



# SR6x StellarStudio Pinmap Configurator

---

## SR6x StellarStudio Pinmap Configurator User Guide

---

# Contents

<b>1 Introduction .....</b>	<b>5</b>
<b>2 Creation of a pin configuration .....</b>	<b>7</b>
2.1 Create a project if needed .....	7
2.2 Create configuration file .....	9
2.3 Modifying a board's predefined pins .....	16
<b>3 Manipulation of a pinmap configuration .....</b>	<b>19</b>
3.1 Magic key .....	19
3.2 Manipulation of the configuration file .....	19
3.3 Code generation .....	23
3.4 HTML generation .....	24
3.5 XLS generation .....	27
3.6 Configuration errors .....	29
3.7 Application migration .....	30
3.8 Package swapping .....	33
3.9 Outline .....	34
3.10 Pin configuration formatting .....	36
<b>4 Graphical configuration of pins .....</b>	<b>38</b>
4.1 Graphical configuration .....	38
4.2 Open the graphical configuration editor .....	38
4.3 Using pinmap configuration outline views .....	40
4.4 Assigning a function to a pin .....	43
4.5 Locate graphical pins in textual editor .....	45
<b>5 Integration .....</b>	<b>47</b>
5.1 Integration of your generated file (C file) .....	47
5.2 Integration of your PinCfg file .....	48
5.3 Adapt project's Makefile .....	50
<b>6 Disclaimer .....</b>	<b>54</b>

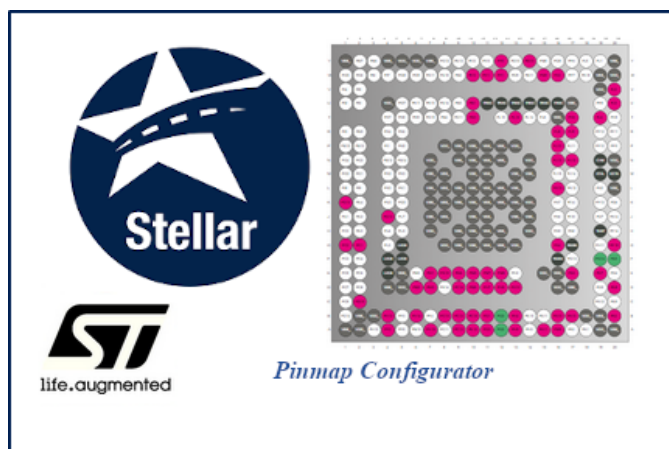
# List of figures

Figure 1. StellarStudio Pinmap configurator - Copyright STMicroelectronics (c) 2021-2025 .....	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 configurator wizard .....	10
Figure 7. Choose predefined board .....	11
Figure 8. Start a configuration from a board .....	14
Figure 9. Copy to current project .....	17
Figure 10. Changed imported board path .....	18
Figure 11. Choose a peripheral .....	20
Figure 12. Choose a pin .....	20
Figure 13. Pin is chosen .....	21
Figure 14. Add some additional pins .....	21
Figure 15. Result of one peripheral configuration .....	21
Figure 16. example of a configuration with several peripherals .....	22
Figure 17. Set an IO configuration for your pin settings .....	23
Figure 18. example of a IO configured to a pin setting .....	23
Figure 19. Code generated from the configuration .....	24
Figure 20. Open an HTML file .....	25
Figure 21. HTML generated from the configuration .....	26
Figure 22. Auto-refresh local changes .....	27
Figure 23. Open an XLS file .....	28
Figure 24. Example of XLS generated from the configuration .....	29
Figure 25. Syntax Errors .....	30
Figure 26. Example of semantic errors and quickfixes .....	30
Figure 27. Old applications Migration .....	31
Figure 28. Launching migration .....	32
Figure 29. Confirm migration .....	32
Figure 30. Migration result .....	33
Figure 31. Unneeded migration .....	33
Figure 32. Package swapping .....	34
Figure 33. Outline for pin configuration file .....	35
Figure 34. Outline for a C generated file .....	36
Figure 35. Example before formatting .....	37
Figure 36. Example after formatting .....	37
Figure 37. Open the graphical editor from project explorer .....	39
Figure 38. Open the graphical editor from Gpio editor menu .....	39
Figure 39. Open the graphical editor from the MCU icon in the toolbar .....	40
Figure 40. Result: opened graphical editor .....	40
Figure 41. Example of selection a peripheral .....	41
Figure 42. Example of selection a function .....	42
Figure 43. Example of searching of a PAD .....	43
Figure 44. Assign a function to a pin .....	44
Figure 45. Unassign a function from a pin .....	44
Figure 46. Show pin in xtext editor .....	45
Figure 47. Copy your C file .....	47

Figure 48. Paste your C file in your favorite developer tool .....	48
Figure 49. Copy your file .....	49
Figure 50. Paste your pincfg file in StellarStudio all-in-one product .....	50
Figure 51. Name of the configuration to be used .....	51
Figure 52. Adapt CONFIG_DEVICE, CONFIG_BOARD and C_INCS .....	52

# 1 Introduction

Figure 1: StellarStudio Pinmap configurator - Copyright STMicroelectronics (c) 2021-2025



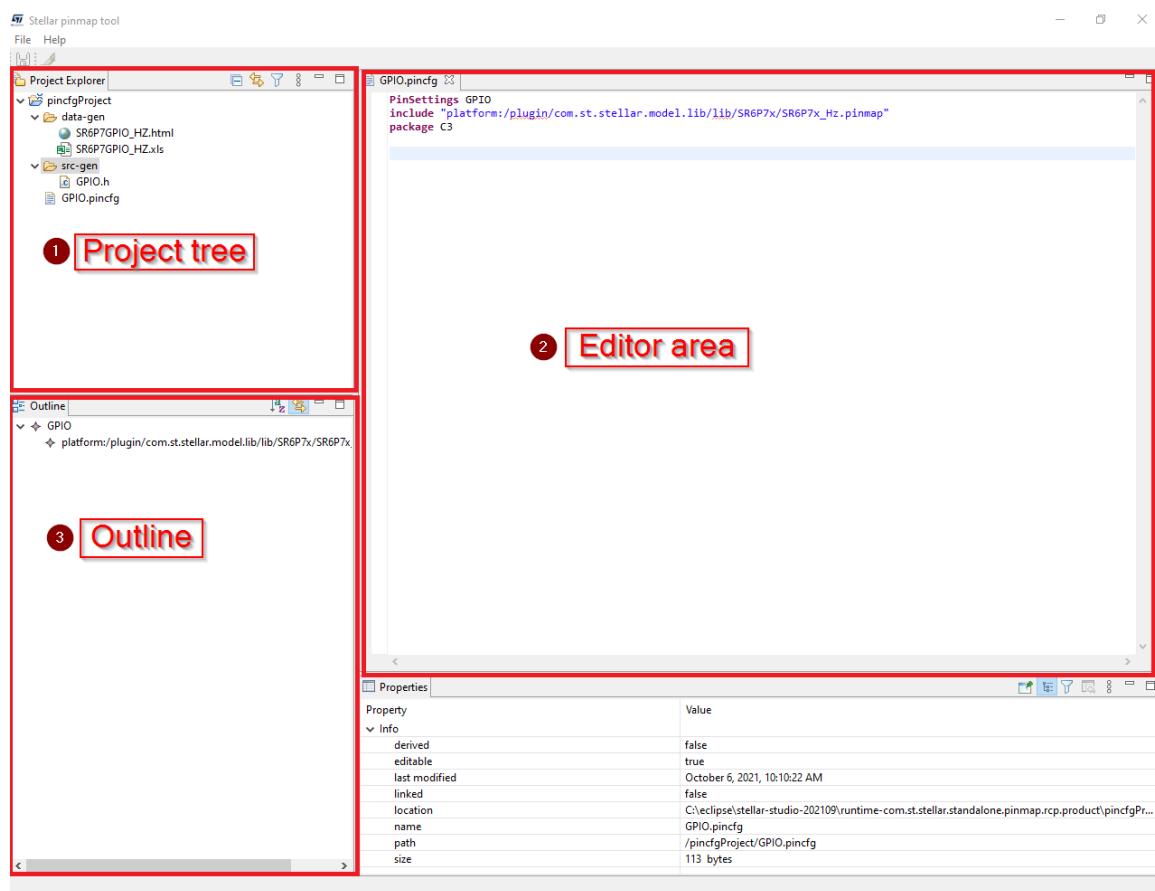
The Pinmap configurator is an intuitive end user graphical assistant dedicated to configure the pinmap of Stellar products.

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

## 2 Creation of a pin configuration

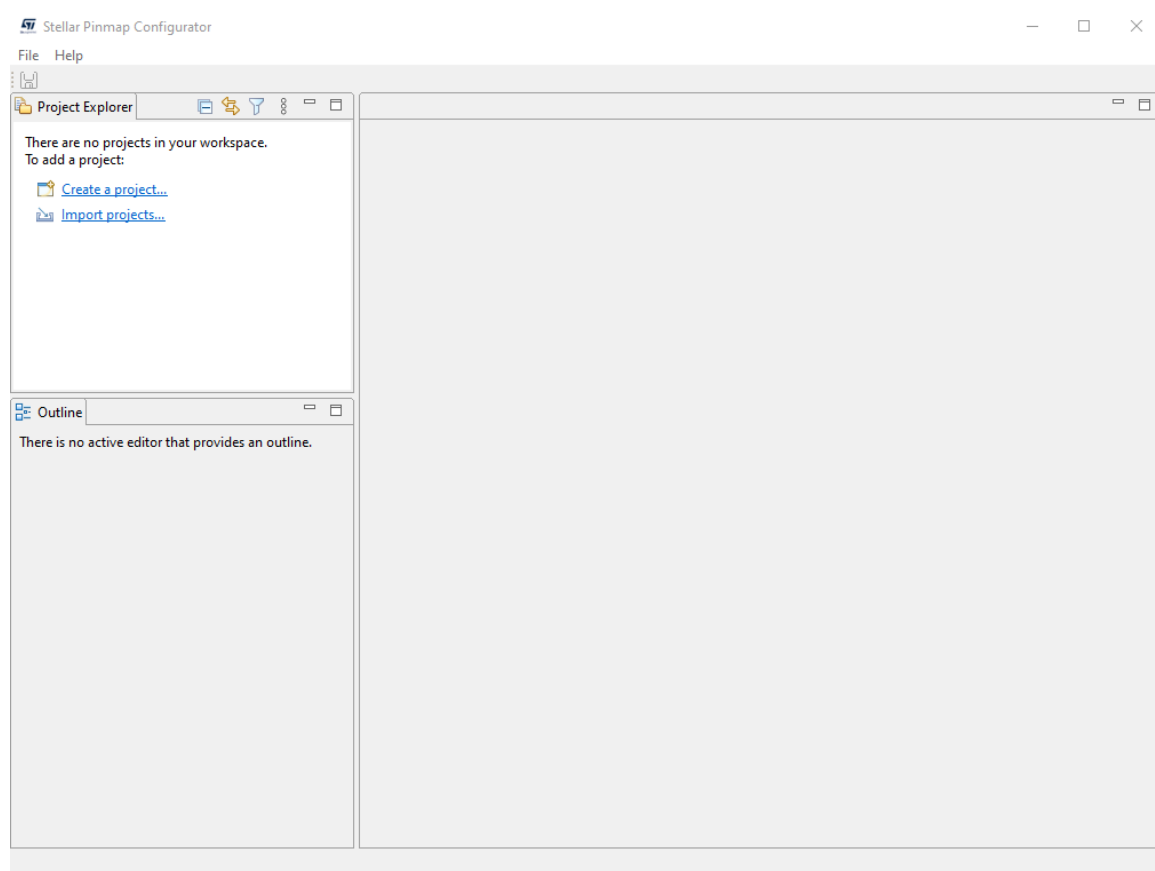
A pin configuration is represented by a configuration file, with the ".pincfg" extension. This file must be present in a project, which hosts it and allows for its manipulation.

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

### 2.1 Create a project if needed

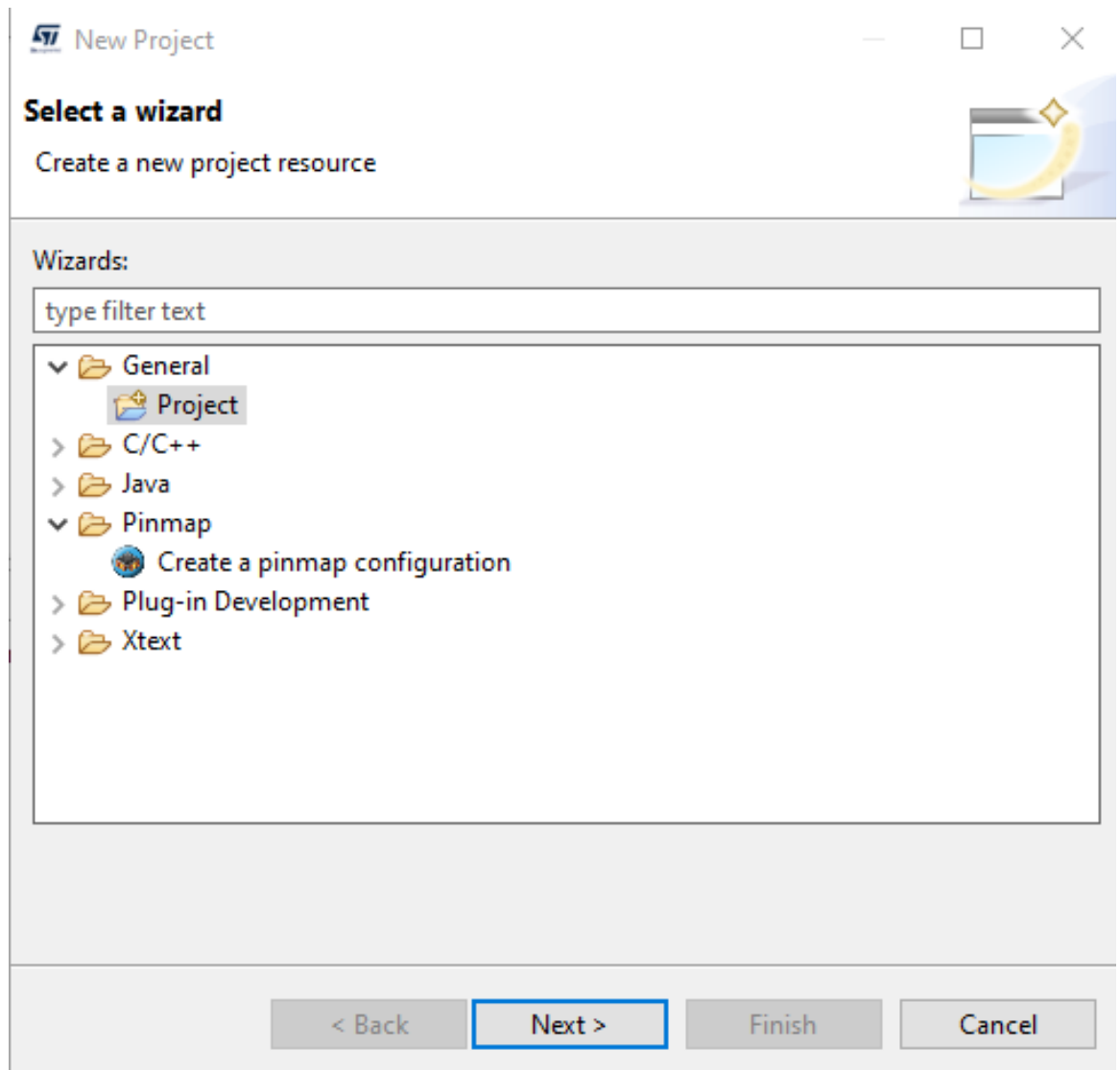
The pin configuration is hosted in a file, that needs a project as container.

**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:

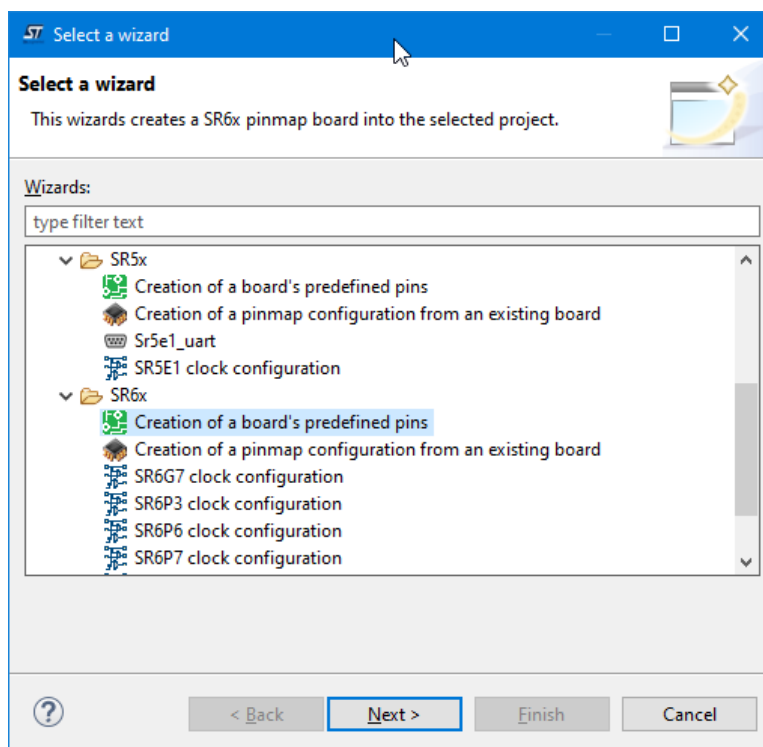
## 2.2 Create configuration file

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

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

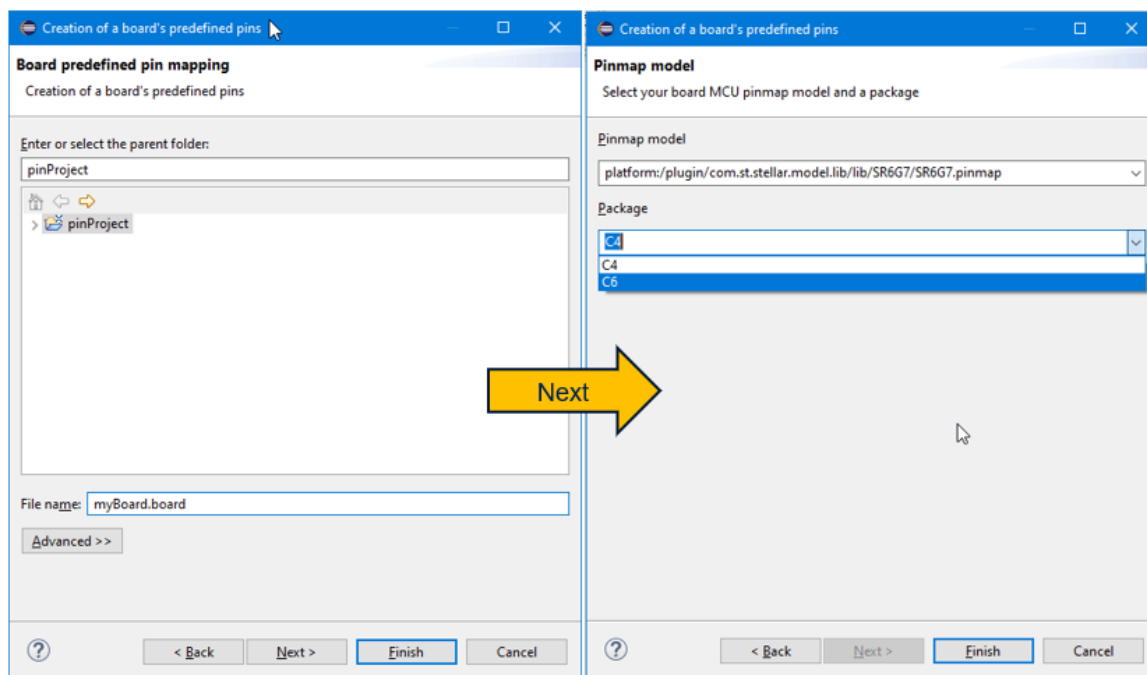
This is done using the same key combination: <CTRL>N, then select the pinmap configurator creation wizard.

