

# S32 SDK Quick Start Guide

Document Number: S32 SDK QSG  
Rev. 2.0, 06/2021

Introduction .....	3
1.1 System requirements .....	3
1.2 Installing S32 SDK.....	3
1.3 Release Notes .....	4
1.4 Terminology .....	4
Working with Projects.....	5
2.1 Creating and building S32DS Application Project .....	5
2.2 Importing an existing project .....	14
2.3 Migrating Projects .....	16
2.4 Using New Project from Example .....	16
2.5 Using a Makefile Project.....	17
2.6 Debugging Projects .....	18
2.7 Graphical Configuration of Drivers .....	20
2.8 Integrating Green Hills Multi and IAR Compiler into S32DS .....	22

# Chapter 1

## Introduction

The S32 Software Development Kit (S32 SDK) is an extensive suite of robust hardware interface and hardware abstraction layers, peripheral drivers, RTOS, stacks and middleware designed to simplify and accelerate application development on NXP S32K microcontrollers.

This manual explains how to use the S32 SDK product, including the use with S32 Design Studio, for easier application creation and configuration with graphical user interface.

### 1.1 System requirements

Hardware	<ul style="list-style-type: none"> <li>• 1.8 GHz processor</li> <li>• 2 GB of RAM</li> </ul>
Operating System	Microsoft® Windows® 7 or higher
Gnu Make	For usage without S32 Design Studio.
Supported Compiler	For usage without S32 Design Studio. See release notes for supported versions.
Disk Space	Approximately 3 GB of free disk space (when installing standalone)

#### NOTE

The S32 Design Studio software development tools requirements are covered by their own Quick Start Guide. S32CT graphical configuration tool can only be used with S32 Design Studio.

### 1.2 Installing S32 SDK

#### 1.2.1 Bundled in S32 Design Studio

S32 SDK is delivered bundled in the S32 Design Studio. In this case it's already configured and ready to use.

## 1.2.2 Standalone

S32 SDK is also delivered through a standalone installer. Using the standalone installer is recommended when using a compiler which is not supported in S32 Design Studio or when the graphical interface is not required. In this case the installer can configure an existing S32 Design Studio to use the configuration files delivered in the installer.

If the integration with the S32 Design Studio is not needed the path to S32 Design Studio can be left empty – and in this case, only the S32 SDK will be installed and configured.

## 1.3 Release Notes

Before using the S32 SDK, read the release notes. These notes contain important information about last-minute changes, bug fixes, incompatible elements, or other topics that may not be included in this manual. The product comes with the release notes installed.

## 1.4 Terminology

The following are some of the terms used in the document.

**Table 1-1. Terminology**

Term	Description
S32 SDK	Software development kit that provides comprehensive software support for NXP S32 devices. The S32 SDK includes a Hardware Abstraction Layer (HAL) for each peripheral and peripheral drivers built on top of the HAL. S32 SDK also contains the latest available RTOS kernels, a LIN stack and SBC middleware to support rapid development on supported S32K devices.
S32CT	Rapid application design tool targeted for NXP microcontrollers providing the following key features: <ul style="list-style-type: none"> <li>• A Graphical User Interface which allows an application to be specified by the functionality needed.</li> <li>• An application created from Embedded Components encapsulating initialization and functionality of basic elements of embedded systems.</li> <li>• An automatic code generator which creates tested and optimized C code which is tuned to your application needs and the selected NXP device.</li> <li>• A built-in knowledge base, which immediately flags resource conflicts and incorrect settings, so errors are caught early in design cycle allowing you to get to market faster with a higher quality product.</li> </ul>

## Chapter 2

# Working with Projects

This chapter explains how to use the S32 Design Studio to create and work with S32 SDK projects.

A project organizes files and various compiler, linker, and debugger settings associated with the applications or libraries you develop. S32 New Project wizard can be used to create projects that group these files and settings into build and launch configurations.

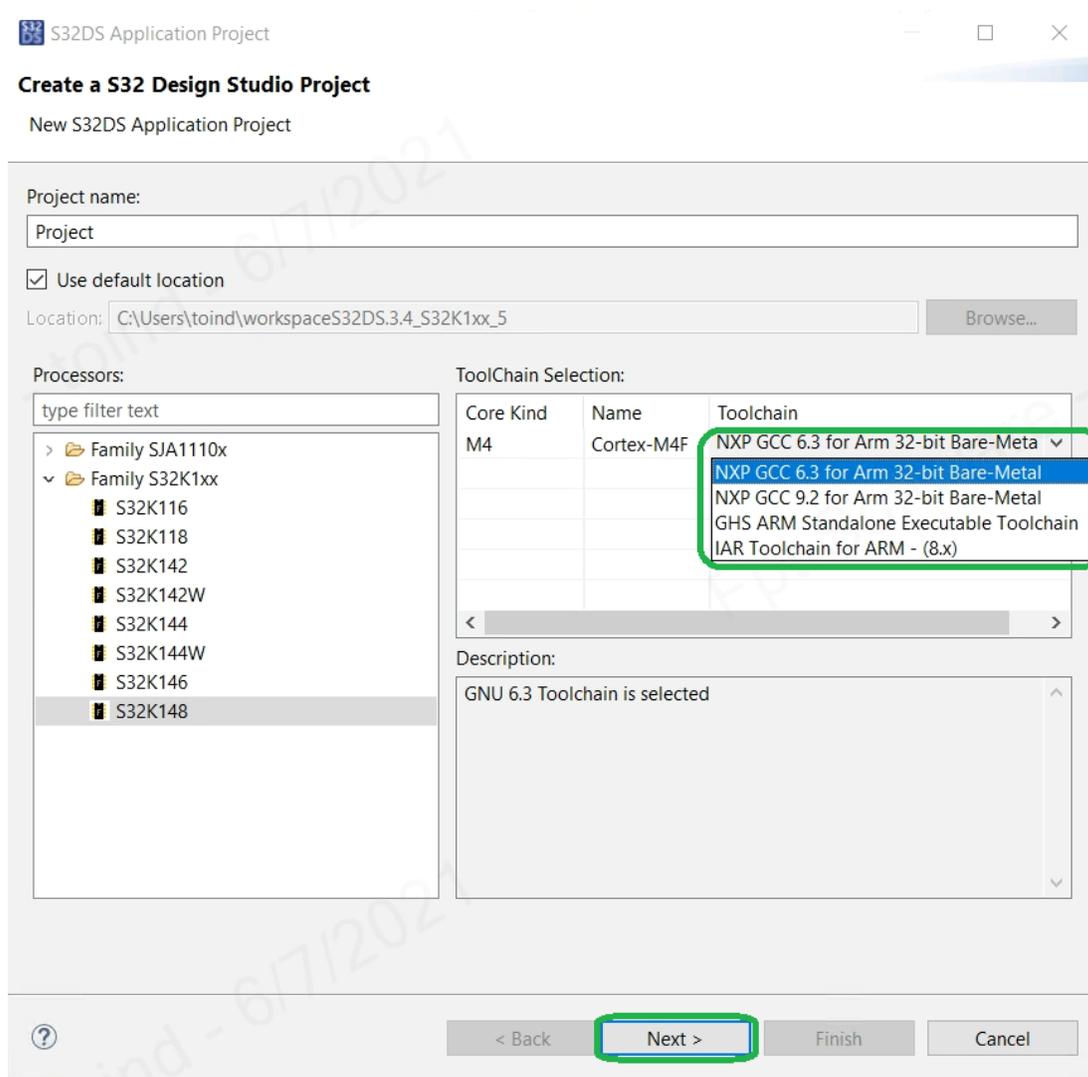
### 2.1 Creating and building S32DS Application Project

The New S32DS Project wizard help you to quickly create new projects. The wizard generates a project with placeholder files and default settings (build and launch configurations) for specified target. After the project has been created, you can easily change any default setting to suit your needs.

