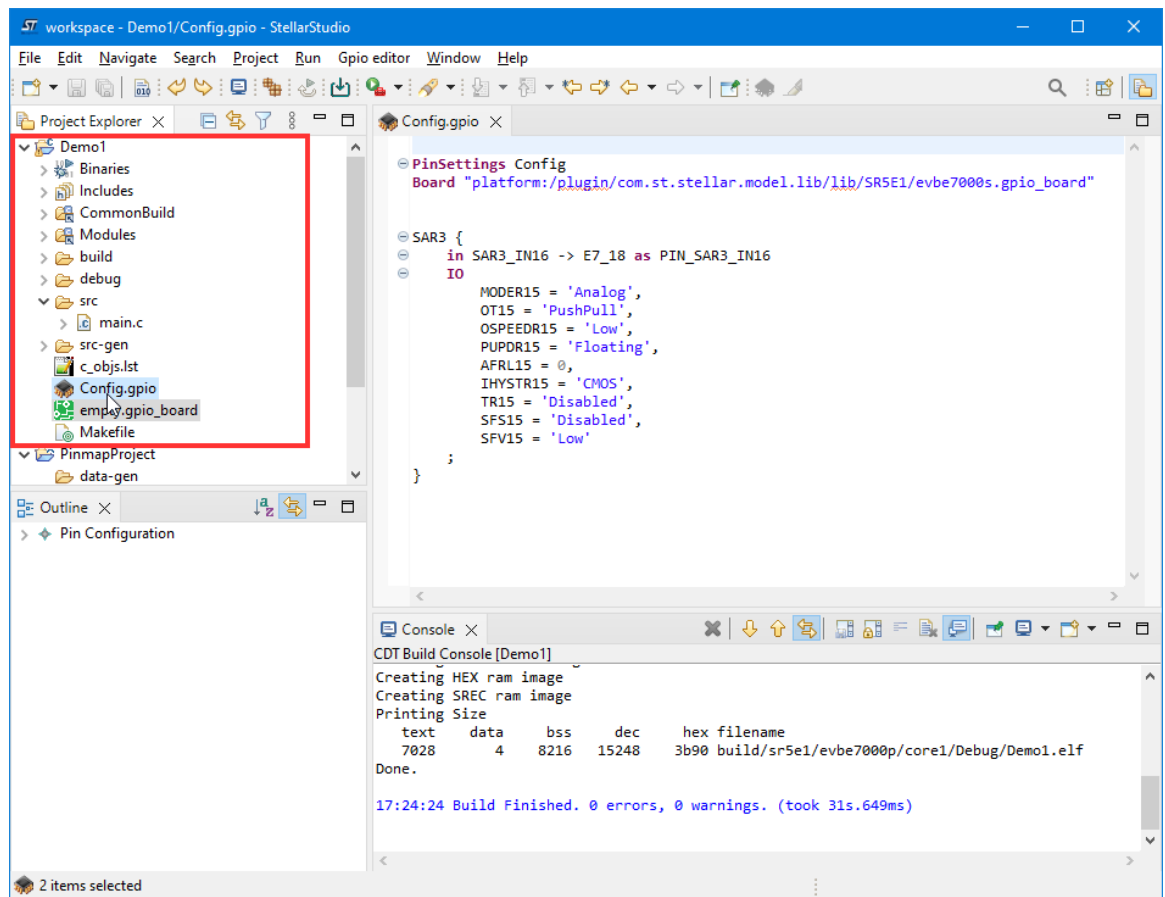


Integrate it, in StellarStudio product.

With contextual menu, paste your file in your favorite project.