Figure 6: Launch pinmap configurator wizard



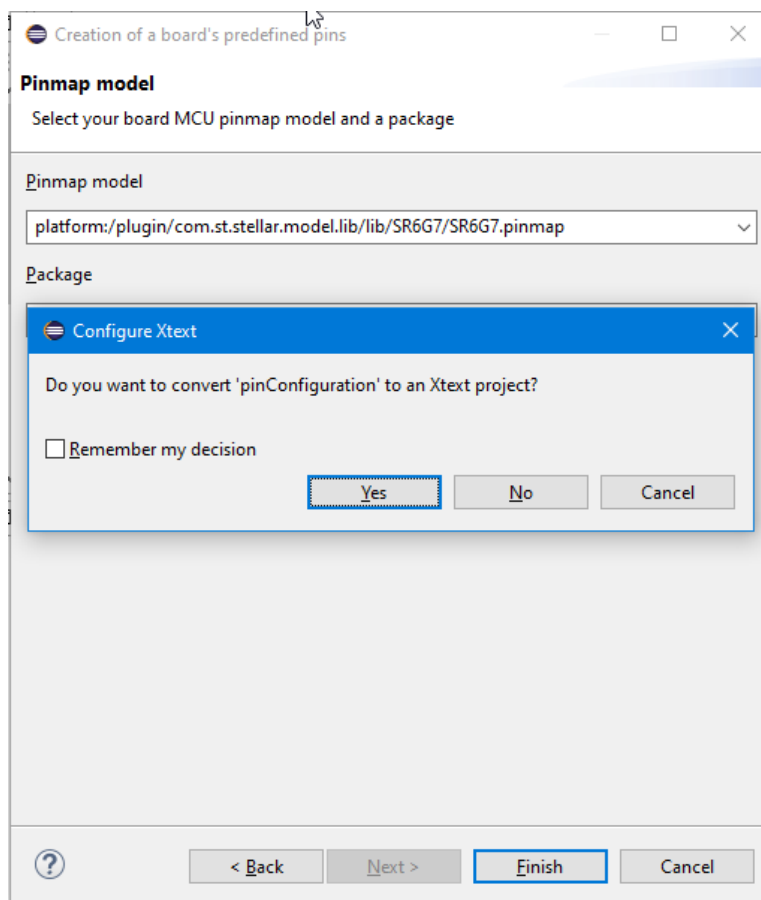
1. "Creation of a board's predefined pins" if you want to create your own board pins definitions. You will be asked to enter a file name (with extension set to .board), then choosing a MCU pinmap model and a package.
2. "Creation of a pinmap configuration from an existing board" if you want to configure an already existing board. You will be asked to enter a file name (with extension set to .pincfg), then choose the board you want to use, from the list of predefined boards.

Figure 7: Choose predefined board



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

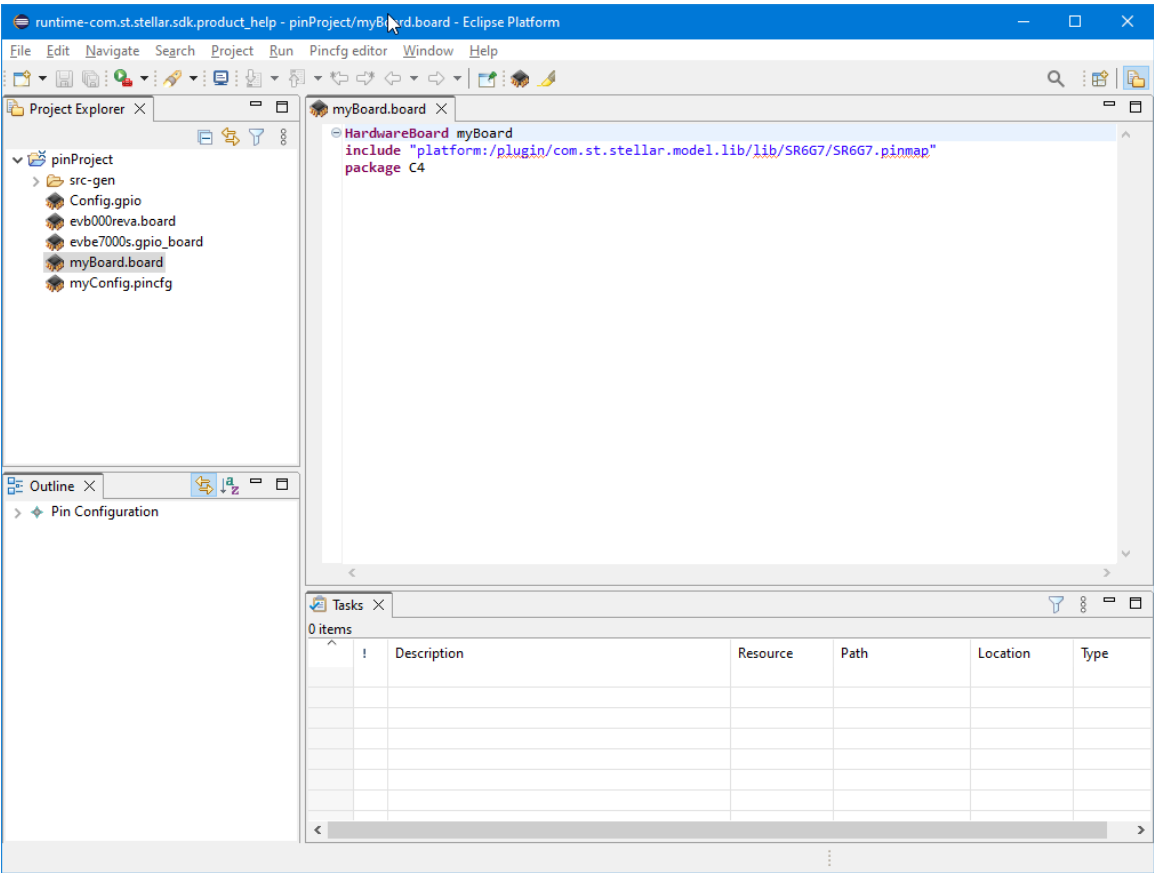
Then choose the pinmap model and the package. The pinmap model is provided by the StellarStudio team and contains the needed pinmap definitions that will be used for a particular Stellar platform. This model file contains some package definitions. You must choose one of them to start your configuration.



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. It allows for launching the appropriate editor when double-clicking on the configuration file.

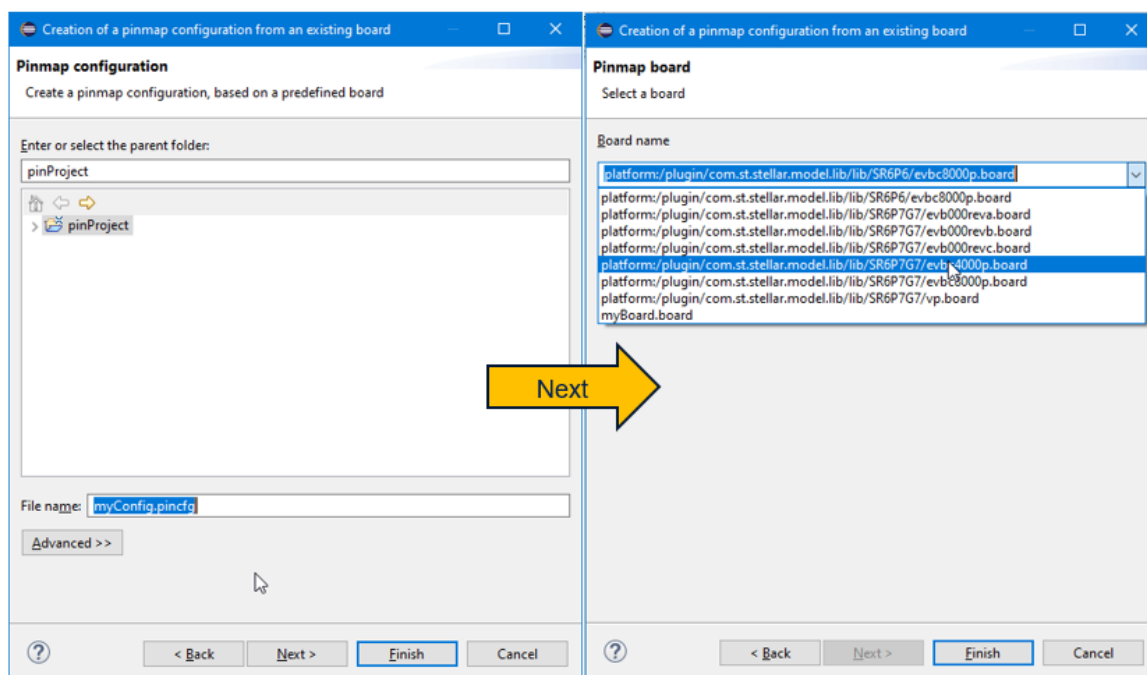
Now your project is ready and open in the editor area of the tool.



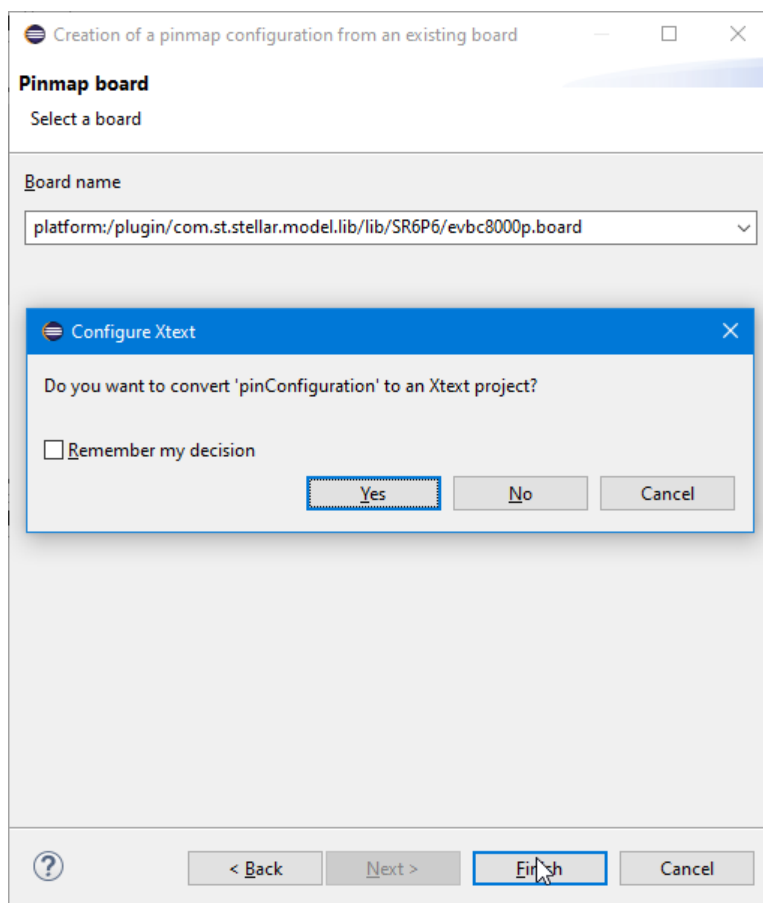
Please note that the file name must have the extension ".board". This file would be part of the available boards you can choose from when creating your pinmap configuration.

Second option for a pinmap wizard is to create a configuration from a board that already exists.

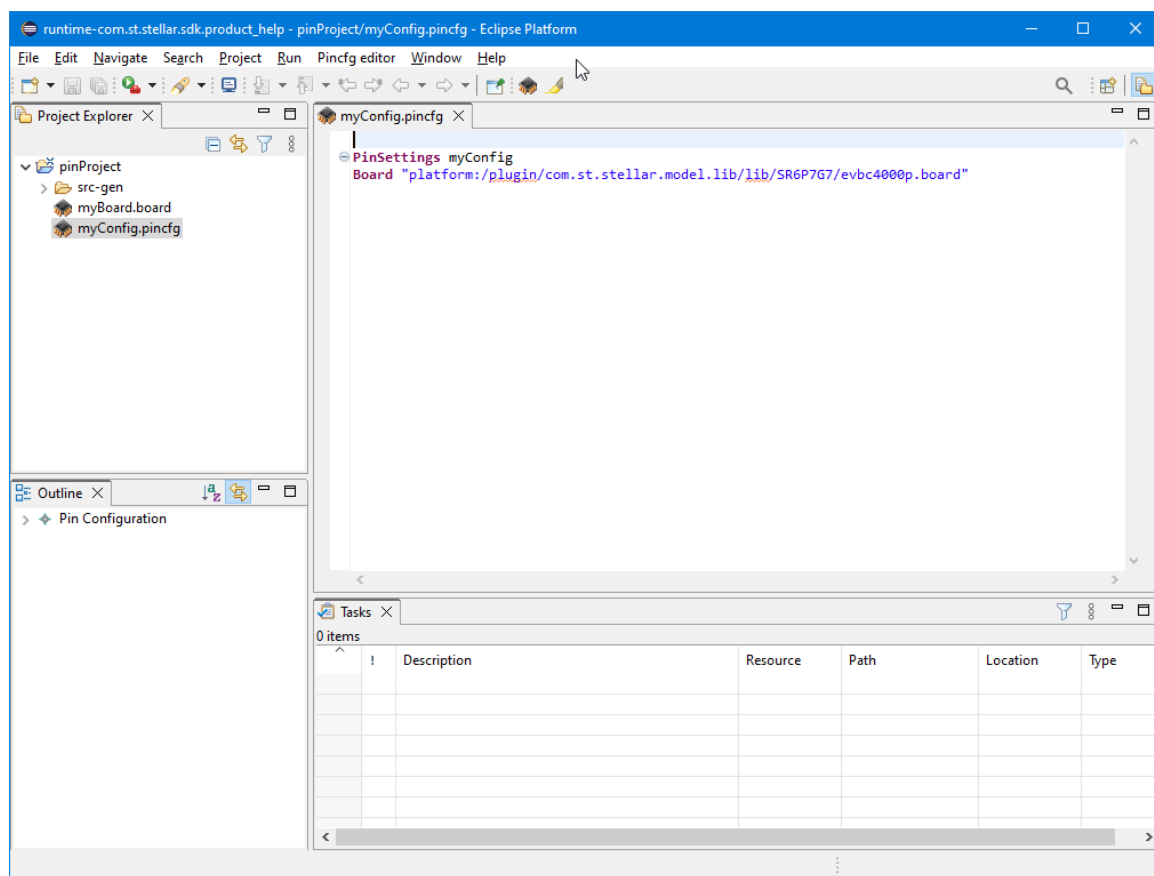
Figure 8: Start a configuration from a board



Choose the board to start from. Again if the project has not yet been converted to Xtext you must do it. This step is mandatory, as the configuration file is based on the Xtext technology, declaring the project as an Xtext project. It allows for launching the appropriate editor when double-clicking on the configuration file.



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



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

## 2.3 Modifying a board's predefined pins

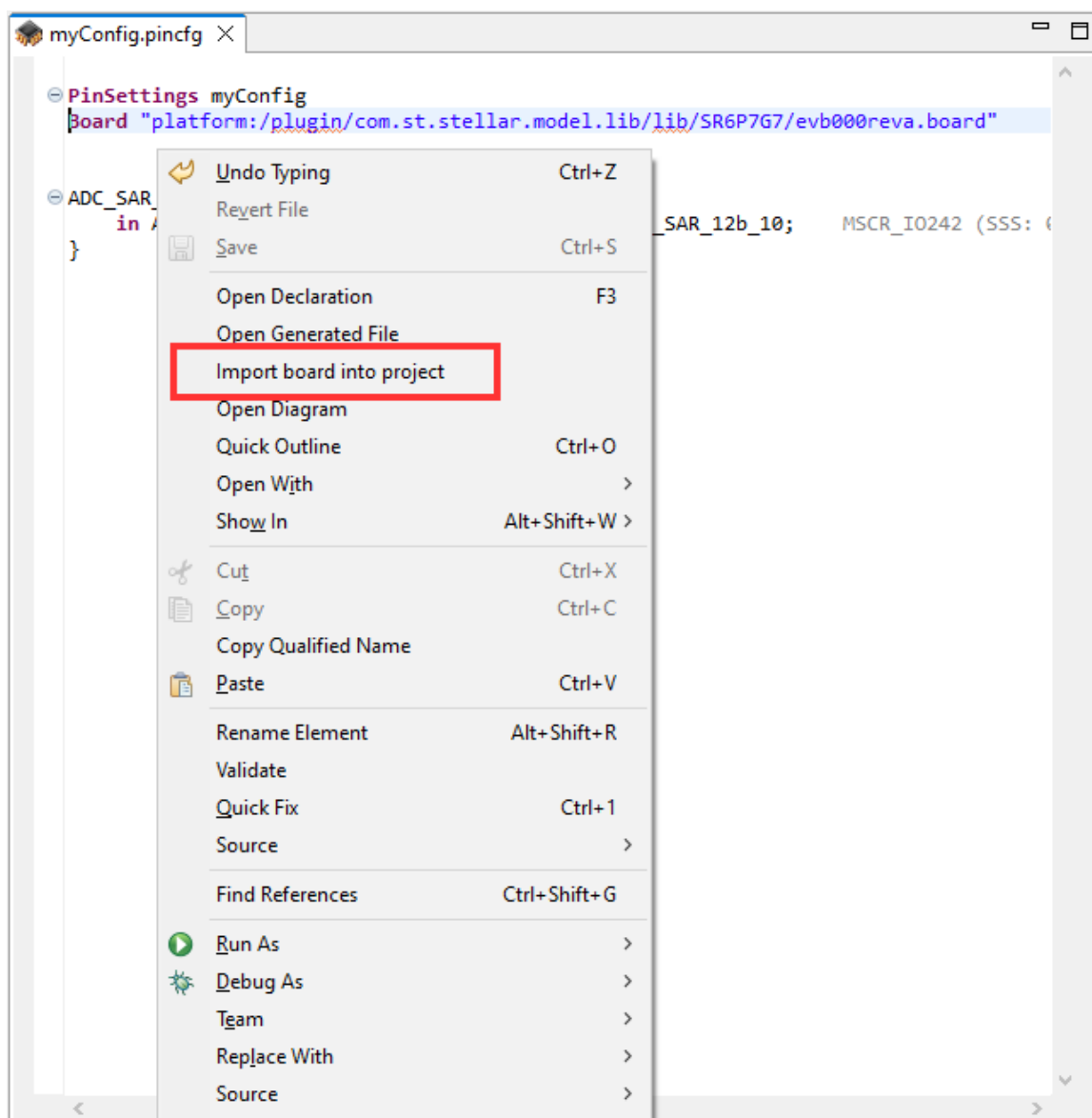
The previous section allows you to start a pin configuration from a predefined board. A predefined board, as provided to you by the StellarStudio team, allows to use predefined pins, but contains pin configurations that you cannot modify. Whenever you need to modify or adapt a pin already configured 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 file, you can right click on the editor text and choose the "Import board into project" menu option.