To create a S32 SDK project using the **New S32DS Application Project** wizard:

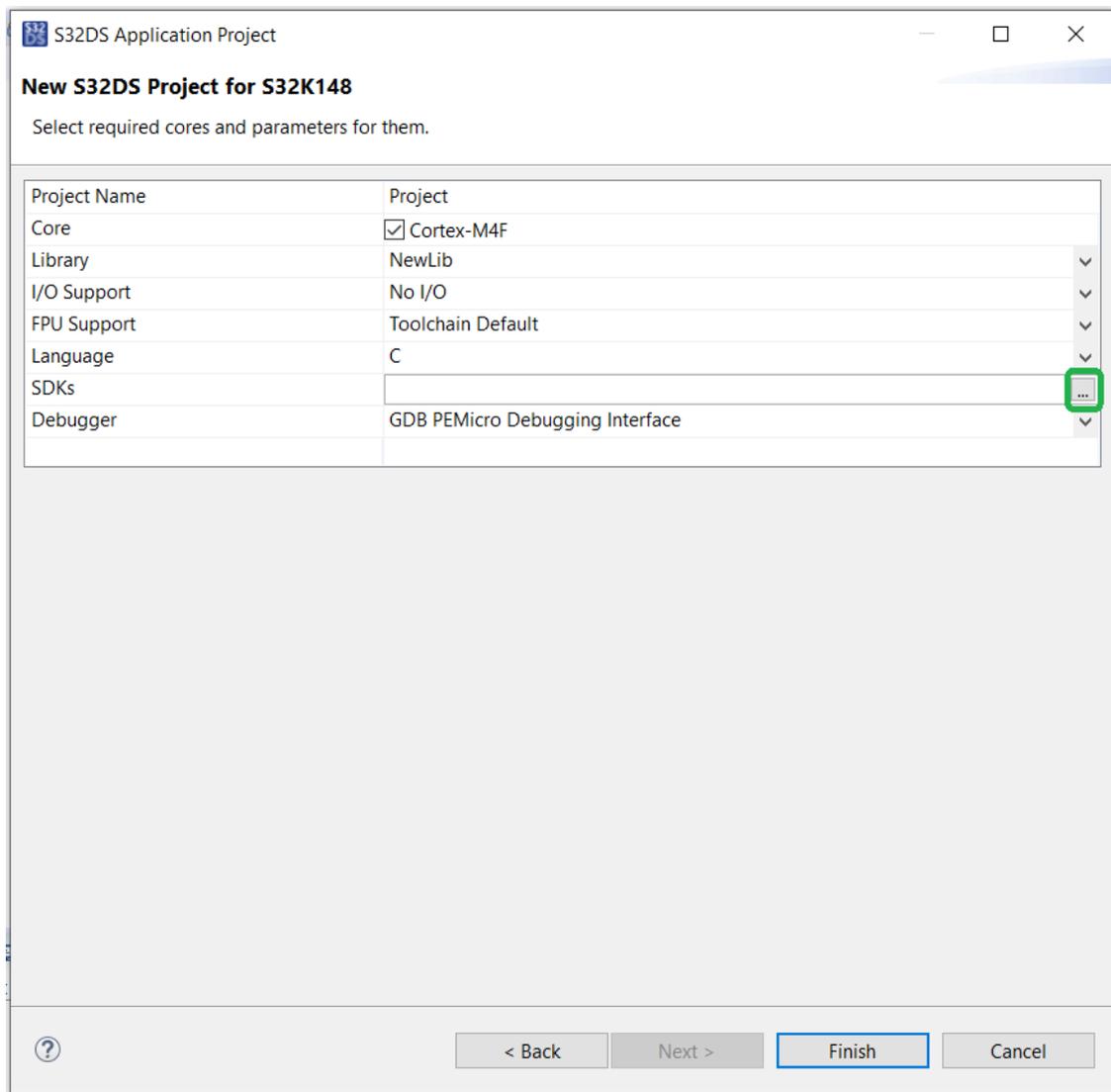
1. Launch the S32 Design Studio. Please refer to S32 Design Studio documentation.
2. Select **File > New > S32DS Application Project**, from the IDE menu bar.

The **S32DS Application Project** page of the **S32DS Application Project** wizard appears.



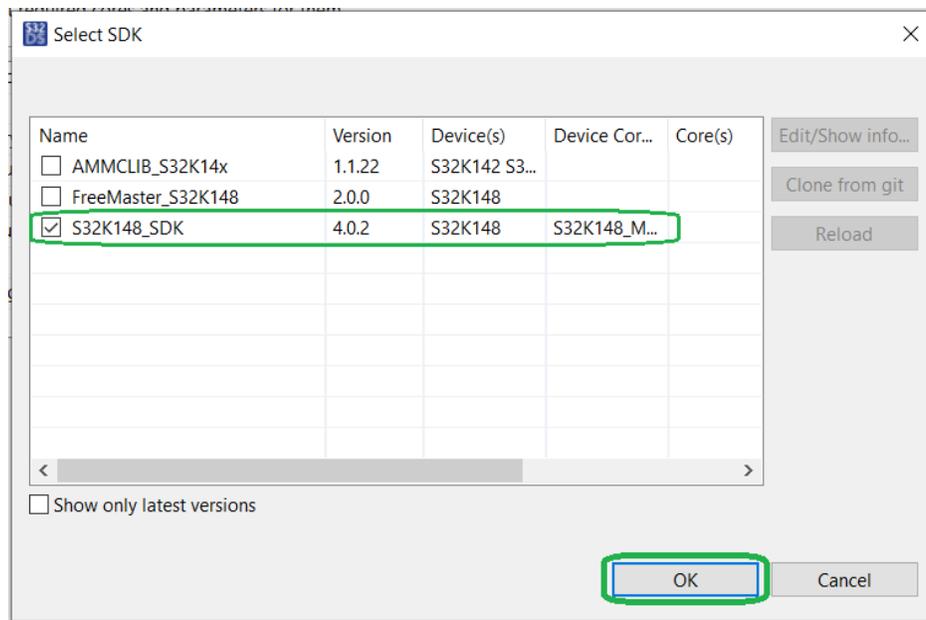
**Figure 1. S32DS Application Project page**