after clicking yes on the conversion on a xtext project,

if your *gpio* is valid, generated files should be visible in *src-gen* folder

Figure 52: Paste your *gpio* file in StellarStudio product

5.3 Adapt project's Makefile

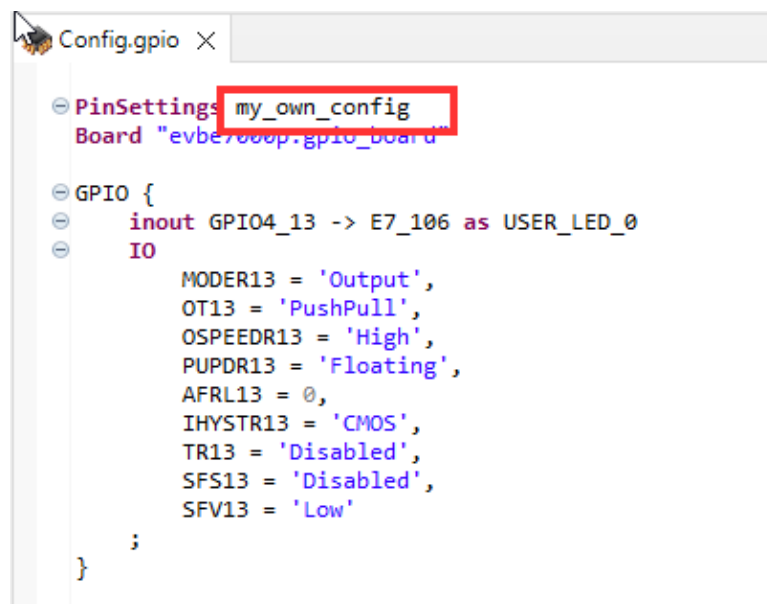
If you want your project to use the pinmap configuration that you've created, you'll need to adapt the Makefile to point to the correct generated configuration header file, generated from your *.gpio* file.

The default Makefile uses the board *evbe7000p* from the E-SDK, we need to replace this name with the name of the generated configuration header file.

The Makefile must contain the correct `CONFIG_BOARD` variable, and we need to adapt the `C_INCS`

1. `CONFIG_BOARD`: must contain the name of the configuration (i.e. the identifier directly following the PinSetting keyword in the *.gpio* file).

Figure 53: Name of the configuration to be used



2. C_INCS: must be augmented with the location of the folder containing the generated header file (i.e. src-gen project's folder).

Here are the two modifications, illustrated in the diff window below. The right part illustrates the default content of the Makefile, while the left part shows the modifications.

Figure 54: Adapt CONFIG_BOARD and C_INCS

```

Local: Makefile
#####
# Project makefile.
#####

# 'Stellar SDK' project values
PROJECTNAME := qsd
SDKID := StellarESDK-1.6.0

# Please Update it if you move your workspace
STELLAR_E_SDK_RELATIVE_PATH := ../../SDKS/$(SDKID)
TEST_ROOT_DIR := $(STELLAR_E_SDK_RELATIVE_PATH)
PROJECT_COMMON_DIR := $(STELLAR_E_SDK_RELATIVE_PATH)/Projects/

CONFIG_BOARD ?= my_own_config 1
include $(PROJECT_COMMON_DIR)/make/test_defs.mk

#####
# Project builds
#####
BUILD_OS_OSAL := 1

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

# Application name
APP_NAME := $(PROJECTNAME)

# C sources
C_SRCS += \
    src/main.c

# C includes
# PLEASE UPDATE IT FOR GENERATED CODE
C_INCS += \
    src-gen/ 2

#####
# Include 'Stellar SDK' top level makefile
#####
include $(STELLAR_E_SDK_BUILD_SYSTEM_DIR)/StellarESDK.mk

Local history: Makefile 30 janv. 2024, 17:15:50
#####
# Project makefile.
#####

# 'Stellar SDK' project values
PROJECTNAME := qsd
SDKID := StellarESDK-1.6.0

# Please Update it if you move your workspace
STELLAR_E_SDK_RELATIVE_PATH := ../../SDKS/$(SDKID)
TEST_ROOT_DIR := $(STELLAR_E_SDK_RELATIVE_PATH)
PROJECT_COMMON_DIR := $(STELLAR_E_SDK_RELATIVE_PATH)/Pr

include $(PROJECT_COMMON_DIR)/make/test_defs.mk

#####
# Project builds
#####
BUILD_OS_OSAL := 1

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

# Application name
APP_NAME := $(PROJECTNAME)

# C sources
C_SRCS += \
    src/main.c

# C includes
# PLEASE UPDATE IT FOR GENERATED CODE
C_INCS += \
    src-gen/ \
    src-gen/<device-name>

#####
# Include 'Stellar SDK' top level makefile
#####
include $(STELLAR_E_SDK_BUILD_SYSTEM_DIR)/StellarESDK.m

```

Now that the Makefile is updated, you can compile your project, which now points to the good configuration.

6 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