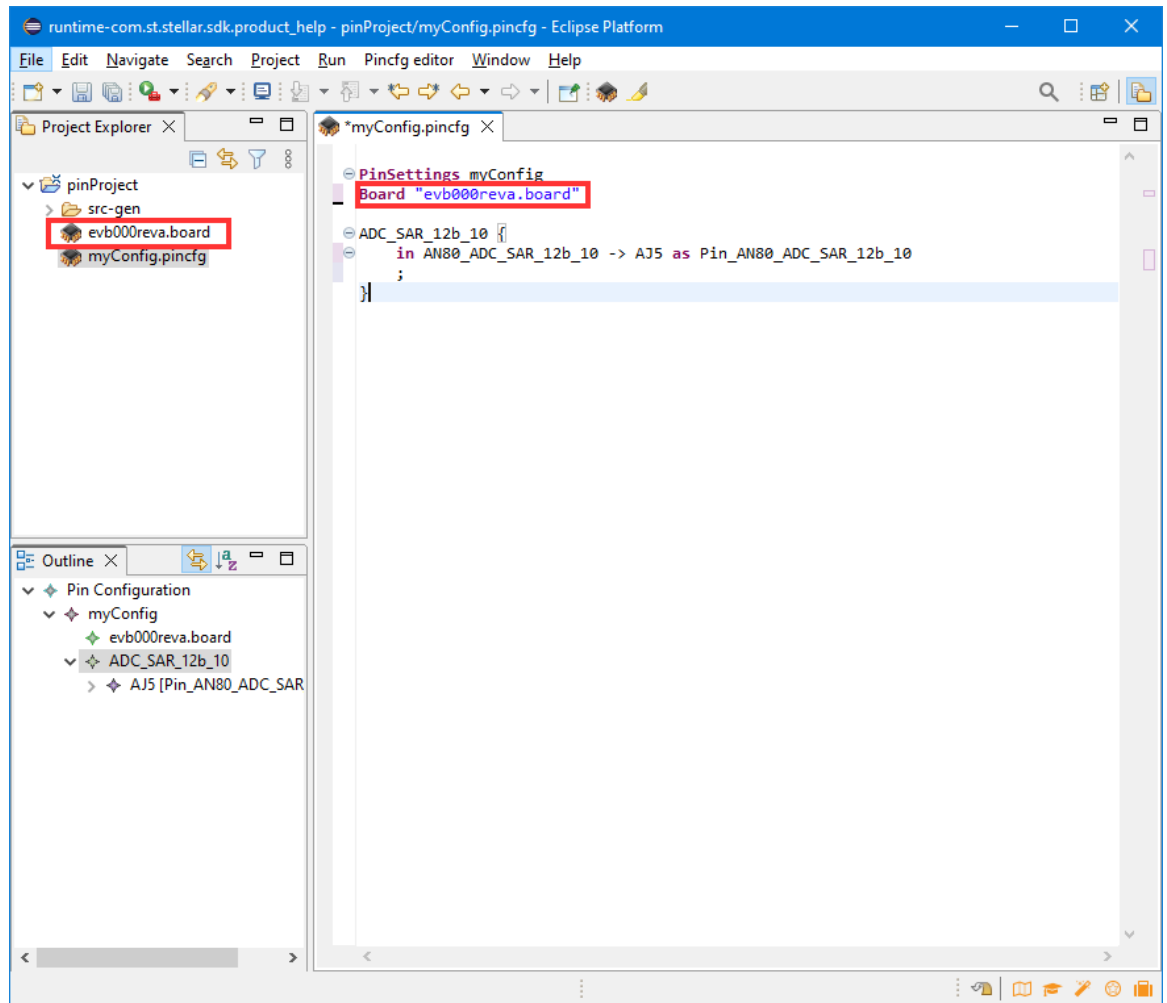
Figure 9: Copy to current project



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

Now that you have created your configuration file, 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>
- 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 file

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.

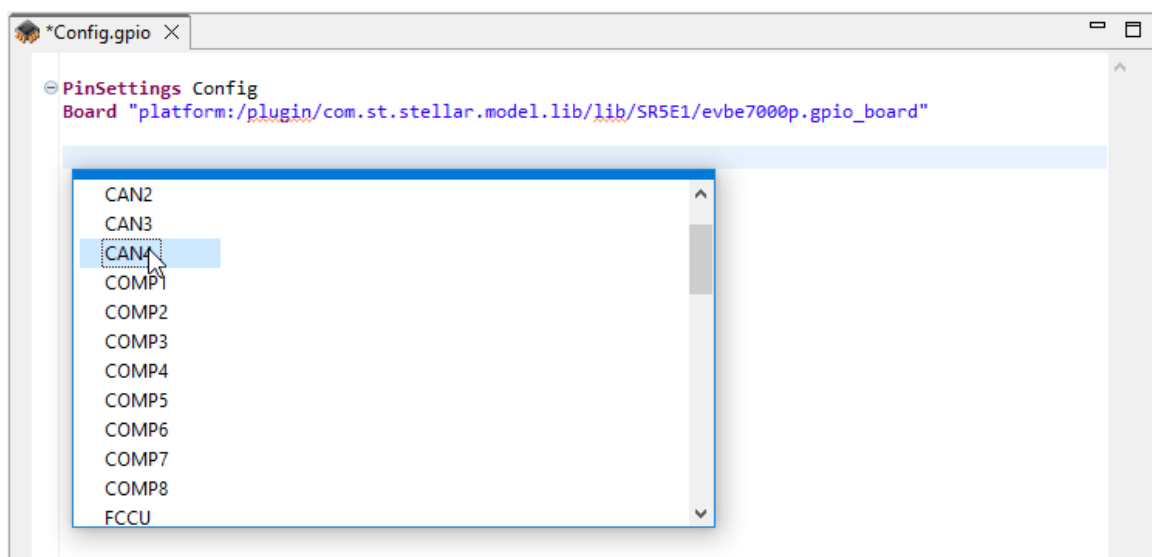
The "include" keyword introduces the URI to the pinmap model... This file is provided by the "StellarStudio" team.

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

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

you can add a comment with your peripheral

Figure 11: Choose a peripheral



open the bracket,

Now choose a pin to configure,

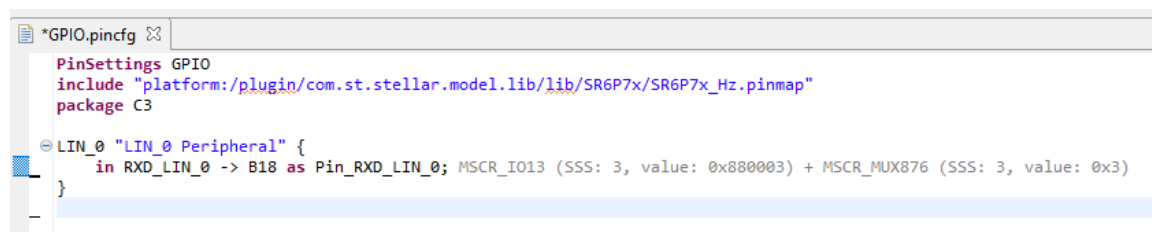
Figure 12: Choose a pin



The developer has to name the pin (NB: the pin is given a default name that the user can change).

MSCR IO value and MSCR Mux are automatically set and put in comment like a developer information.

**Figure 13: Pin is chosen**



```
*GPIO.pincfg
PinSettings GPIO
include "platform:/plugin/com.st.stellar.model.lib/lib/SR6P7x/SR6P7x_Hz.pinmap"
package C3

LIN_0 "LIN_0 Peripheral" {
  in RXD_LIN_0 -> B18 as Pin_RXD_LIN_0; MSCR_I013 (SSS: 3, value: 0x880003) + MSCR_MUX876 (SSS: 3, value: 0x3)
}
```

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

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

**Figure 14: Add some additional pins**



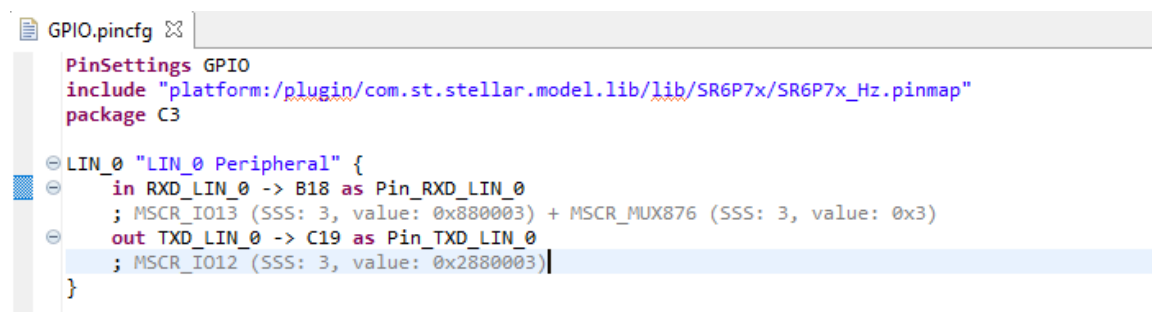
```
*GPIO.pincfg
PinSettings GPIO
include "platform:/plugin/com.st.stellar.model.lib/lib/SR6P7x/SR6P7x_Hz.pinmap"
package C3

LIN_0 "LIN_0 Peripheral" {
  in RXD_LIN_0 -> B18 as Pin_RXD_LIN_0; MSCR_I013 (SSS: 3, value: 0x880003) + MSCR_MUX876 (SSS: 3, value: 0x3)
}
```

- out TXD\_LIN\_0 on pin B18 (pad: PA13)
- out TXD\_LIN\_0 on pin C19 (pad: PA12)
- out TXD\_LIN\_0 on pin D15 (pad: PH10)
- out TXD\_LIN\_0 on pin E17 (pad: PA11)
- out TXD\_LIN\_0 on pin G17 (pad: PA10)
- out TXD\_LIN\_0 on pin Y13 (pad: PB10)

which will result in the following configuration:

**Figure 15: Result of one peripheral configuration**



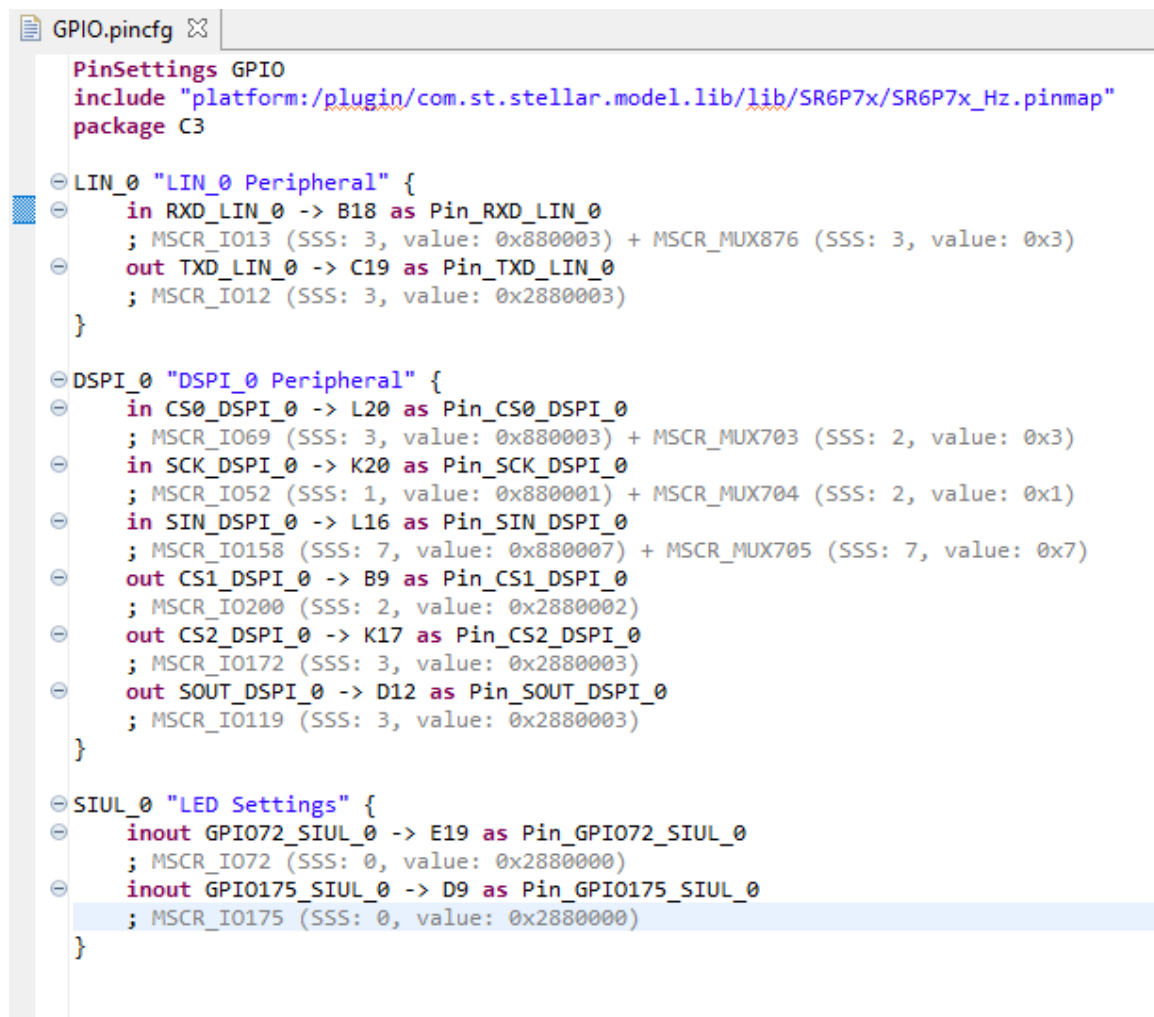
```
GPIO.pincfg
PinSettings GPIO
include "platform:/plugin/com.st.stellar.model.lib/lib/SR6P7x/SR6P7x_Hz.pinmap"
package C3

LIN_0 "LIN_0 Peripheral" {
  in RXD_LIN_0 -> B18 as Pin_RXD_LIN_0
  ; MSCR_I013 (SSS: 3, value: 0x880003) + MSCR_MUX876 (SSS: 3, value: 0x3)
  out TXD_LIN_0 -> C19 as Pin_TXD_LIN_0
  ; MSCR_I012 (SSS: 3, value: 0x2880003)
}
```

Of course, it is possible to add some new peripherals.

The result could be:

**Figure 16: example of a configuration with several peripherals**



```

GPIO.pincfg
PinSettings GPIO
include "platform:/plugin/com.st.stellar.model.lib/lib/SR6P7x/SR6P7x_Hz.pinmap"
package C3

LIN_0 "LIN_0 Peripheral" {
  in RXD_LIN_0 -> B18 as Pin_RXD_LIN_0
  ; MSCR_IO13 (SSS: 3, value: 0x880003) + MSCR_MUX876 (SSS: 3, value: 0x3)
  out TXD_LIN_0 -> C19 as Pin_TXD_LIN_0
  ; MSCR_IO12 (SSS: 3, value: 0x2880003)
}

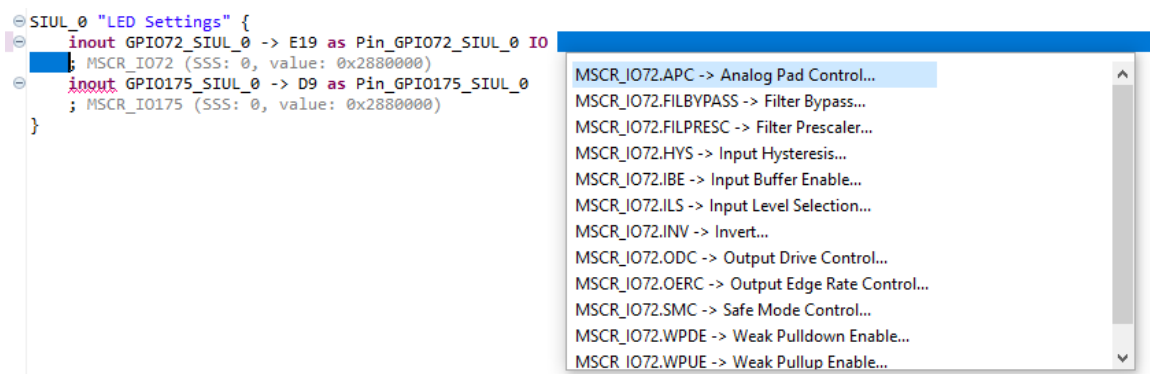
DSPI_0 "DSPI_0 Peripheral" {
  in CS0_DSPI_0 -> L20 as Pin_CS0_DSPI_0
  ; MSCR_IO69 (SSS: 3, value: 0x880003) + MSCR_MUX703 (SSS: 2, value: 0x3)
  in SCK_DSPI_0 -> K20 as Pin_SCK_DSPI_0
  ; MSCR_IO52 (SSS: 1, value: 0x880001) + MSCR_MUX704 (SSS: 2, value: 0x1)
  in SIN_DSPI_0 -> L16 as Pin_SIN_DSPI_0
  ; MSCR_IO158 (SSS: 7, value: 0x880007) + MSCR_MUX705 (SSS: 7, value: 0x7)
  out CS1_DSPI_0 -> B9 as Pin_CS1_DSPI_0
  ; MSCR_IO200 (SSS: 2, value: 0x2880002)
  out CS2_DSPI_0 -> K17 as Pin_CS2_DSPI_0
  ; MSCR_IO172 (SSS: 3, value: 0x2880003)
  out SOUT_DSPI_0 -> D12 as Pin_SOUT_DSPI_0
  ; MSCR_IO119 (SSS: 3, value: 0x2880003)
}

SIUL_0 "LED Settings" {
  inout GPIO72_SIUL_0 -> E19 as Pin_GPIO72_SIUL_0
  ; MSCR_IO72 (SSS: 0, value: 0x2880000)
  inout GPIO175_SIUL_0 -> D9 as Pin_GPIO175_SIUL_0
  ; MSCR_IO175 (SSS: 0, value: 0x2880000)
}

```

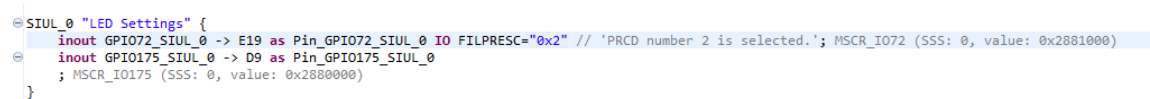
It is possible to update MSCR IO and MSCR MUX after the pin name.