3. Specify a name for the new project. For example, enter the project name as `Project1`.
4. Select the Processor you want to use from **Processors** tab.
5. Select the desired toolchain. Currently, SDK package supports creating project with GCC 6.3 for Arm, GHS ARM and IAR Compiler. By default, only “NXP GCC 6.3 for Arm 32-bit Bare-Meta” is integrated into S32DS 3.4. For other compilers, please refer to **Chapter 2.8**
6. Click **Next**, The **New S32DS Project for <CPU\_NAME>** page of the **New S32DS Application Project** wizard appears



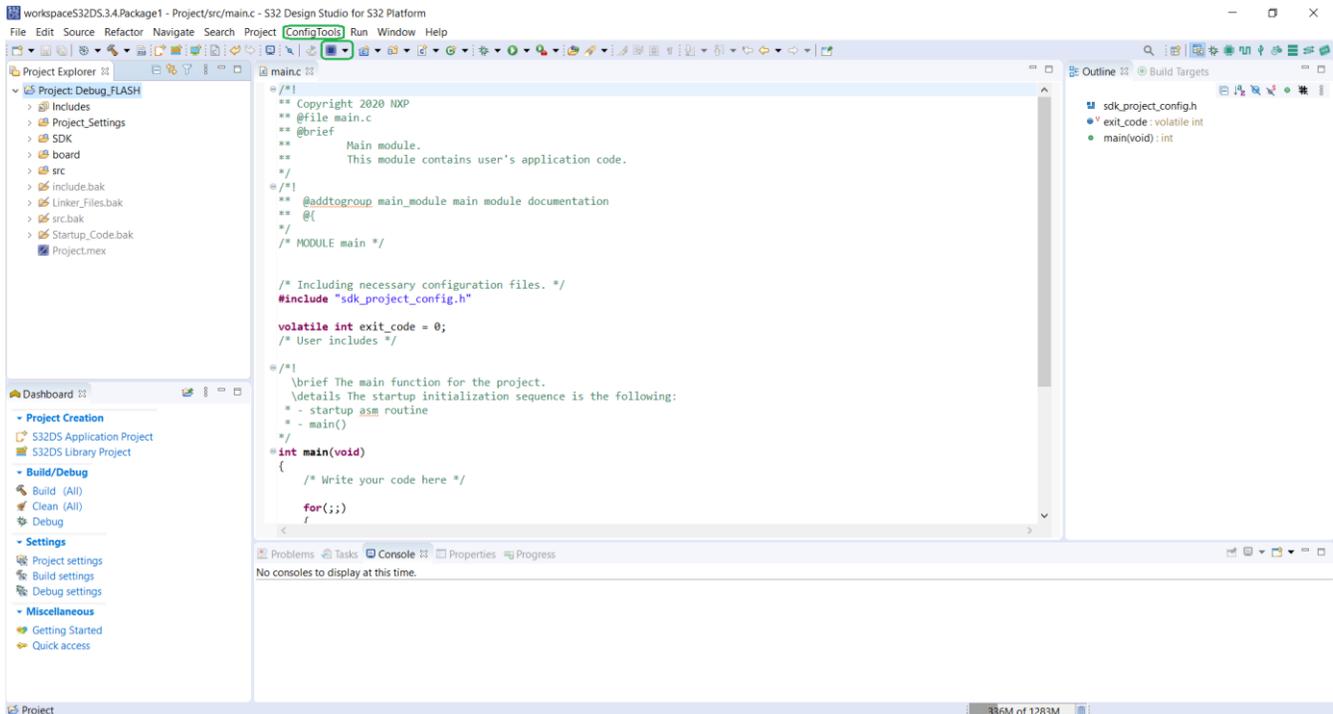
**Figure 2. New S32DS Project for <CPU\_NAME> page**

7. Click on  button to select from available SDKs, The **Select SDK** page of the **New S32DS Application Project** wizard appears



**Figure 3. Select SDK page**

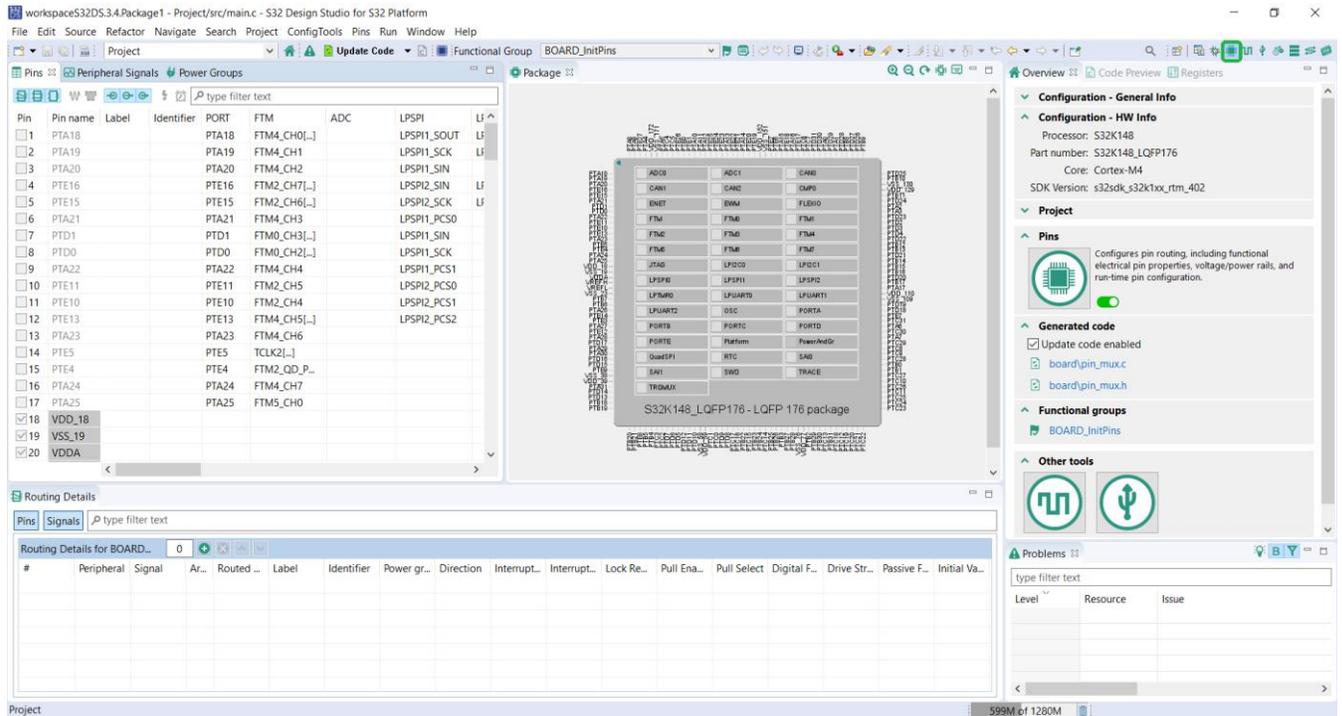
8. Select the latest version of **SDK**, and then click **OK**.
9. Review the settings and click **Finish**, project should be created and default view should be presented.



**Figure 4. Default view after project creation**

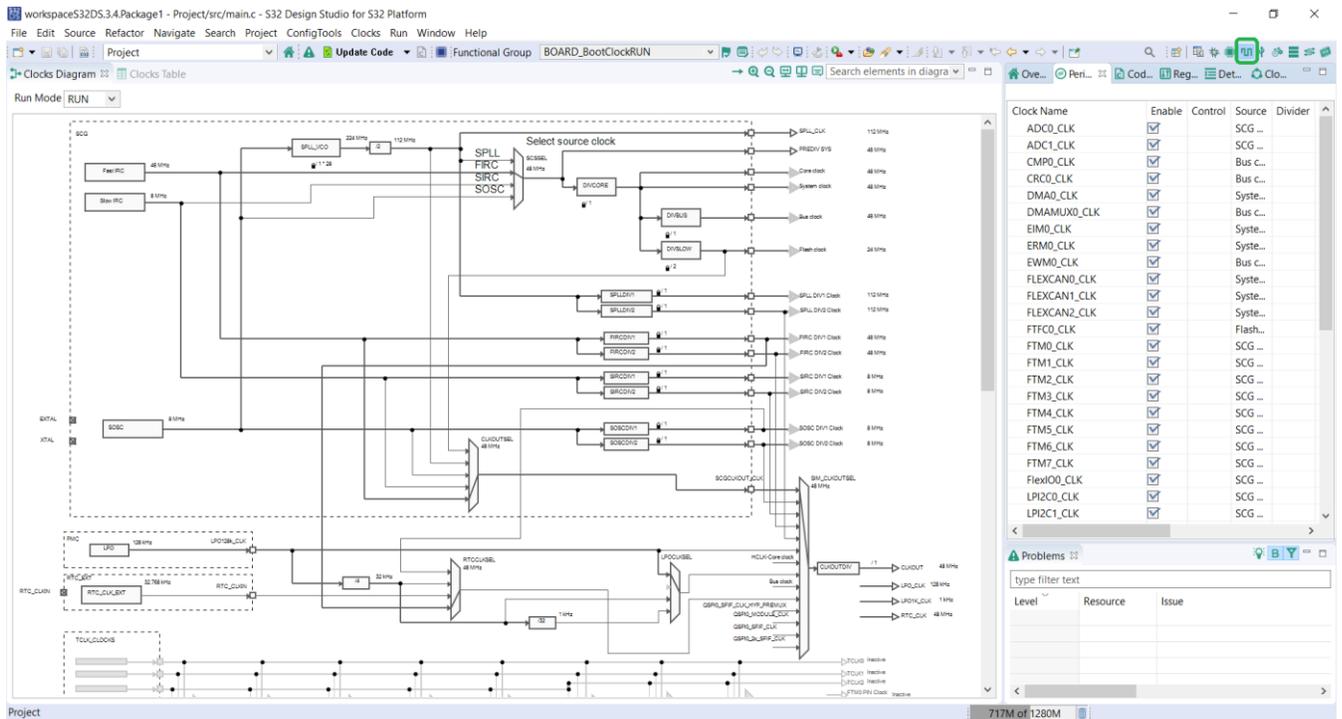
10. S32 Configuration Tool components can be observed in the Menu Toolbar “ConfigTools” or in the “Quick Access” bar.

11. Select the “Pins Tool” by clicking 



**Figure 5. Default view for Pins Tool**

12. Select “Clocks Tool” by clicking 



**Figure 6. Default view for Clocks Tool**

### 13. Select the “Peripherals Tool” by clicking

**Configuration - General Info**

**Configuration - HW Info**  
 Processor: S32K148  
 Part number: S32K148\_LQFP176  
 Core: Cortex-M4  
 SDK Version: s32sdk\_s32k1xx\_rtm\_402

**Project**

**Peripherals**  
 Configures the initialization of the SDK peripheral drivers.

**Generated code**  
 Update code enabled  
 board\sdk\_project\_config.h

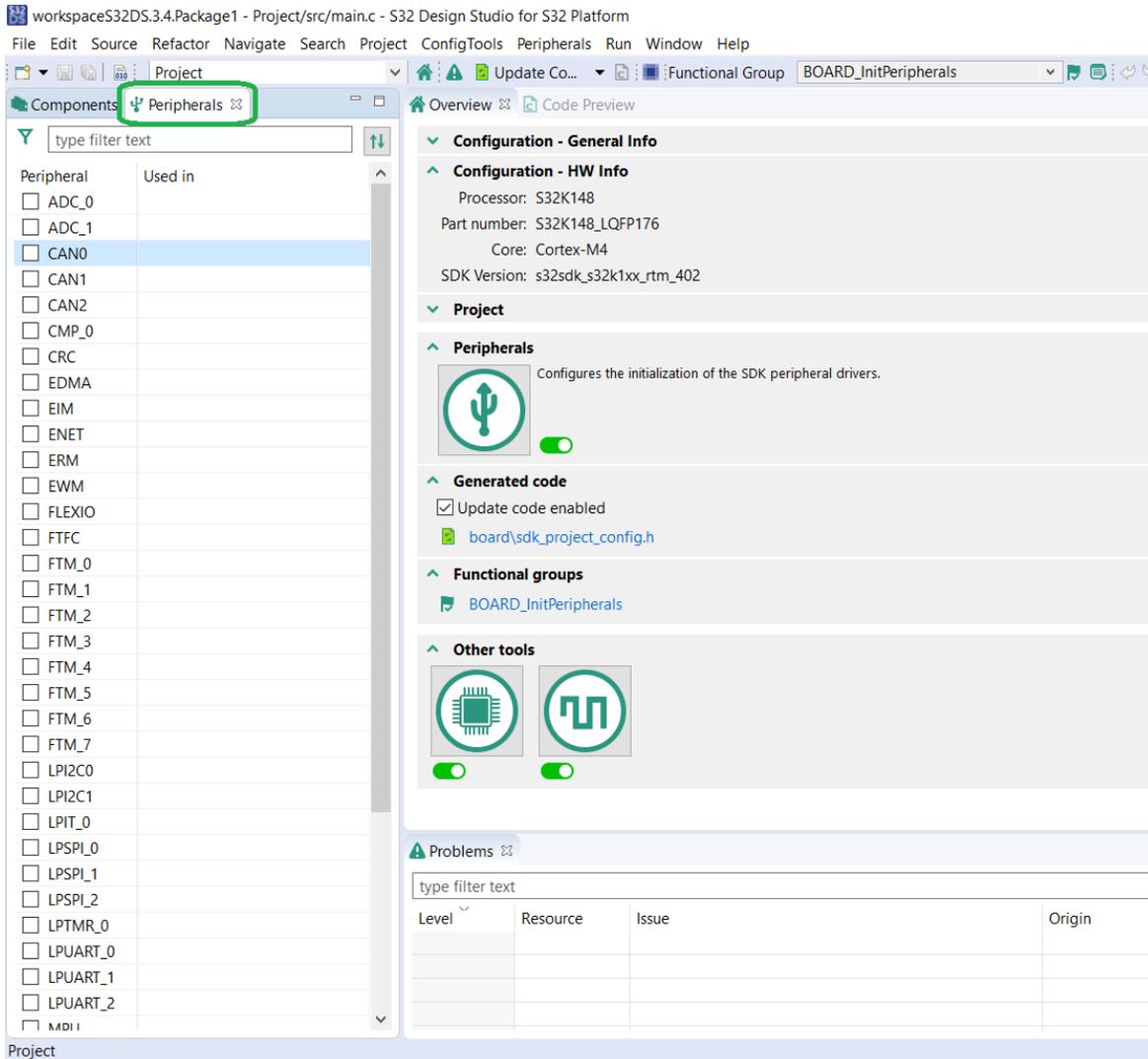
**Functional groups**  
 BOARD\_InitPeripherals