Figure 17: Set an IO configuration for your pin settings



here the result will be:

Figure 18: example of a IO configured to a pin setting



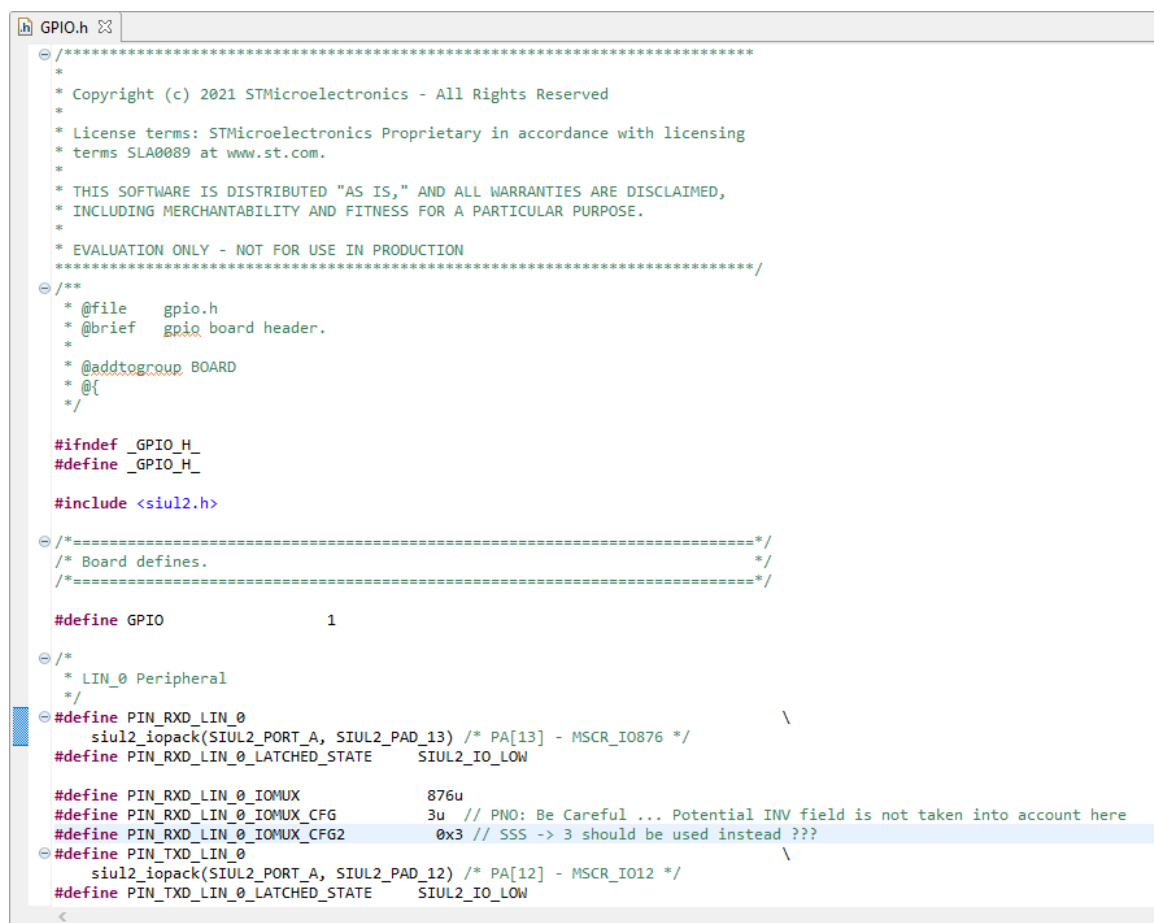
You can reproduce this procedure for all the needed configurations.

### 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 file is saved.

Figure 19: Code generated from the configuration



```

GPIO.h
*****
* Copyright (c) 2021 STMicroelectronics - All Rights Reserved
* License terms: STMicroelectronics Proprietary in accordance with licensing
* terms SLA0089 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
*****/

/**
 * @file    gpio.h
 * @brief   gpio board header.
 *
 * @addtogroup BOARD
 * @{
 */

#ifndef _GPIO_H_
#define _GPIO_H_

#include <siul2.h>

/*=====*/
/* Board defines. */
/*=====*/

#define GPIO 1

/*
 * LIN_0 Peripheral
 */
#define PIN_RXD_LIN_0 \
    siul2_iopack(SIUL2_PORT_A, SIUL2_PAD_13) /* PA[13] - MSCR_IO876 */
#define PIN_RXD_LIN_0_LATCHED_STATE SIUL2_IO_LOW

#define PIN_RXD_LIN_0_IOMUX 876u
#define PIN_RXD_LIN_0_IOMUX_CFG 3u // PNO: Be Careful ... Potential INV field is not taken into account here
#define PIN_RXD_LIN_0_IOMUX_CFG2 0x3 // SSS -> 3 should be used instead ???
#define PIN_TXD_LIN_0 \
    siul2_iopack(SIUL2_PORT_A, SIUL2_PAD_12) /* PA[12] - MSCR_IO12 */
#define PIN_TXD_LIN_0_LATCHED_STATE SIUL2_IO_LOW

```

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

As soon as you have a configuration without any error (i.e. without any red markers), the tool will generate an HTML file corresponding to the current configuration in the folder "src-gen".

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

The developer can open the HTML file by the contextual menu on *.pincfg* file

Alternatively he/she can use the dedicated popup "*Open diagram*" option (2) or toolbar option icon (3) or "*Pincfg editor*" menu option (4)



Figure 20: Open an HTML file

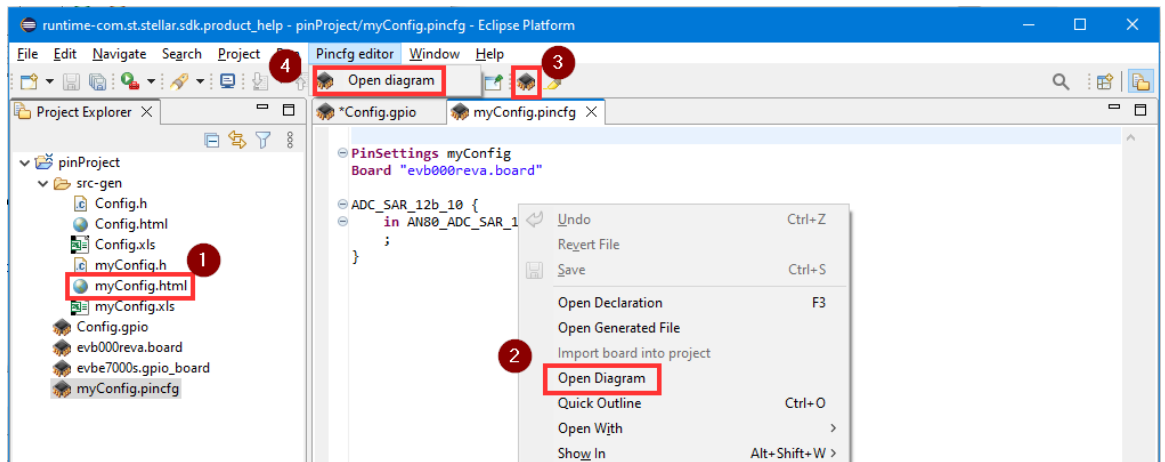
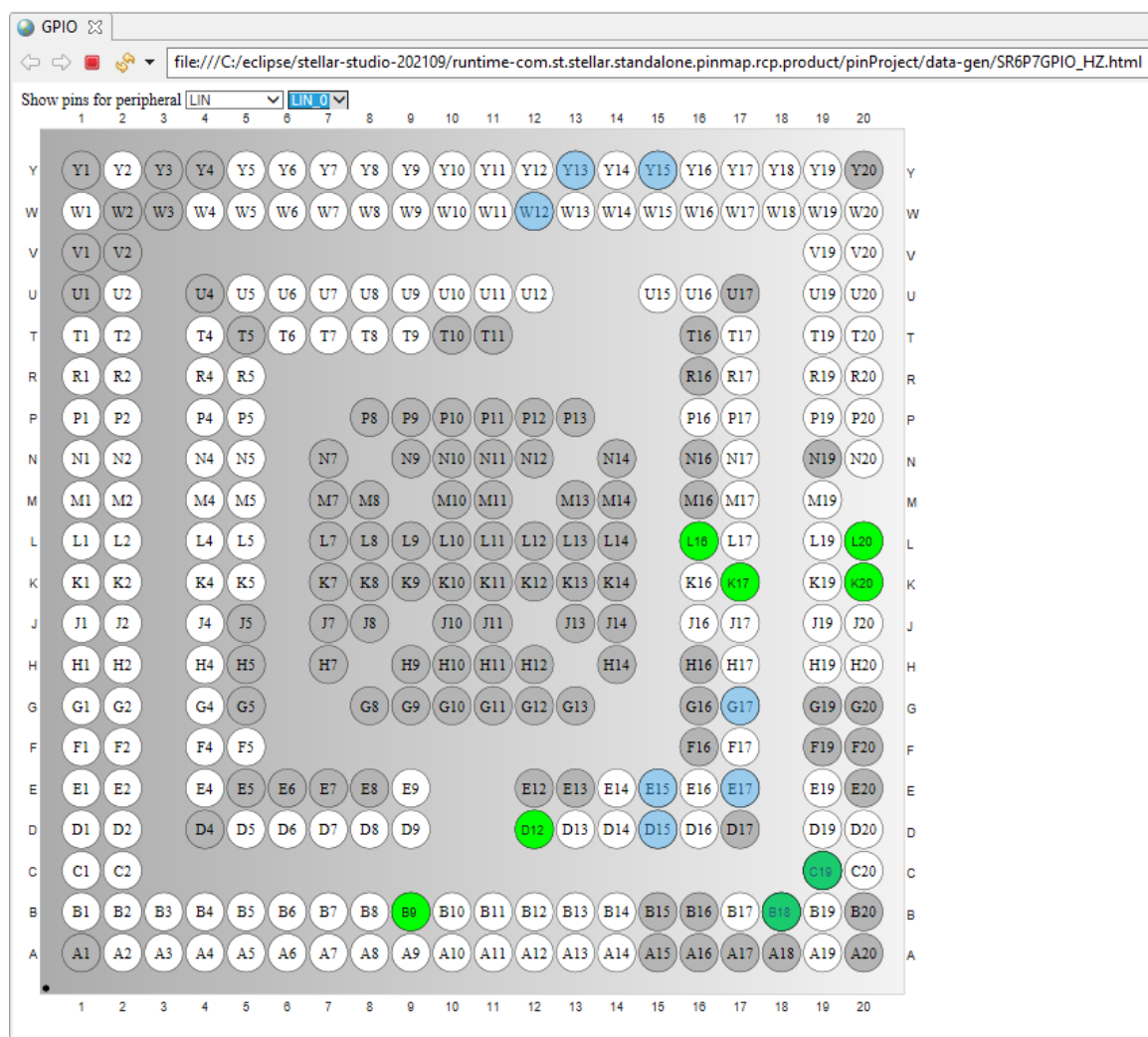


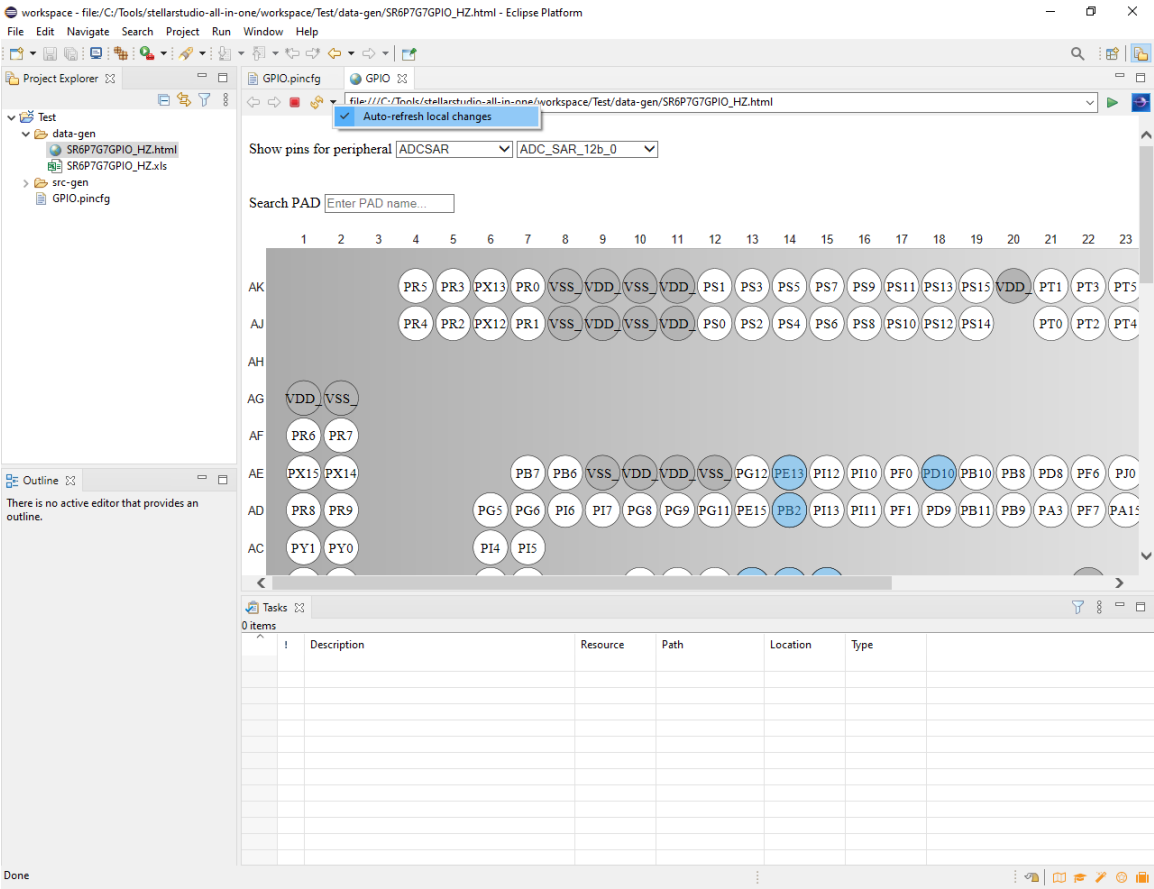
Figure 21: HTML generated from the configuration



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 22: 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

As soon as you have a configuration without any error (i.e. without any red markers), the tool will generate an XLS file corresponding to the current configuration in the folder "data-gen".

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

The developer can open the XLS file by Microsoft Excel® in the *data-gen* folder

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

Figure 23: Open an XLS file

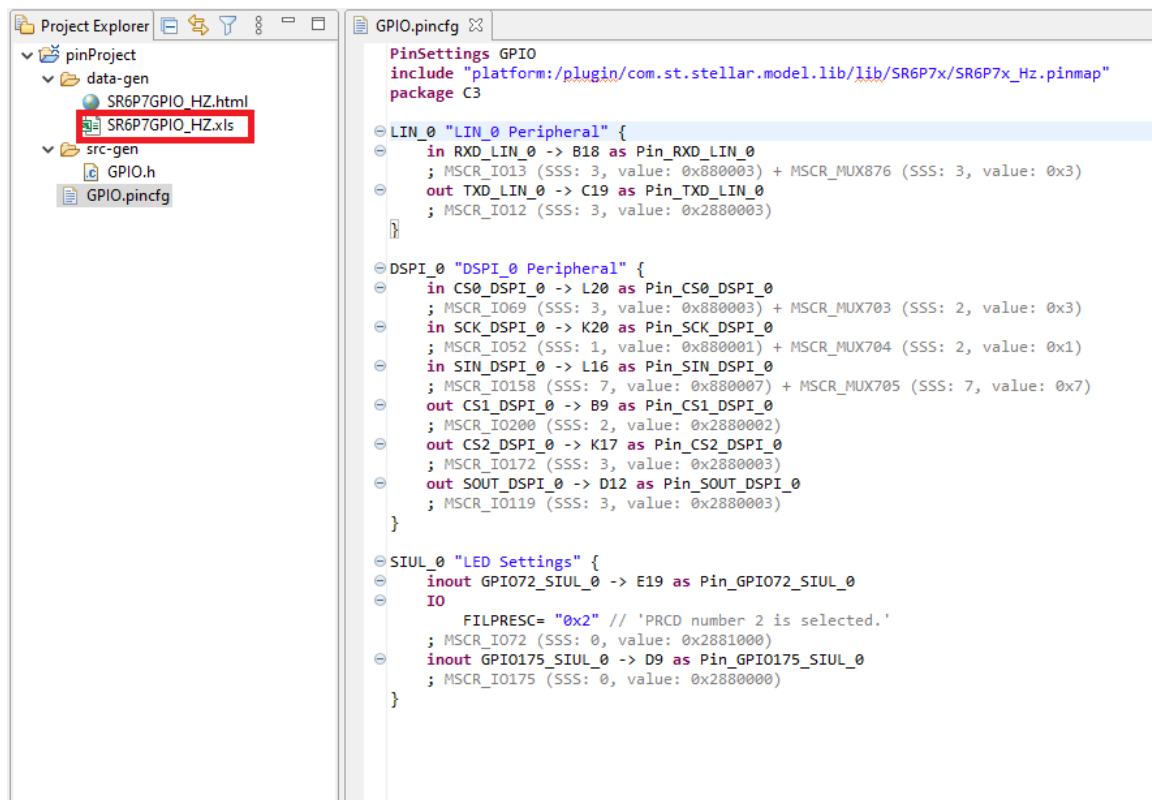


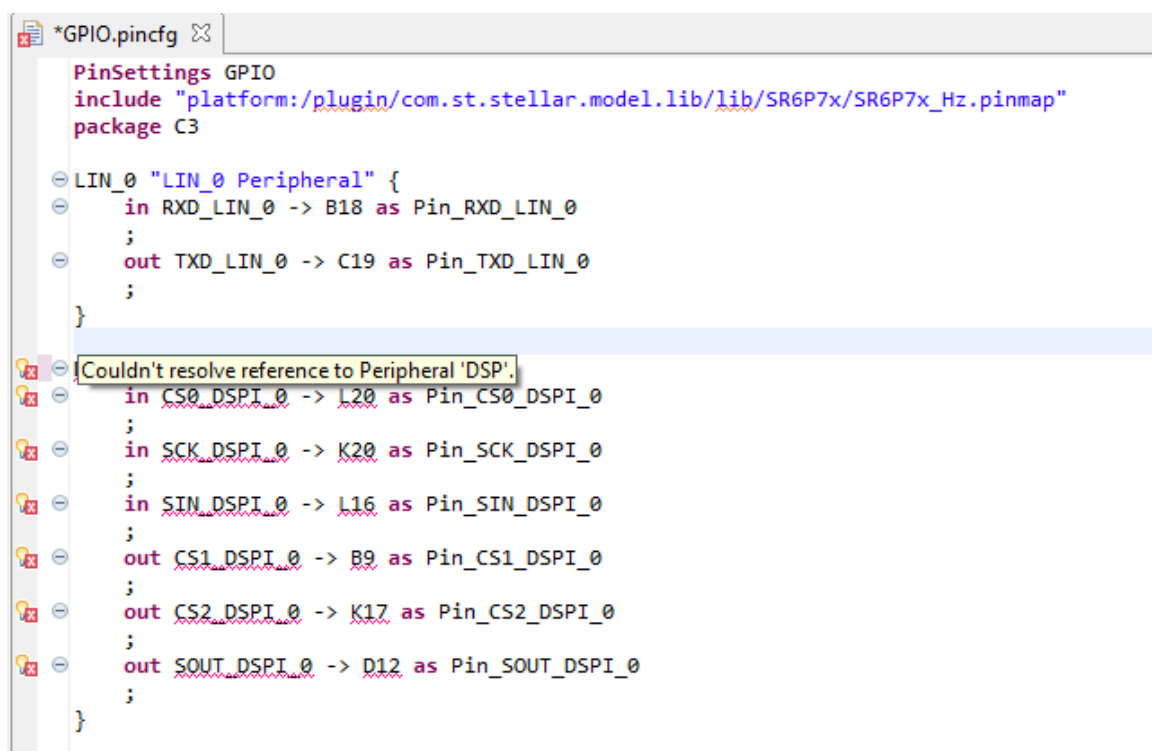
Figure 24: Example of XLS generated from the configuration