**Other tools**

Level	Resource	Issue	Origin	Target	Type

**Figure 7. Default view for Peripherals Tool**

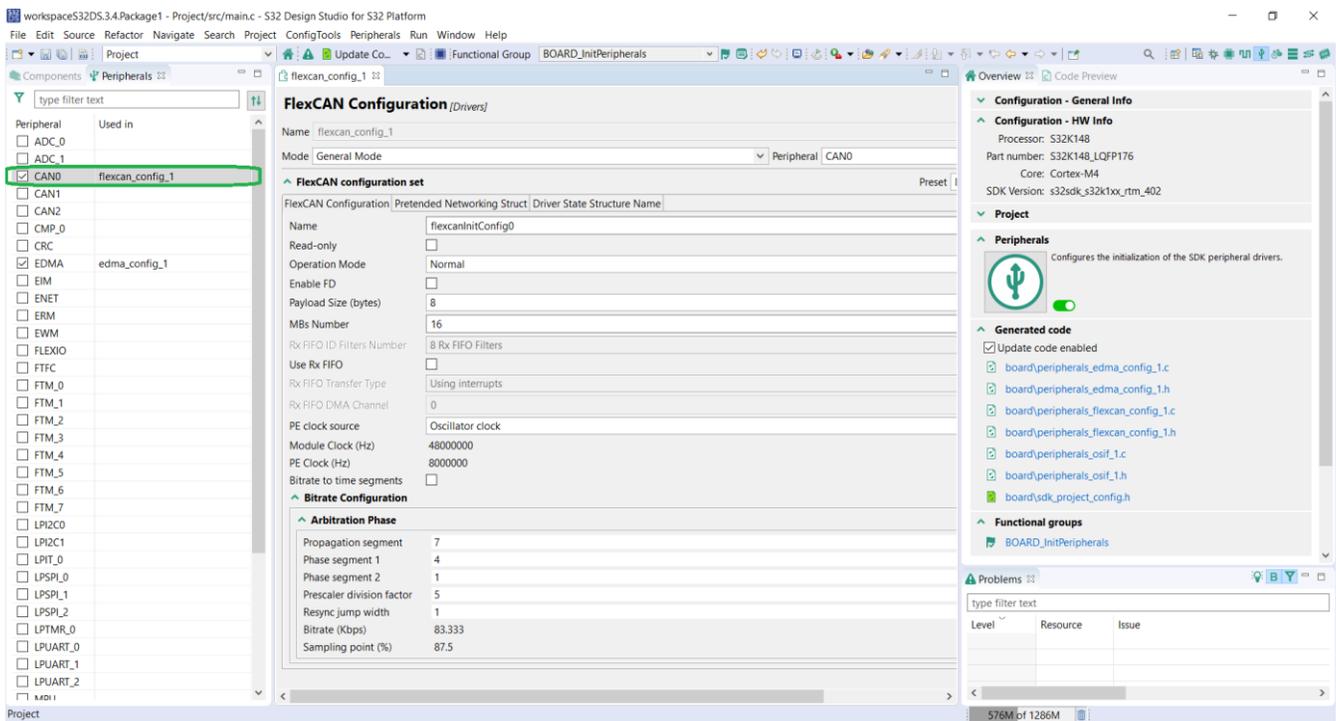
14. Select the  Peripher... view in the top left corner.



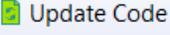
**Figure 8. Peripheral list view.**

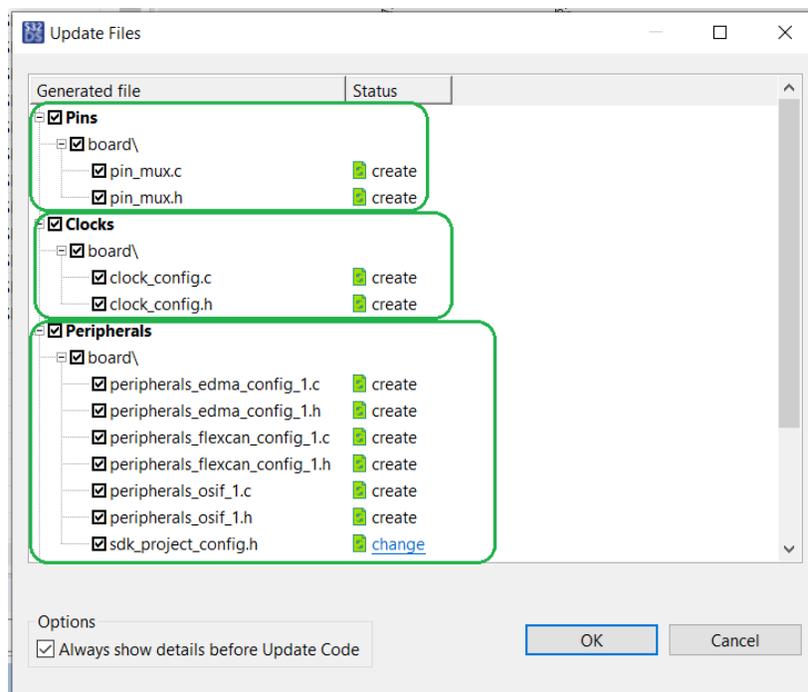
15. Add any component by checking the corresponding checkbox.

16. The configuration component appears. User can configure the selected component parameters. FlexCAN is used as an example.



**Figure 9. Default view for FlexCAN Configuration**

17. Select the “**Update project**” by clicking  and the required drivers source files to the project shall be added.



**Figure 10. Default view for Update Files**

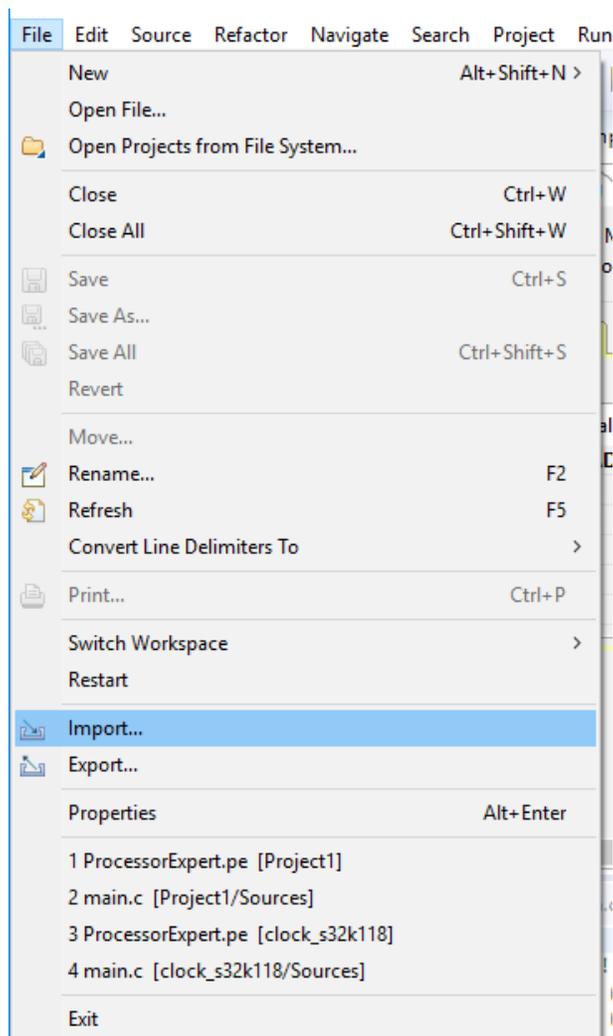
18. Checkboxes for **Pins**, **Clocks** and **Peripherals** shall be checked in order to generate the specified files. Click **OK** and wait for the configuration to be generated.
  
19. Returning to the Default project view, the “board” folder now contains configuration files for **Pins**, **Clocks**, **Peripherals** and **SDK**. The “SDK” folder will contain the selected drivers.

## 2.2 Importing an existing project

This section explains how to import an existing S32 Design Studio SDK project in S32 Design Studio.

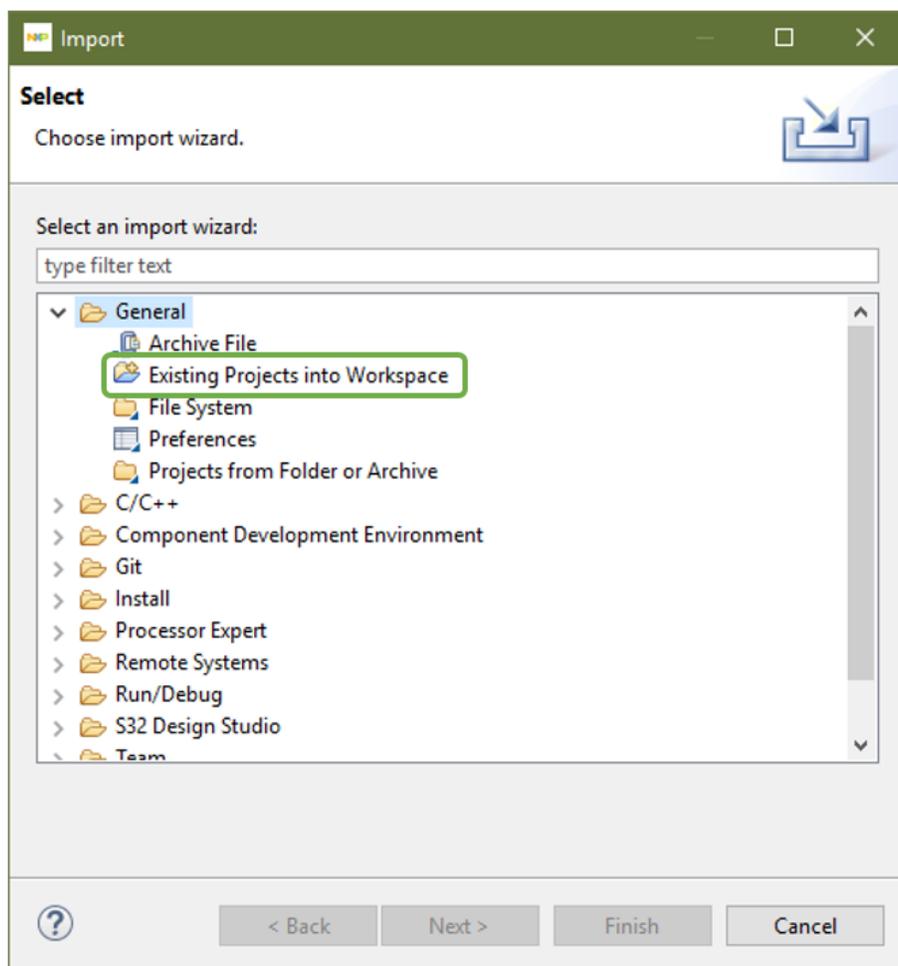
To import an existing project:

1. Select **File > Import**, from the IDE menu. **Choose import source** page will appear.



**Figure 11. Select import option**

2. Expand **General** tree and Select **Existing Projects into Workspace**



**Figure 12. Choose import source**

3. Click **Next**. The Import projects screen appears.
4. Click **Browse** and select the **example** folder from SDK installation directory to search for an existing Eclipse project.

**NOTE**

When using the S32 SDK bundled inside S32 Design Studio the example folder is located in `<S32 Design Studio install>\S32DS\S32_SDK_<RELEASE_VERSION>\examples`

5. Select the project you want to import in your Workspace (It is useful that **Copy to Workspace** is selected in this window. In this way the original project will be preserved).
6. Click **Finish**. The imported project will appear in the Project Explorer view. Similar to **Default view after project creation**.