#	A	B	C	D	E	F	G	H
		Pin identifier	Active	Function direction	FunctionName	Peripheral	Function alternate	Pin advanced settings
1	B18	Pin_RXD_LIN_0	true	Input	RXD_LIN_0	LIN_0	RXD_LIN_0(876_3)	
2	C19	Pin_TXD_LIN_0	true	Output	TXD_LIN_0	LIN_0	TXD_LIN_0(12_3)	
3	L20	Pin_CS0_DSPI_0	true	Input	CS0_DSPI_0	DSPI_0	CS0_DSPI_0(69_3)	
4	K20	Pin_SCK_DSPI_0	true	Input	SCK_DSPI_0	DSPI_0	SCK_DSPI_0(52_1)	
5	L16	Pin_SIN_DSPI_0	true	Input	SIN_DSPI_0	DSPI_0	SIN_DSPI_0(705_7)	
6	B9	Pin_CS1_DSPI_0	true	Output	CS1_DSPI_0	DSPI_0	CS1_DSPI_0(200_2)	
7	K17	Pin_CS2_DSPI_0	true	Output	CS2_DSPI_0	DSPI_0	CS2_DSPI_0(172_3)	
8	D12	Pin_SOUT_DSPI_0	true	Output	SOUT_DSPI_0	DSPI_0	SOUT_DSPI_0(119_3)	
9	E19	Pin_GPIO72_SIUL_0	true	InputOutput	GPIO72_SIUL_0	SIUL_0	GPIO72_SIUL_0(72_0)	
10	D9	Pin_GPIO175_SIUL_0	true	InputOutput	GPIO175_SIUL_0	SIUL_0	GPIO175_SIUL_0(175_0)	
11	A1		NONE	Ground	VSS_HV			
12	A10		NONE		PF2			
13	A11		NONE		PD2			
14	A12		NONE		PD0			
15	A13		NONE		PF5			
16	A14		NONE		PM10			
17	A15		NONE	Power	SMPS_PMOS			
18	A16		NONE	Power	SMPS_NMOS			
19	A17		NONE	Ground	VSS_HV			
20	A18		NONE	Power	SMPS_VLX			
21	A19		NONE		PA1			
22	A2		NONE		PD15			
23	A20		NONE	Ground	VSS_HV			
24	A3		NONE		PB11			
25	A4		NONE		PH15			
26	A5		NONE		PM4			
27	A6		NONE		PC15			
28	A7		NONE		PC13			
29	A8		NONE		PM6			
30	A9		NONE		PC12			
31	B1		NONE		PC8			
32	B10		NONE		PH13			
33	B11		NONE		PD3			
34	B12		NONE		PD1			
35	B13		NONE		PH12			
36	B14		NONE		PH14			
37	B15		NONE	Power	VDD_UV_SMPS			
38	B16		NONE	Power	VDD_UV_SMPS			
39	B17		NONE		PE10			
40	B18		NONE		PA13			
41	B19		NONE		PA0			
42	B2		NONE		PC14			

### 3.6 Configuration errors

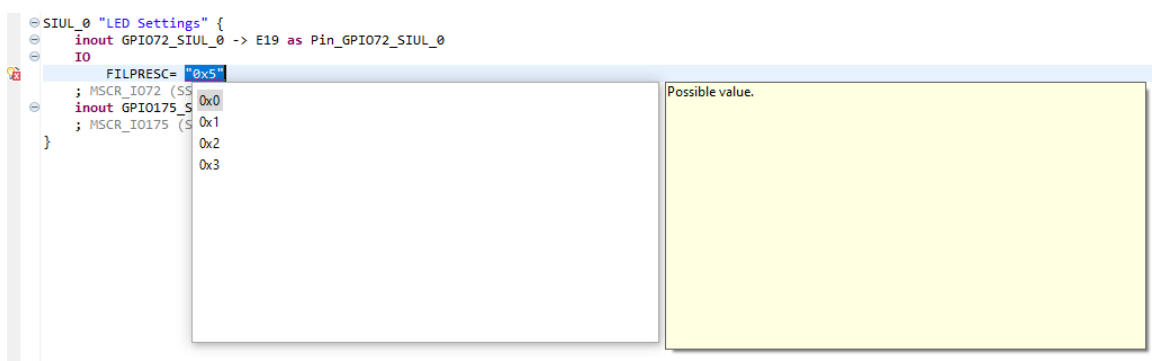
Errors in naming peripherals, pin configuration, as well as syntax errors in the expected keywords, are identified with some markers in the configuration editor.

Figure 25: Syntax Errors



For some errors a quick fix is available.

Figure 26: Example of semantic errors and quickfixes

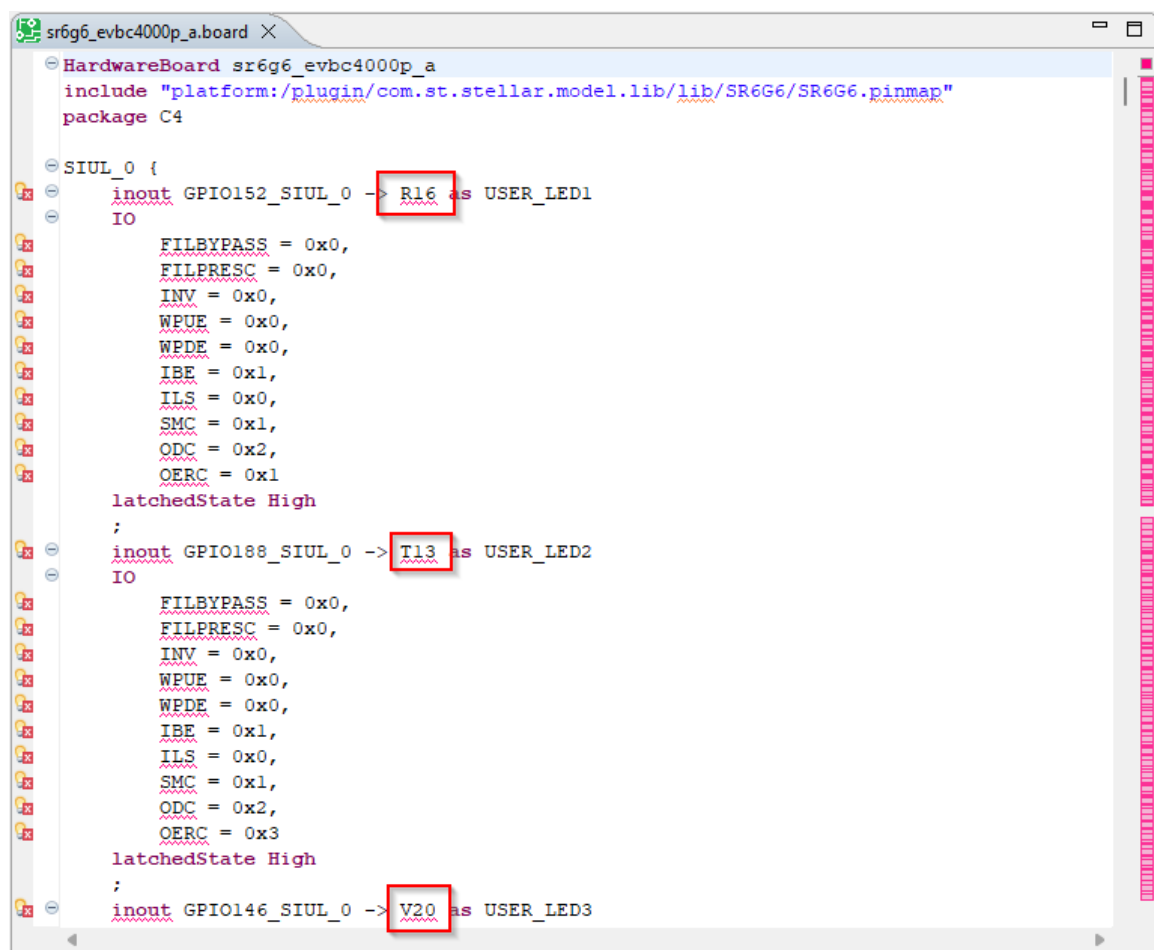


### 3.7 Application migration

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.

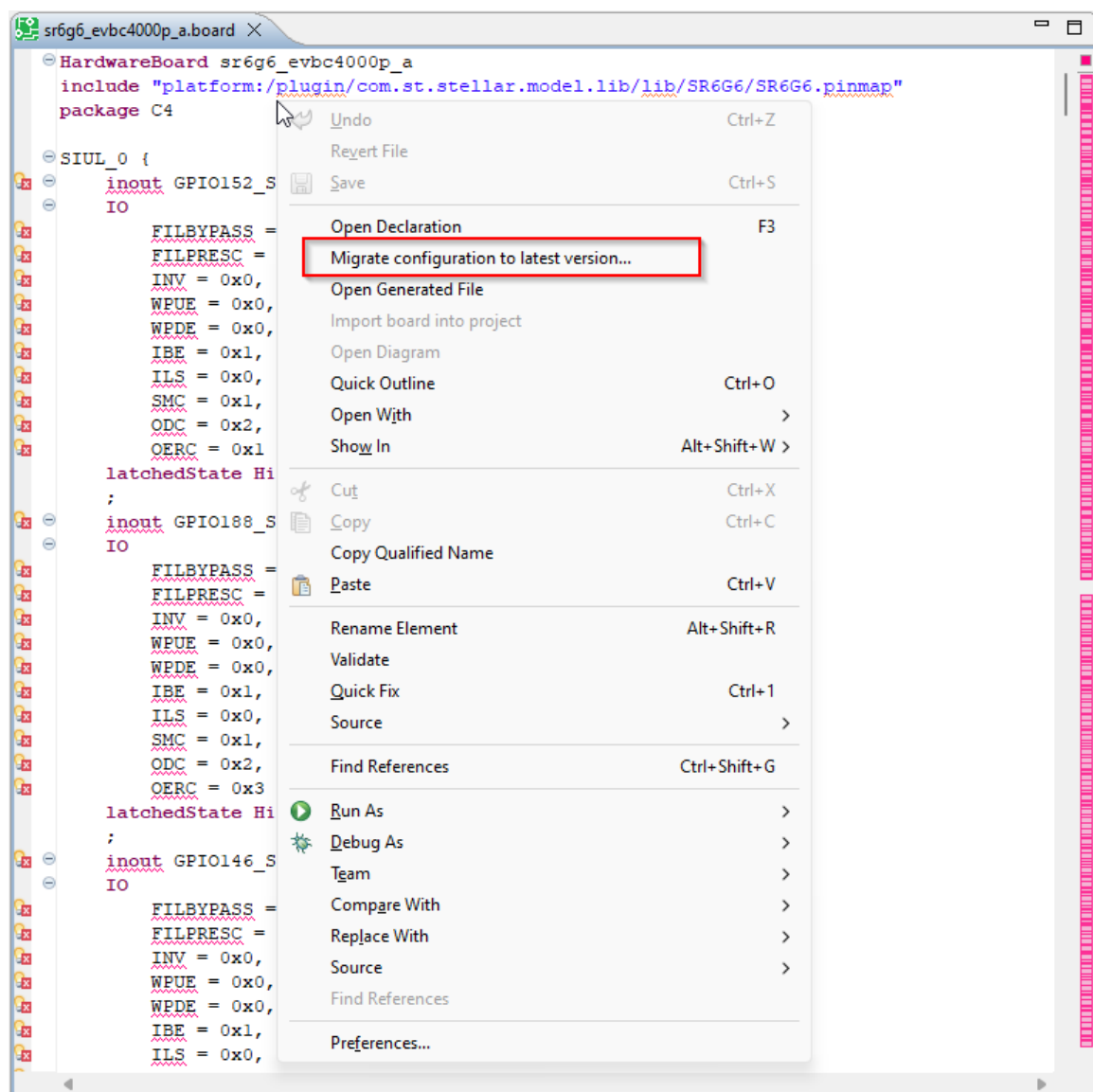
Figure 27: Old applications Migration



In the example above the pinmap configuration has changed: instead of pin name we are now using the PAD name. The file needs to be migrated.

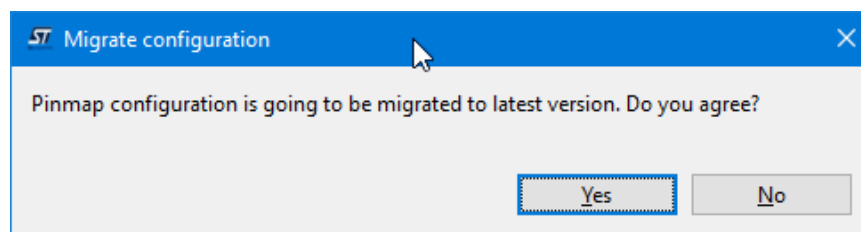
In order to fix these errors, you must use the migration tool that is accessible using right click inside the configuration editor.

Figure 28: Launching migration



You'll be asked to confirm your choice.

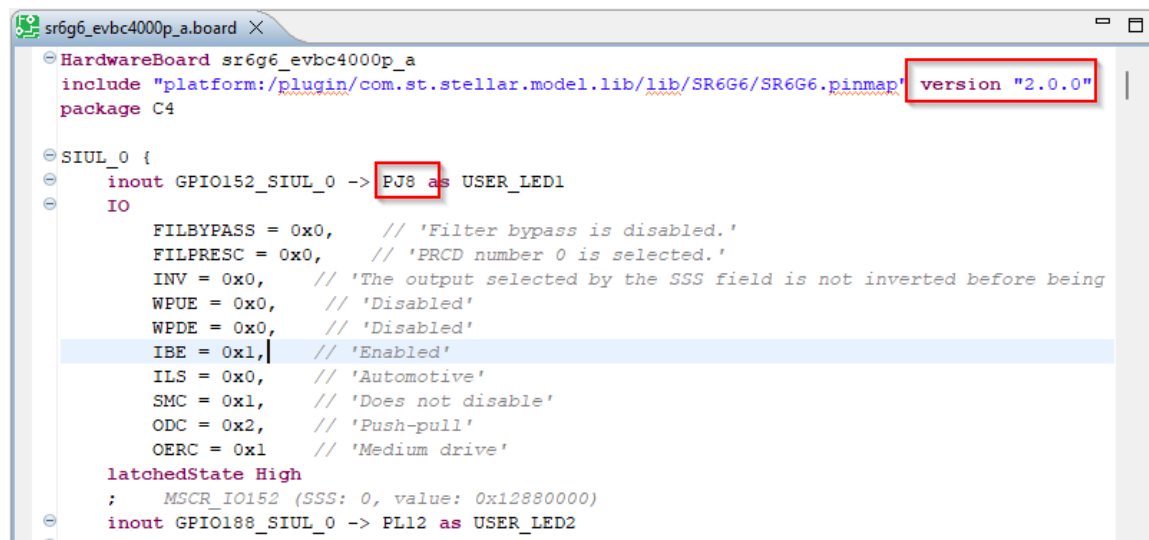
Figure 29: Confirm migration





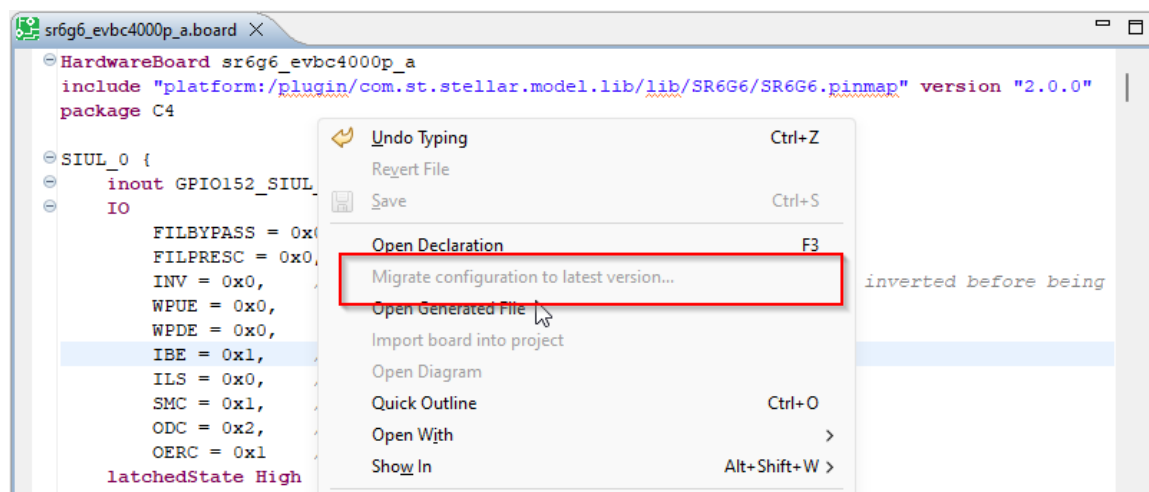
As a result the pin names have been adapted to the new names (pin name R16 has been replaced by the pad PJ[8]) and the version of the model is now changed and appears inside the converted file (here 2.0.0).

Figure 30: Migration result



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

Figure 31: Unneeded migration



### 3.8 Package swapping

With the version 7.1 of StellarStudio comes the possibility of swapping a pin configuration from one package to another.

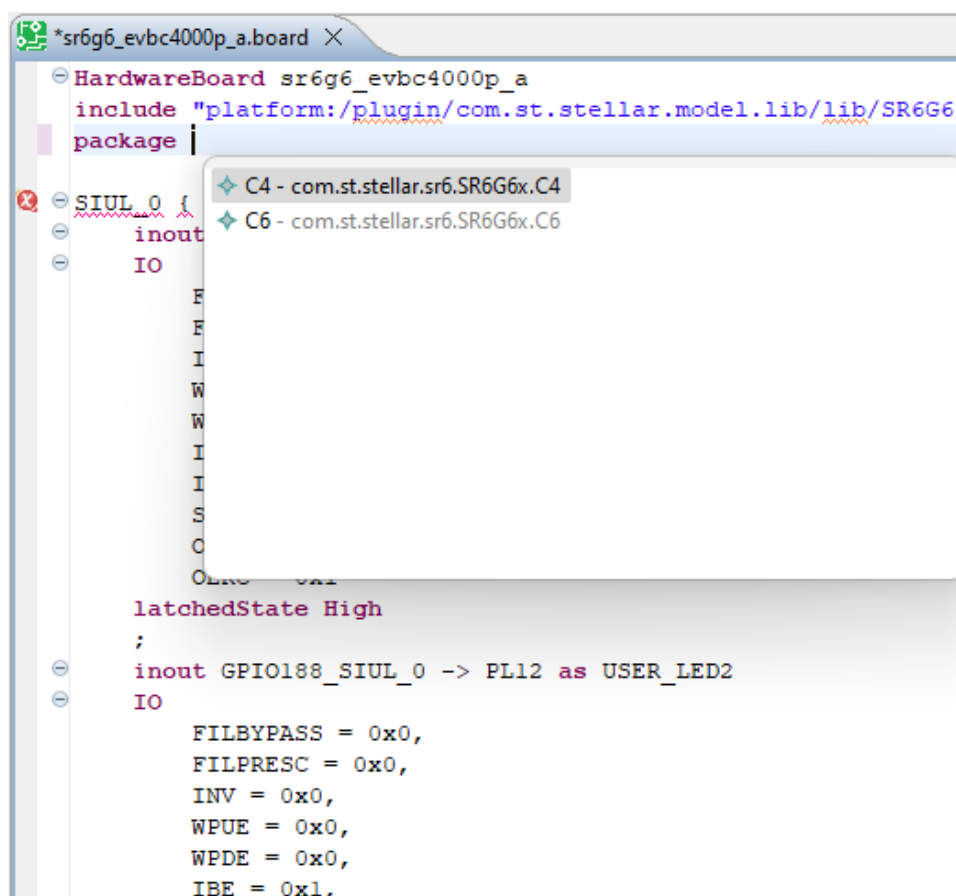
Moving from a small package to a bigger package will always be possible, as assigned pads will always be present in the bigger package.

On the contrary, moving from a big package to a smaller one may reveal some errors, as there are less accessible pads in the old package than in the new one.

In such cases, the user will have to fix allocation errors inside the textual configuration, removing configuration that are not possible, or assign functions to other, not allocated pads.

The swap in itself is done using the textual editor, changing the package name (for example C4 in the example below) inside the board file.

**Figure 32: Package swapping**



You can choose the package you want to use from the list of available packages, using the CTRL-SPACE (or CTRL-TAB) keyboard combinations as usual.

### 3.9 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 33: Outline for pin configuration file

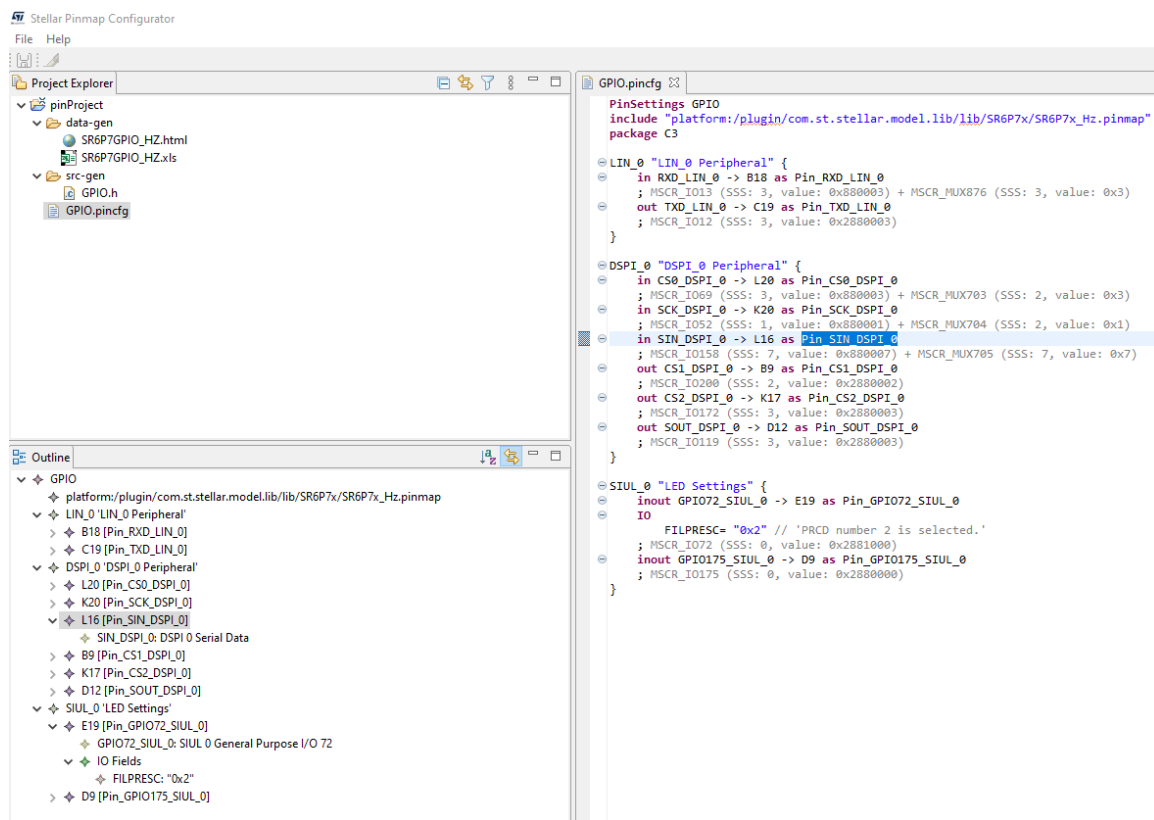
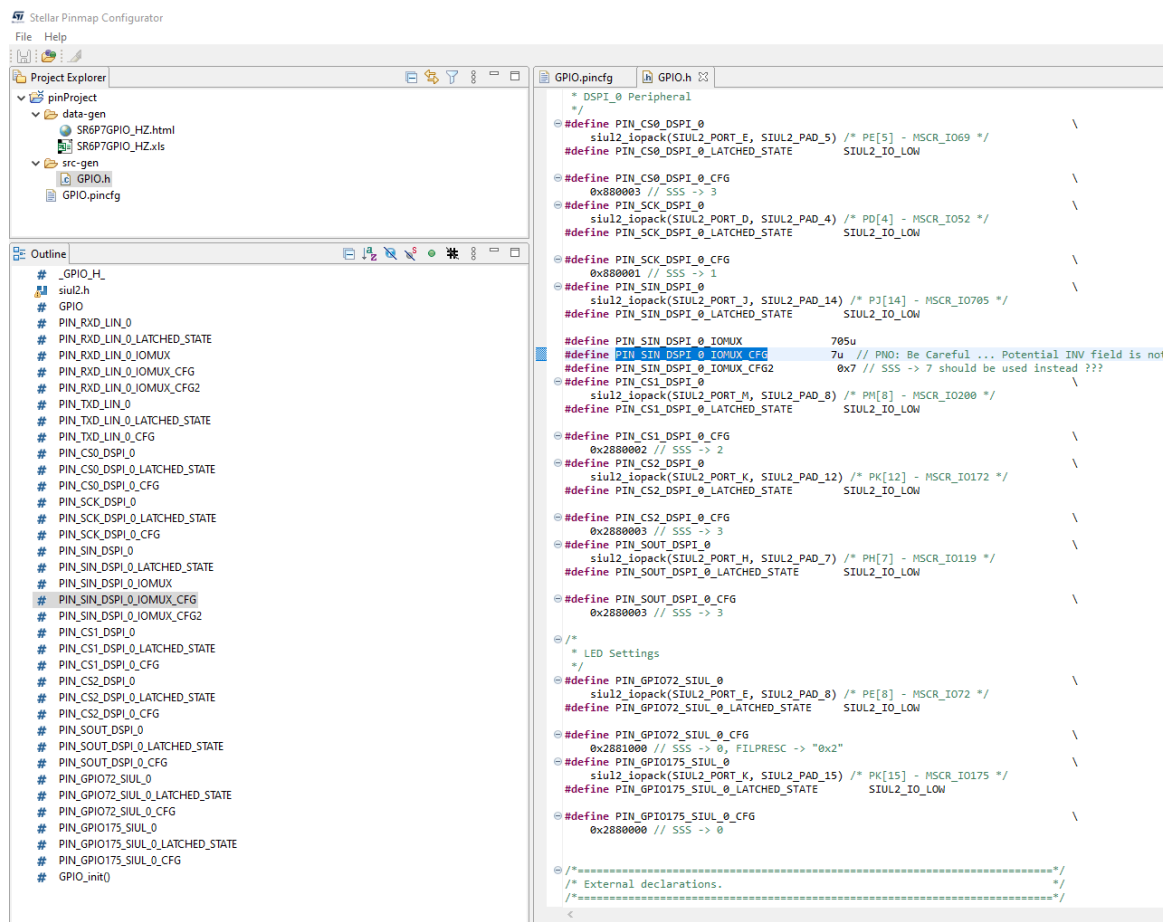


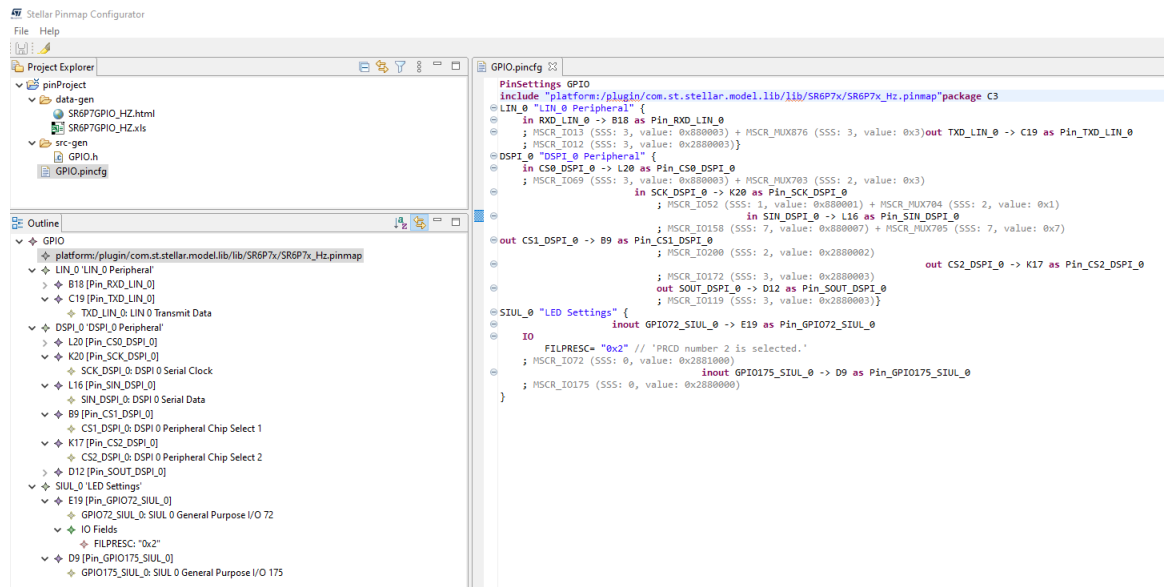
Figure 34: Outline for a C generated file



### 3.10 Pin configuration formatting

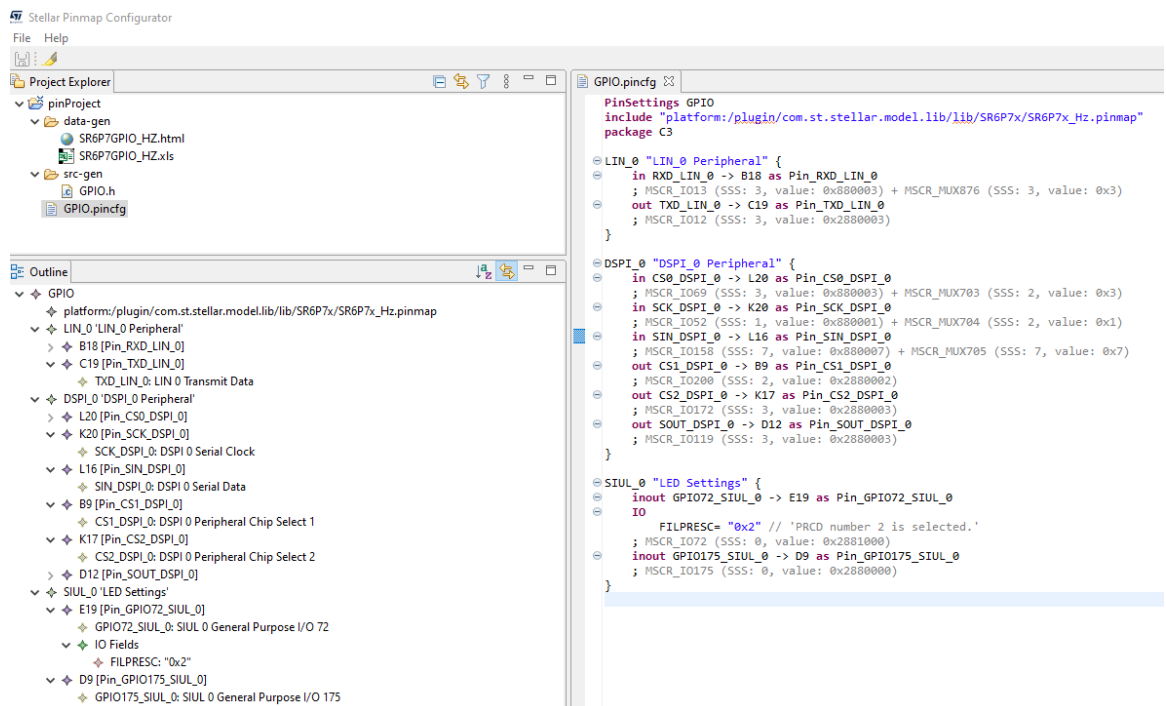
As any other textual editor, you can rearrange its appearance using a default formatting option, accessible with the key combination <CTRL><SHIFT>F

Figure 35: Example before formatting



And here is how it looks like after formatting:

Figure 36: 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:

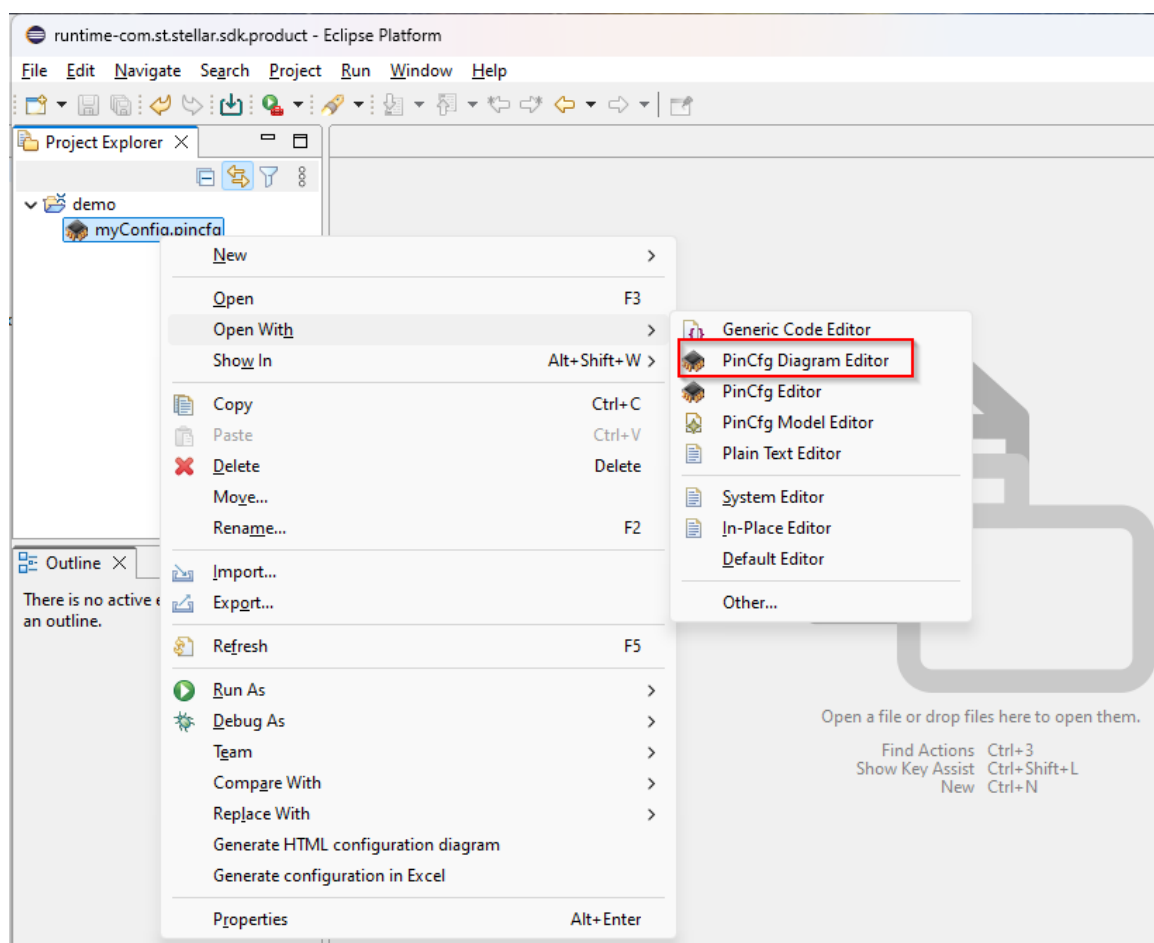
- Open the graphical configuration editor
- Use the outline view to identify where peripherals, functions or PADs are accessible,
- Graphically assign/unassign a function to/from a pin,
- Ask to show a configured pin inside the textual configuration editor,

### 4.2 Open the graphical configuration editor

There are several ways to open the graphical configuration editor.

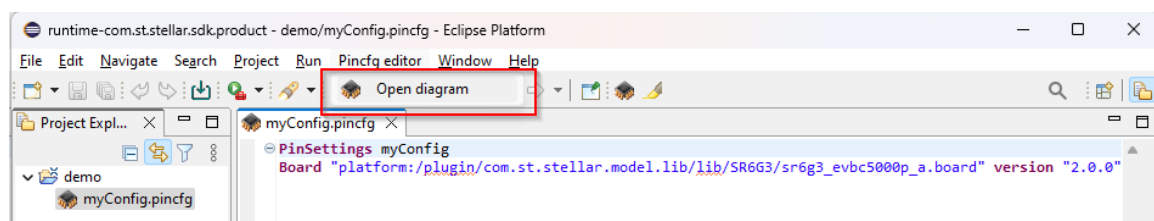
You can open it from the project explorer, using a right click on a *.gpio* file, then choose the Open with/Pincfg diagram editor.

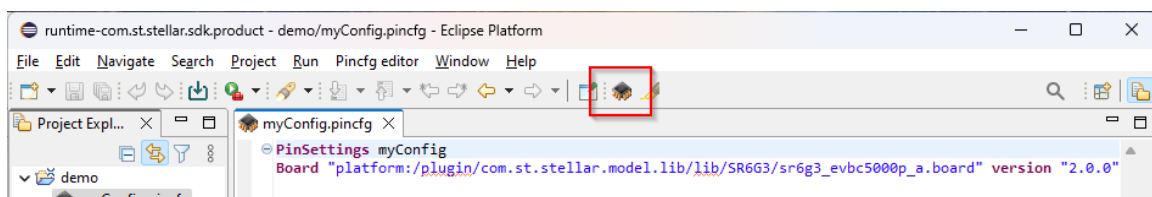
Figure 37: Open the graphical editor from project explorer



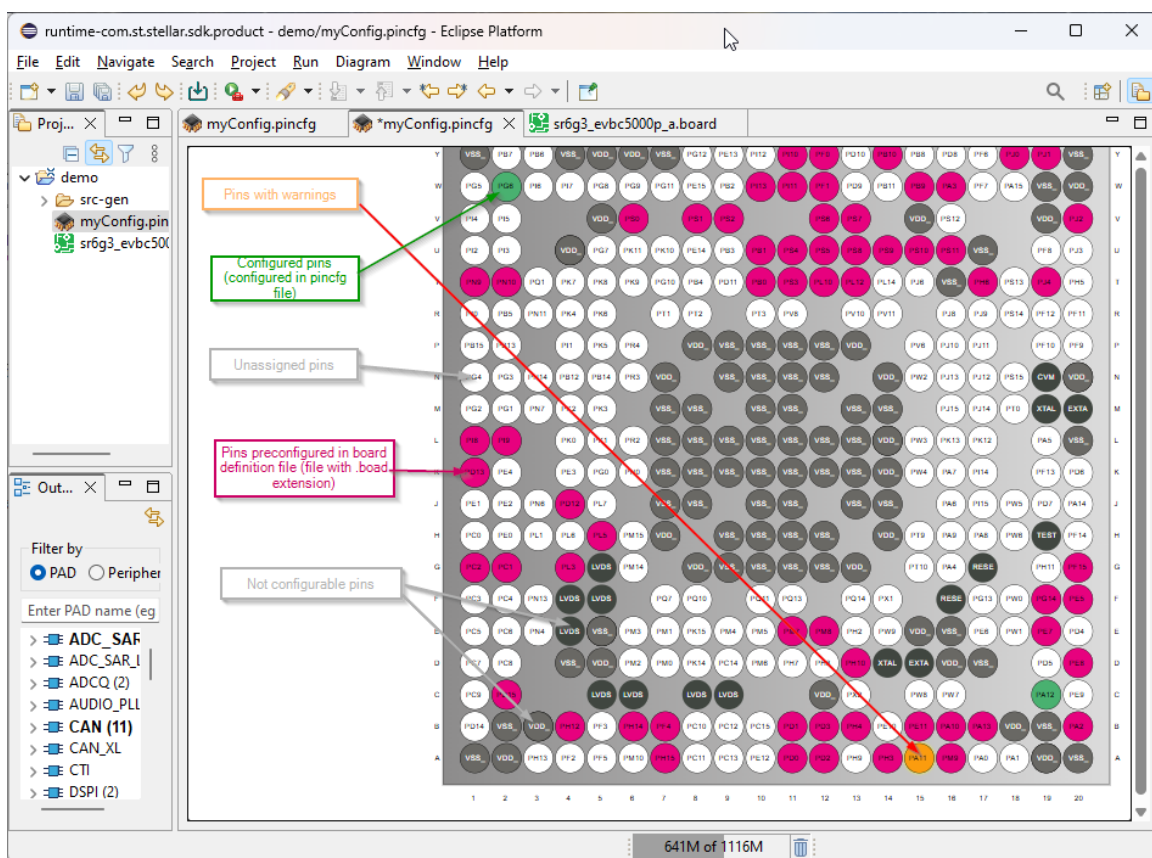
Alternatively you can use the Gpio editor menu, or click on the MCU icon in the toolbar.