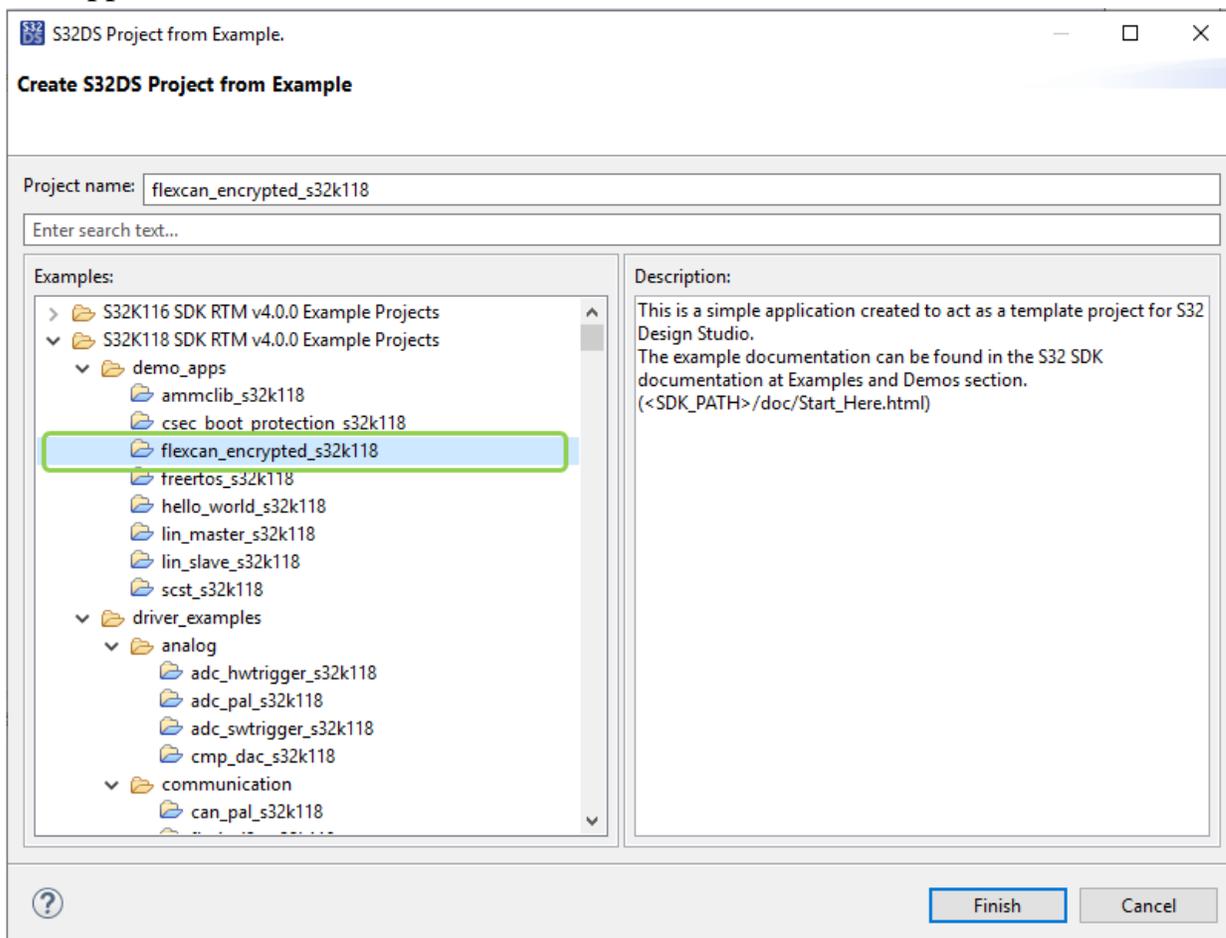
## 2.3 Migrating Projects

S32 Design Studio 3.4 supports migrating project from **SDK RTM 3.0.x releases (PEX)** to **SDK RTM 4.0.2 release (S32CT)**.

## 2.4 Using New Project from Example

S32 Design Studio provides support for accessing the examples from the S32 SDK using the **New Project from Example**. To use this feature follow the next steps:

1. Select **File -> New -> S32DS Project From Example**. The following window will appear.



**Figure 13. S32DS Project from Example**

2. Select the desired project from the release version and click on **Finish**. The project will be copied in your workspace.

## 2.5 Using a Makefile Project

To build a Makefile project without S32 Design Studio the system needs to have a make utility (GNU version 3.0 and above, or equivalent) and a supported compiler. Please refer to S32 SDK Release Notes for a list of supported compilers and their supported version. Example projects are delivered for all supported compilers.

The makefile projects assumes that the make utility and corresponding compiler are included in path.

1. Open a command line window (Start->Run->cmd.exe)
2. Navigate to project folder - makefile project are locate in example folder.

### NOTE

Please check Release Notes or S32 SDK documentation for list of available makefile projects

3. Execute **make all** command. This, depending on the used makefile, will generate one or more elf files.

### NOTE

For not relying on System PATH variable the user can temporary modify the **PATH** variable as follows:

```
set PATH=%PATH%;<path to make>;<path to compiler>
```

4. The elf files can be deployed using one of the supported debugger. Please refer to debugger documentation for flashing and debugging the application.

Example:

```

S:\>cd examples\S32K144\demo_apps\blinking_LED\GCC-MKF
S:\examples\S32K144\demo_apps\blinking_LED\GCC-MKF>make all
=====
Checked for uname, found: MINGW32_NT-6.1
Assuming Unix like environment
=====
Compiler found in PATH
=====
Creating directory for object files
=====
Compiling obj/main.o
=====
Compiling obj/system_S32K144.o
=====
Compiling obj/startup.o
=====
Compiling obj/pcc_hal.o
=====
Compiling obj/port_hal.o
=====
Compiling obj/startup_S32K144.o
=====
Linking to app_ram.elf
=====
Linking to app_flash.elf
=====
Build complete!

S:\examples\S32K144\demo_apps\blinking_LED\GCC-MKF>dir
Volume in drive S is Primary
Volume Serial Number is BC74-DCFE

Directory of S:\examples\S32K144\demo_apps\blinking_LED\GCC-MKF

02/09/2017  04:50 PM    <DIR>          .
02/09/2017  04:50 PM    <DIR>          ..
02/09/2017  04:50 PM              56,054 app_flash.elf
02/09/2017  04:50 PM              22,485 app_flash.map
02/09/2017  04:50 PM              55,681 app_ram.elf
02/09/2017  04:50 PM              21,195 app_ram.map
02/07/2017  05:12 PM               4,149 Makefile
02/07/2017  05:12 PM               3,747 Makefile-arm
02/09/2017  04:50 PM    <DIR>          obj
                6 File(s)      163,311 bytes
                3 Dir(s)  282,046,857,216 bytes free

```

Figure 14. Build the project successfully.

## 2.6 Debugging Projects

When you use the **S32 Design Studio** to create a new project, the wizard sets the debugger settings of the project's launch configurations to default values. You can change these default values based on your requirements.

To debug a project, perform these steps.