Figure 38: Open the graphical editor from Gpio editor menu



**Figure 39: Open the graphical editor from the MCU icon in the toolbar**

The result would be similar to the below picture.

**Figure 40: Result: opened graphical editor**

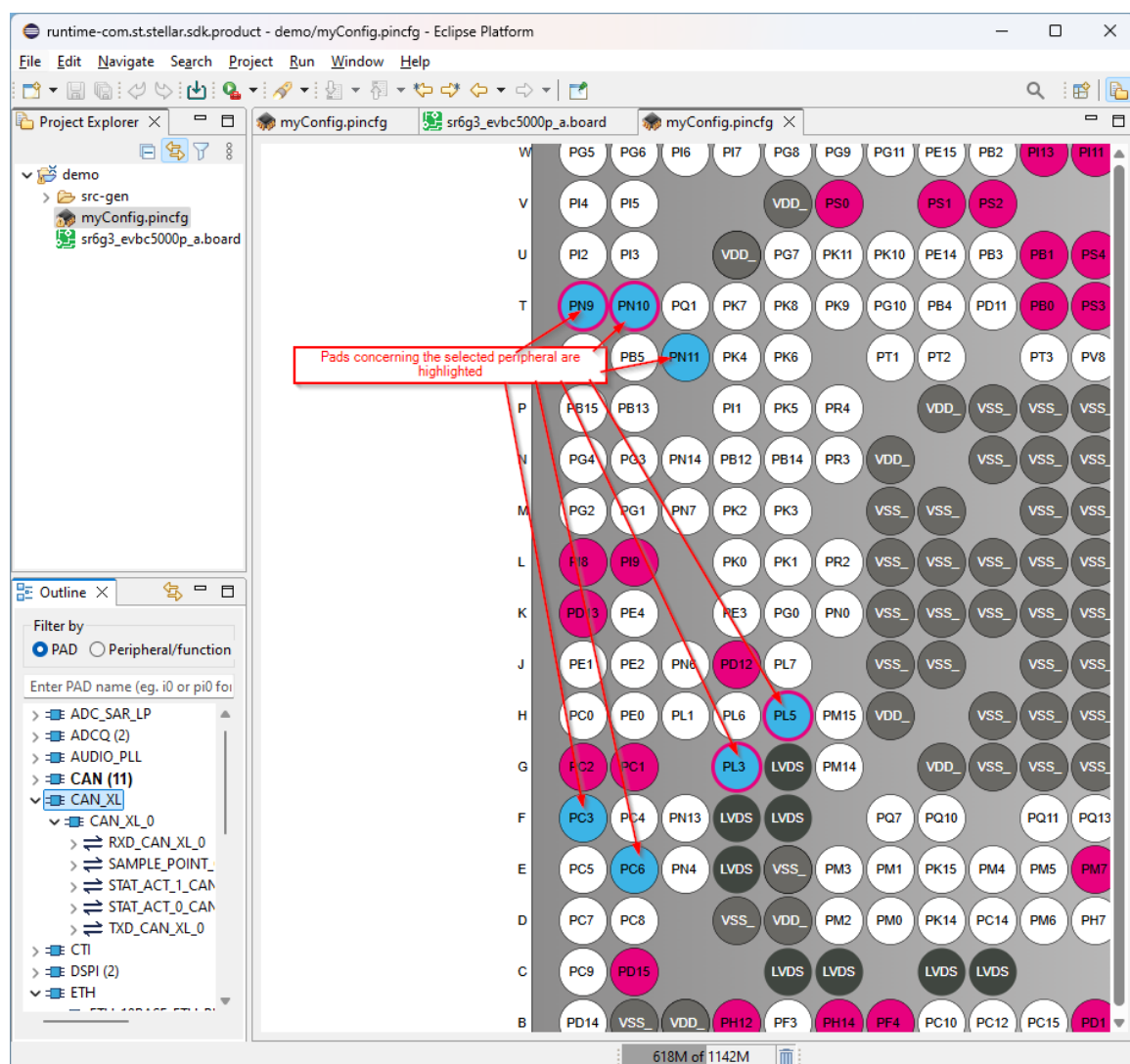
### 4.3 Using pinmap configuration outline views

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

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

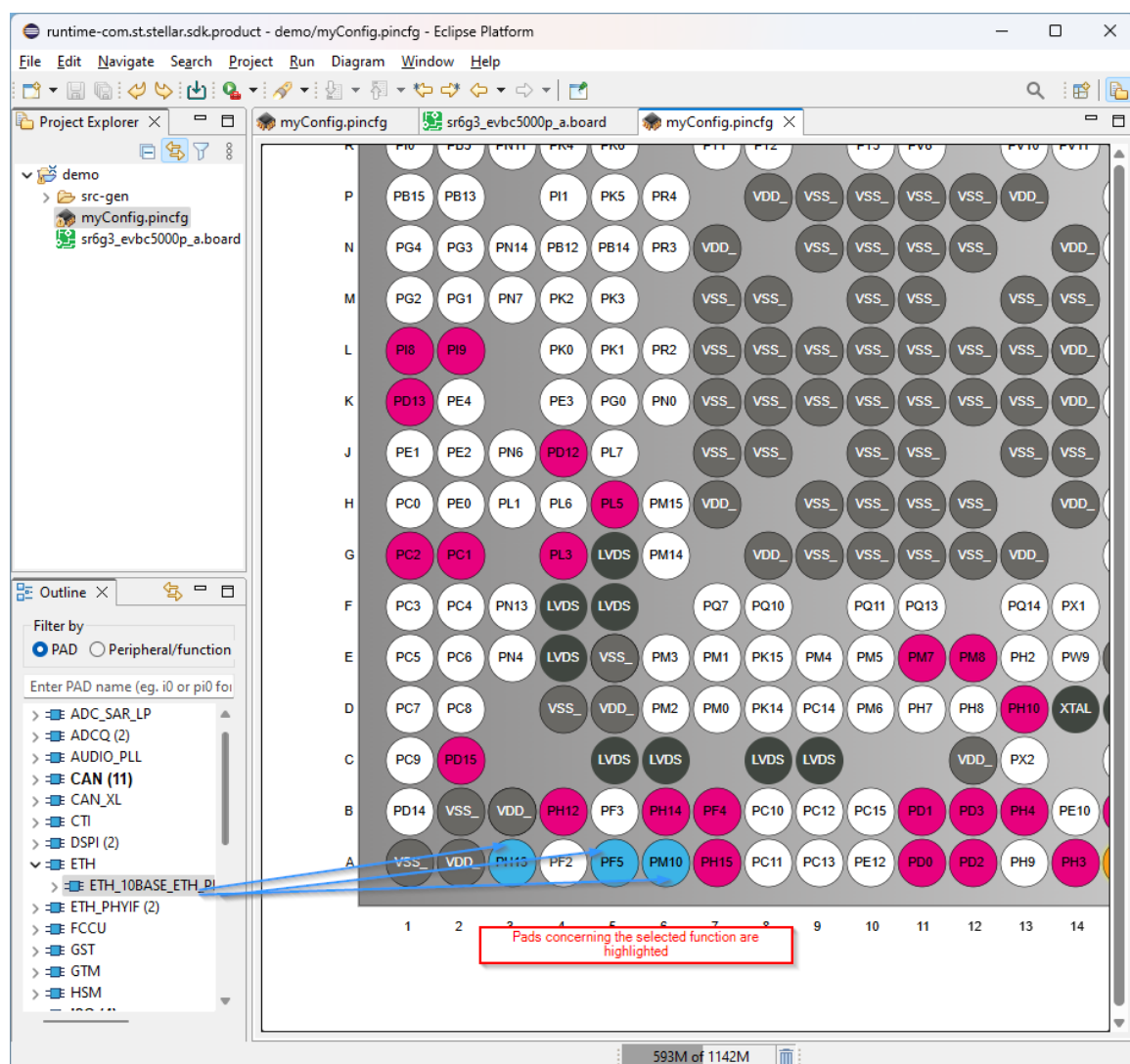


Figure 41: Example of selection a peripheral



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

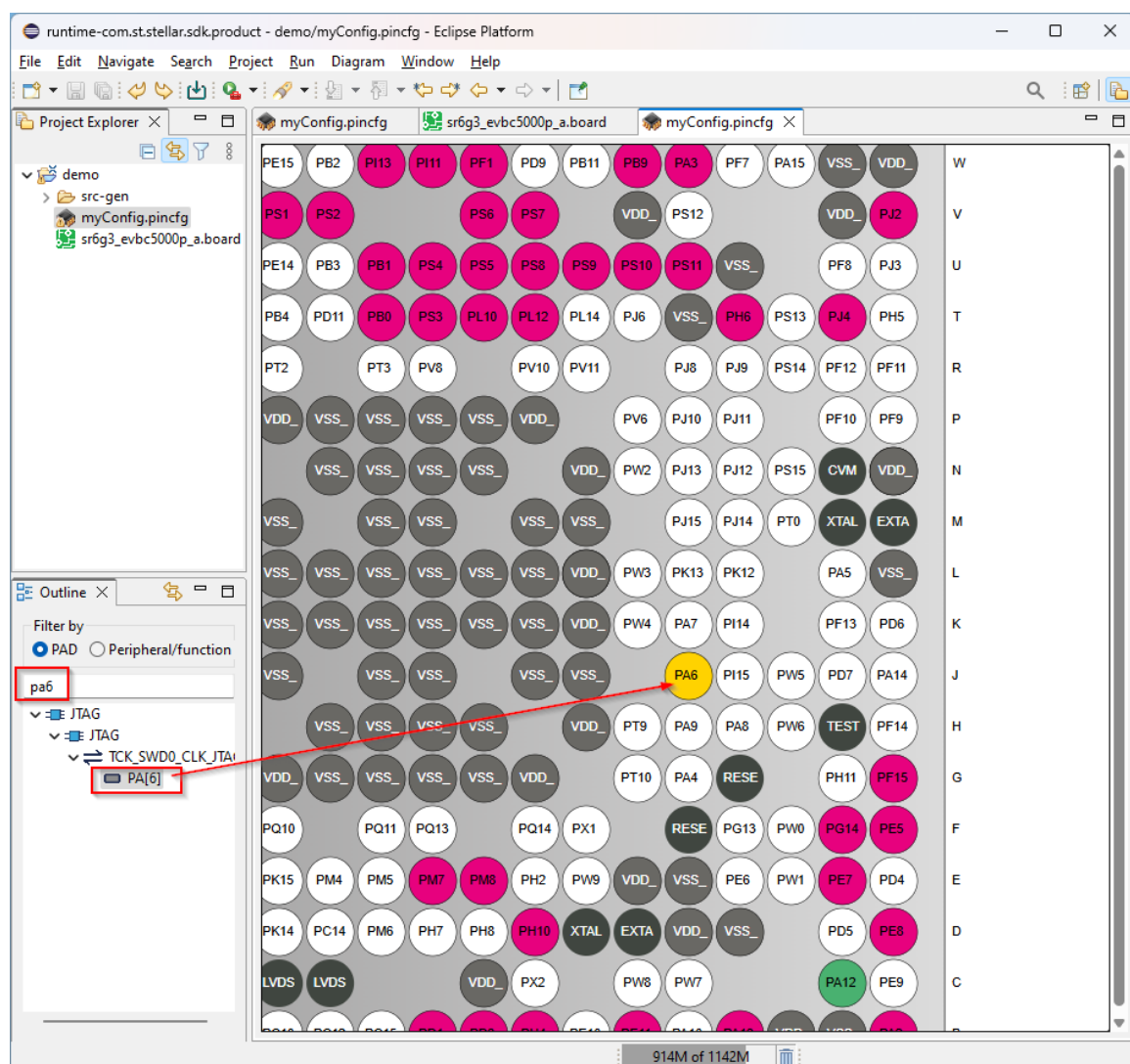
Figure 42: Example of selection a function



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 43: Example of searching of a PAD

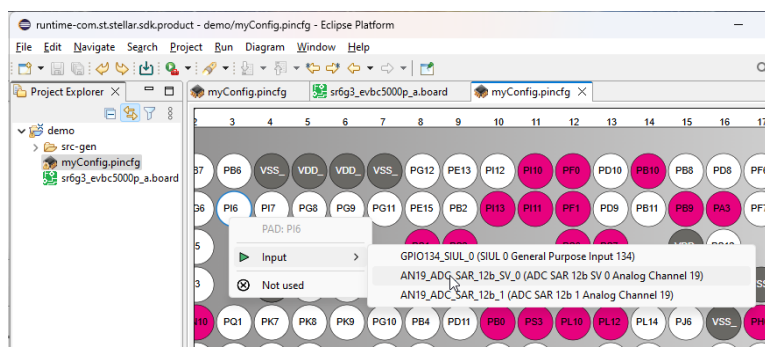


## 4.4 Assigning a function to a pin

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 44: Assign a function to a pin

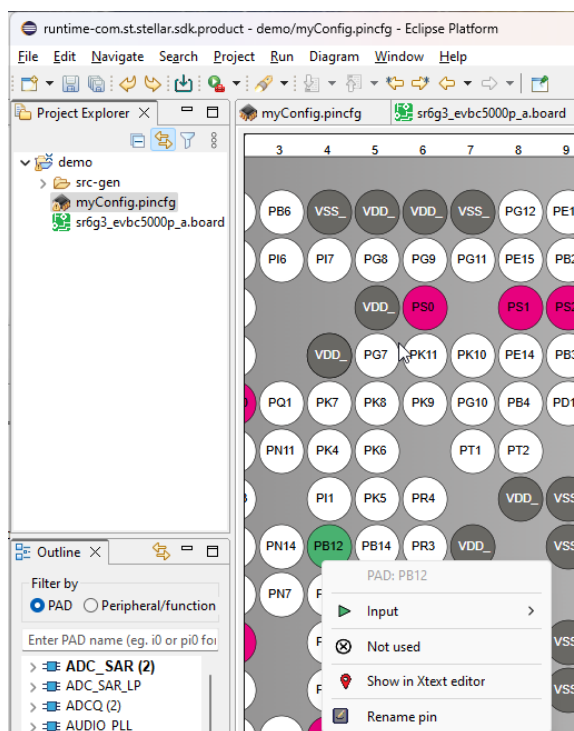


The result will show the assigned pin, with its default name.

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 45: 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.

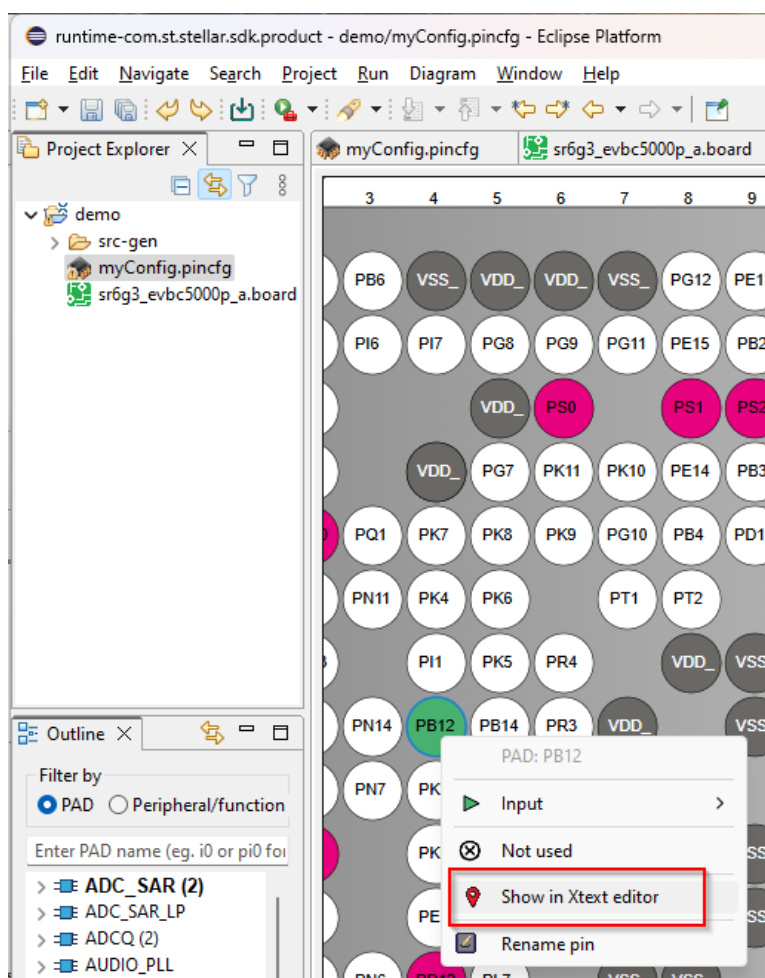
NB: Saving it (using the floppy disk from the toolbar icon) is sometimes not available. If this occurs, you'll need to click on the graphical editor tab in the editor area, then the save icon will be accessible.

## 4.5 Locate graphical pins in textual editor

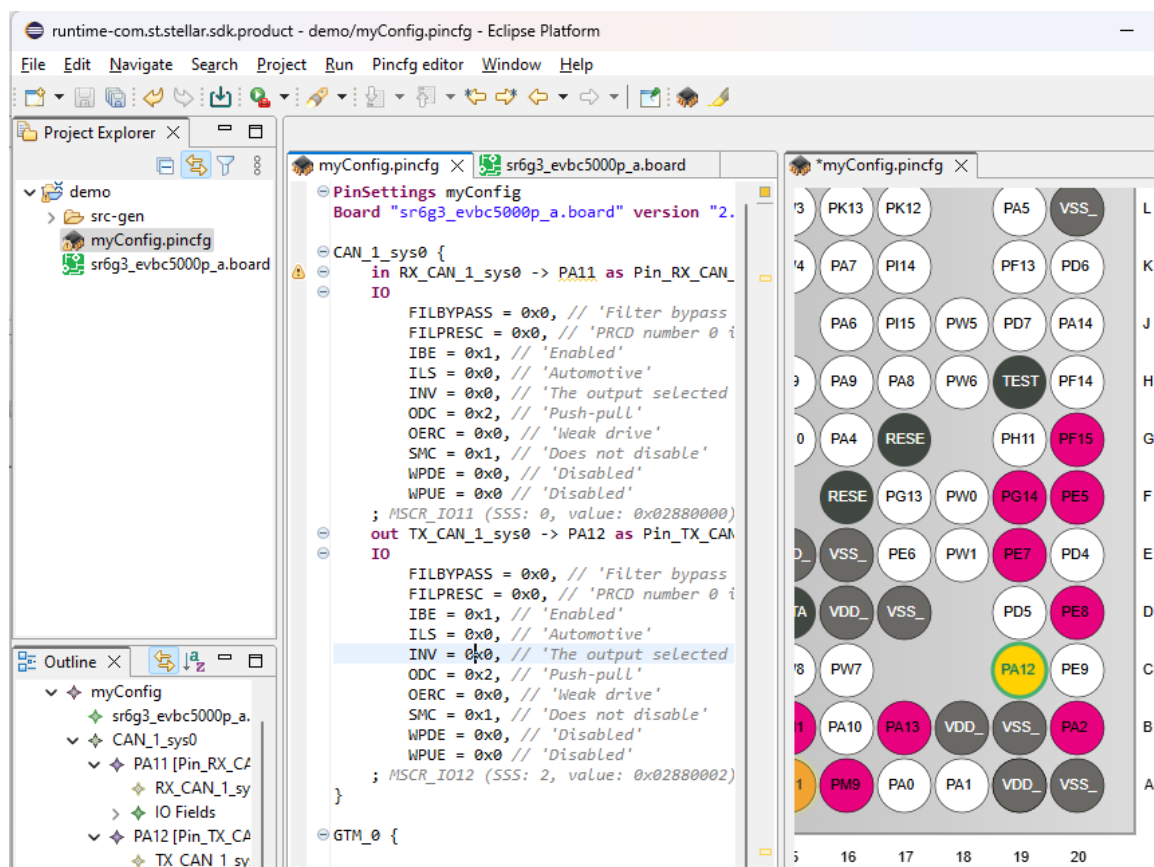
As the graphical configuration editor show a lot a pins, it is not obvious to see where is it present inside the textual editor.

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 46: 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 file and configured your peripherals, the following section describes how to integrate your generated file or pin configuration file in IDE product.

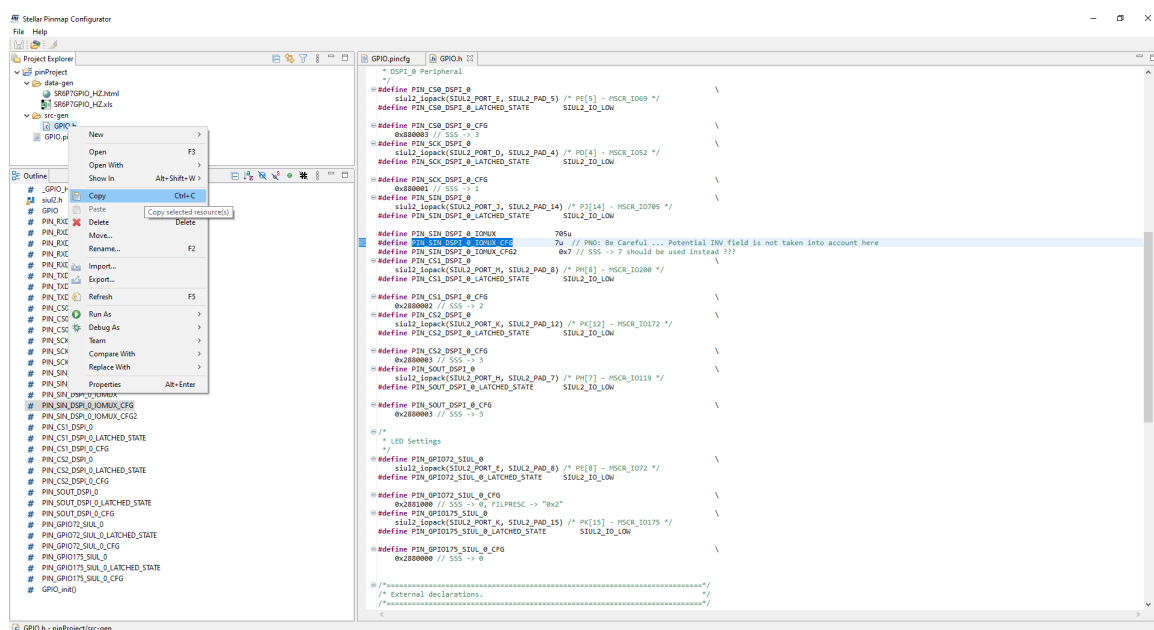
### 5.1 Integration of your generated file (C file)

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

Select your C file

With contextual menu, copy your file

Figure 47: Copy your C file

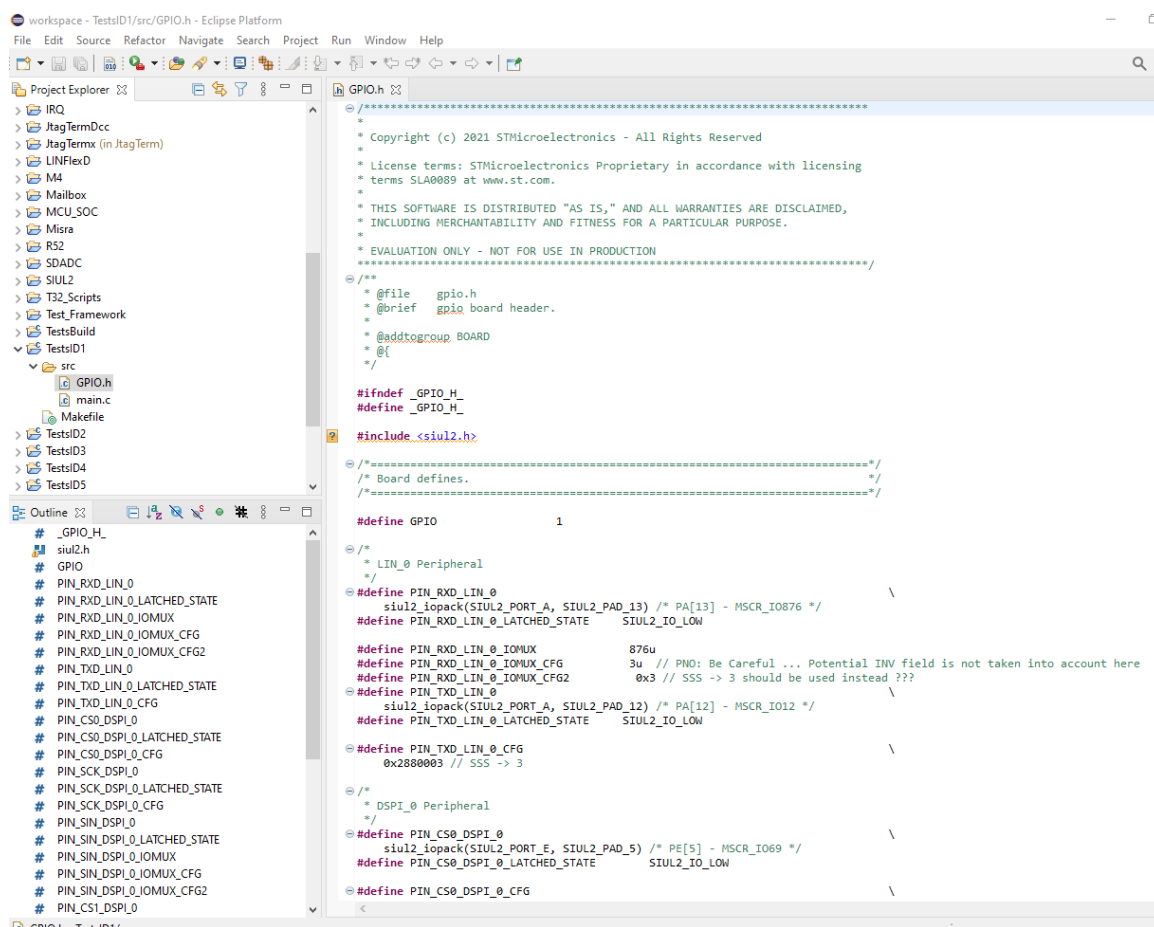


Integrate it, in your favorite developer tool (Eclipse, Visual Studio Code and so on ..).

With contextual menu, paste your file.



Figure 48: Paste your C file in your favorite developer tool



## 5.2 Integration of your PinCfg file

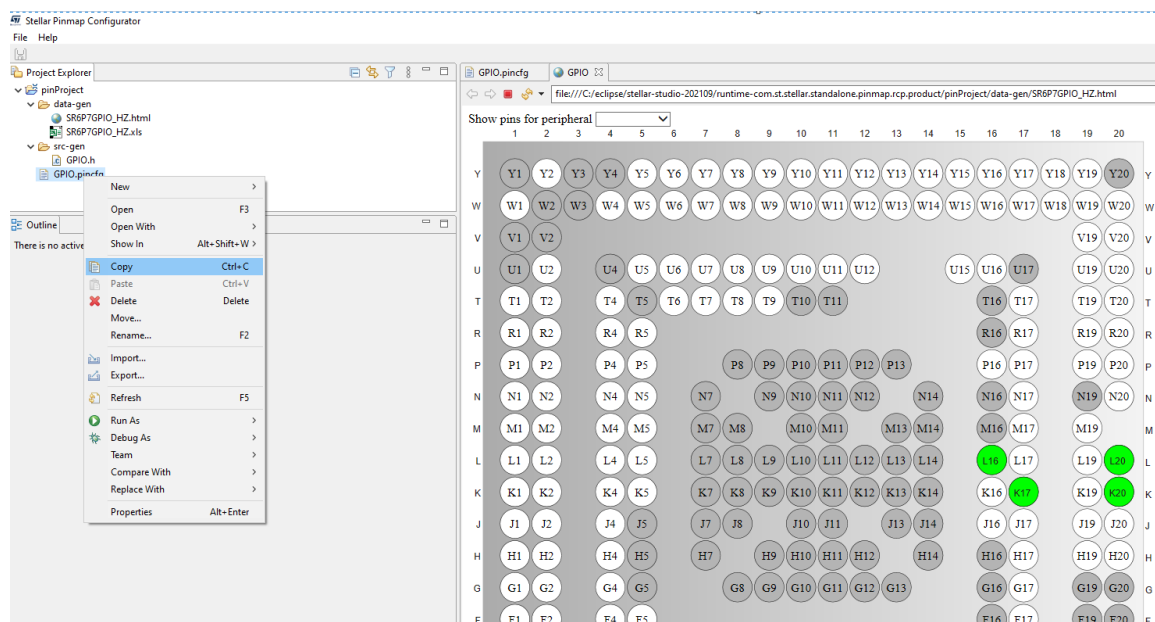
you can integrate your pinmap configuration in StellarStudio all-in-one product

From StellarStudio Pinmap Configurator, select your *pincfg* file

With contextual menu, copy your file



Figure 49: Copy your file



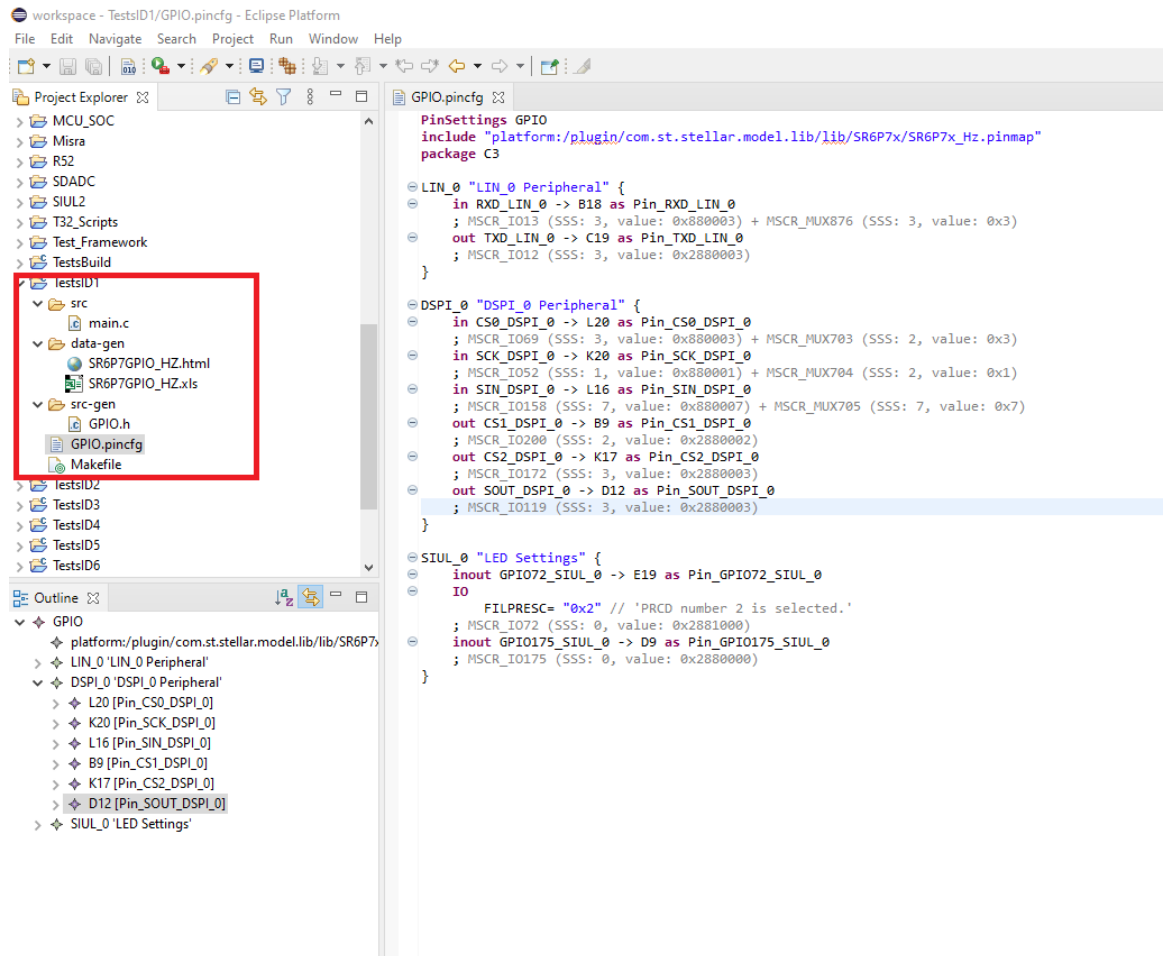
Integrate it, in StellarStudio all-in-one product.

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

after clicking yes on the conversion on a xtext project,

if your *pinconf* is valid, generated files should displayed in *data-gen* and *src-gen* folder

Figure 50: Paste your pincfg file in StellarStudio all-in-one product



### 5.3 Adapt project's Makefile

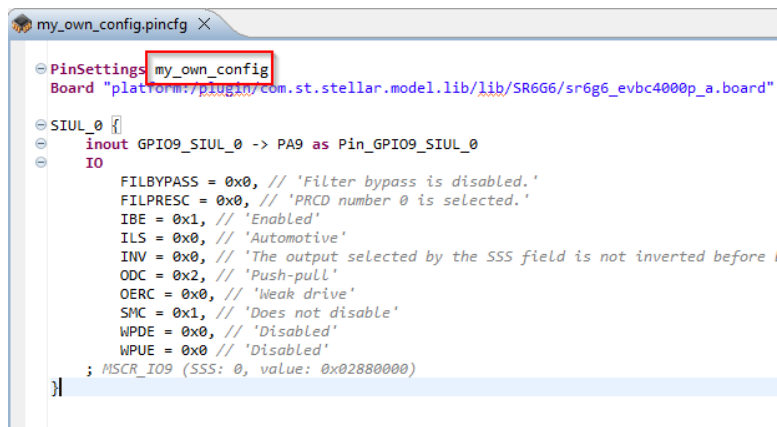
If you want your project to use the pinmap configuration that you've created, you'll need to adapt the Makefile to point to the correct generated configuration header file, generated from your *.pincfg* file.

The default Makefile uses the board *evbc4000p* from the S-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_DEVICE`: must contain the device name, and `CONFIG_BOARD`: must contain the name of the configuration (i.e. the identifier directly following the PinSetting keyword in the *.pincfg* file).
2. `C_INCS`: must be augmented with the location of the folder containing the generated header file (i.e. *src-gen* project's folder).

Figure 51: Name of the configuration to be used



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 52: Adapt CONFIG\_DEVICE, CONFIG\_BOARD and C\_INCS

```

CONFIG_DEVICE ?= sr6g6
CONFIG_BOARD ?= my_own_config
CONFIG_TARGET_MEMORY ?= ram # ram or nvm

include $(PROJECT_COMMON_DIR)/mk/config.mk

ifeq ($(wildcard $(STELLAR_SDK_BUILD_SYSTEM_DIR)/\
$(error $(STELLAR_SDK_BUILD_SYSTEM_DIR)/StellarSDK
$(error Install the $(SDKID) or Correct STELLAR_SDK
endif

#####
# Project builds
#####

BUILD_BOOT_R52                := 1
BUILD_PLATFORM_BOARD           := 1
BUILD_PLATFORM_CLOCK           := 1
BUILD_PLATFORM_CORE            := 1
BUILD_PLATFORM_IRQ             := 1
BUILD_PLATFORM_MCU             := 1
BUILD_DRIVERS_SIUL2            := 1
BUILD_DRIVERS_AGT              := 1
BUILD_DRIVERS_GST              := 1

BUILD_DRIVERS_ME               := 1
BUILD_DRIVERS_EDMA             := 1
BUILD_DRIVERS_LINFLEXD         := 1
BUILD_DRIVERS_UART             := 1
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
# DO NOT FORGET TO CLEAN THE PROJECT
# FOR THE DEPENDENCIES FILES
#####
C_INCS += \
    src-gen/

C_INCS += \
    include/

```

```

CONFIG_DEVICE ?= sr6p7g7
CONFIG_BOARD ?= sr6x7 evbc8000p_c
CONFIG_TARGET_MEMORY ?= ram # ram or nvm

include $(PROJECT_COMMON_DIR)/mk/config.mk

ifeq ($(wildcard $(STELLAR_SDK_BUILD_SYSTEM_DIR)/\
$(error $(STELLAR_SDK_BUILD_SYSTEM_DIR)/StellarSDK
$(error Install the $(SDKID) or Correct STELLAR_SDK
endif

#####
# Project builds
#####

BUILD_BOOT_R52                := 1
BUILD_PLATFORM_BOARD           := 1
BUILD_PLATFORM_CLOCK           := 1
BUILD_PLATFORM_CORE            := 1
BUILD_PLATFORM_IRQ             := 1
BUILD_PLATFORM_MCU             := 1
BUILD_DRIVERS_SIUL2            := 1
BUILD_DRIVERS_AGT              := 1
BUILD_DRIVERS_GST              := 1

BUILD_DRIVERS_ME               := 1
BUILD_DRIVERS_EDMA             := 1
BUILD_DRIVERS_LINFLEXD         := 1
BUILD_DRIVERS_UART             := 1
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
# DO NOT FORGET TO CLEAN THE PROJECT
# FOR THE DEPENDENCIES FILES
#####
C_INCS += \
    src-gen/ \
    src-gen/$(CONFIG_DEVICE)

C_INCS += \
    include/

```

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 - Sin-  
gapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

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