1. Launch the IDE and have a project opened and selected.

## NOTE

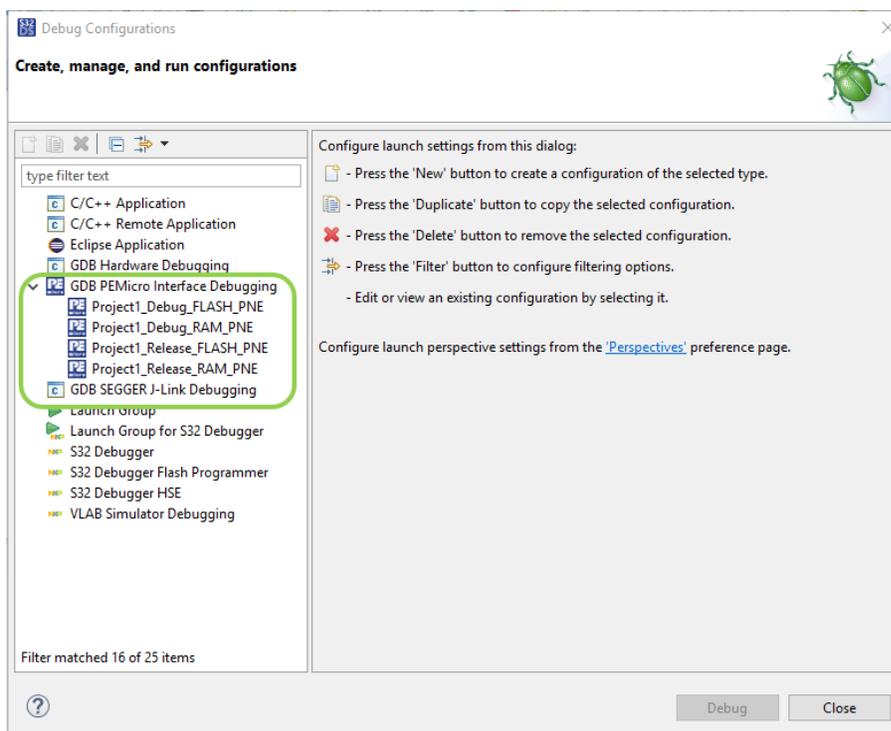
Please ensure that the current perspective is **C/C++** or **Debug**. Current perspective is indicated usually in upper right corner in eclipse window. Use **Window -> Open Perspective** to change current eclipse perspective. Use **Window-> Reset perspective** to reset all the Views to their default location and visibility.

- Click  The **Launch Configuration Selection** dialog appears.  
Alternatively, you can select **Run > Debug Configurations** from the IDE menu bar.

## NOTE

The launch/debug configurations are populated with the default settings; these might need to be adjusted to accommodate various configuration. Please refer to S32 Design Studio documentation for Launch configuration settings.

- Select the launch configuration you want to debug.



**Figure 15. Select launch configuration**

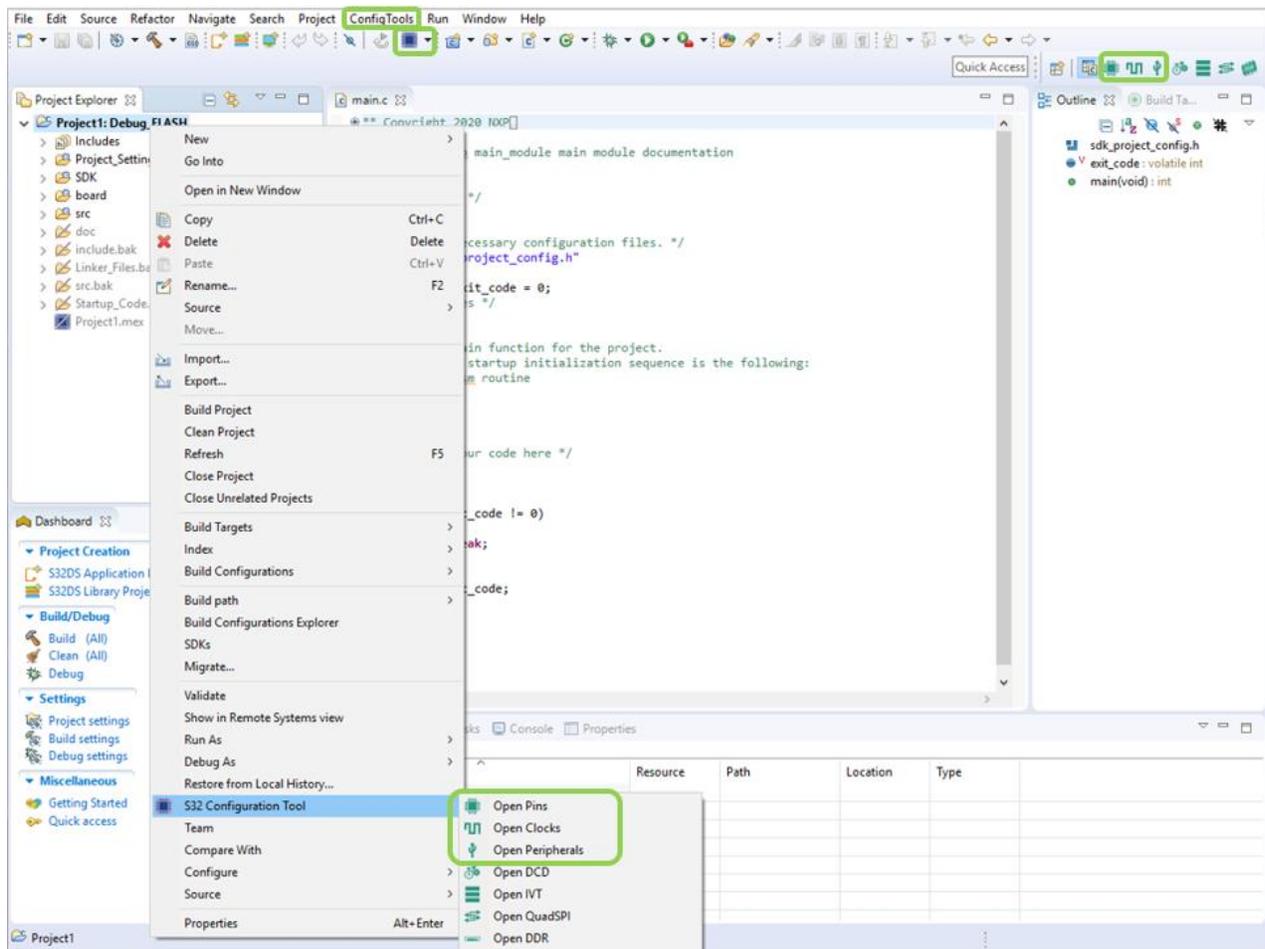
- Click **OK**. The IDE uses the settings in the launch configuration to generate debugging information and initiate communications with the target board.

## 2.7 Graphical Configuration of Drivers

The simplest way to configure the driver is to use the Configuration Tool configurator available in **S32 Design Studio**. Alternatively, the configuration can be written manually in a text editor.

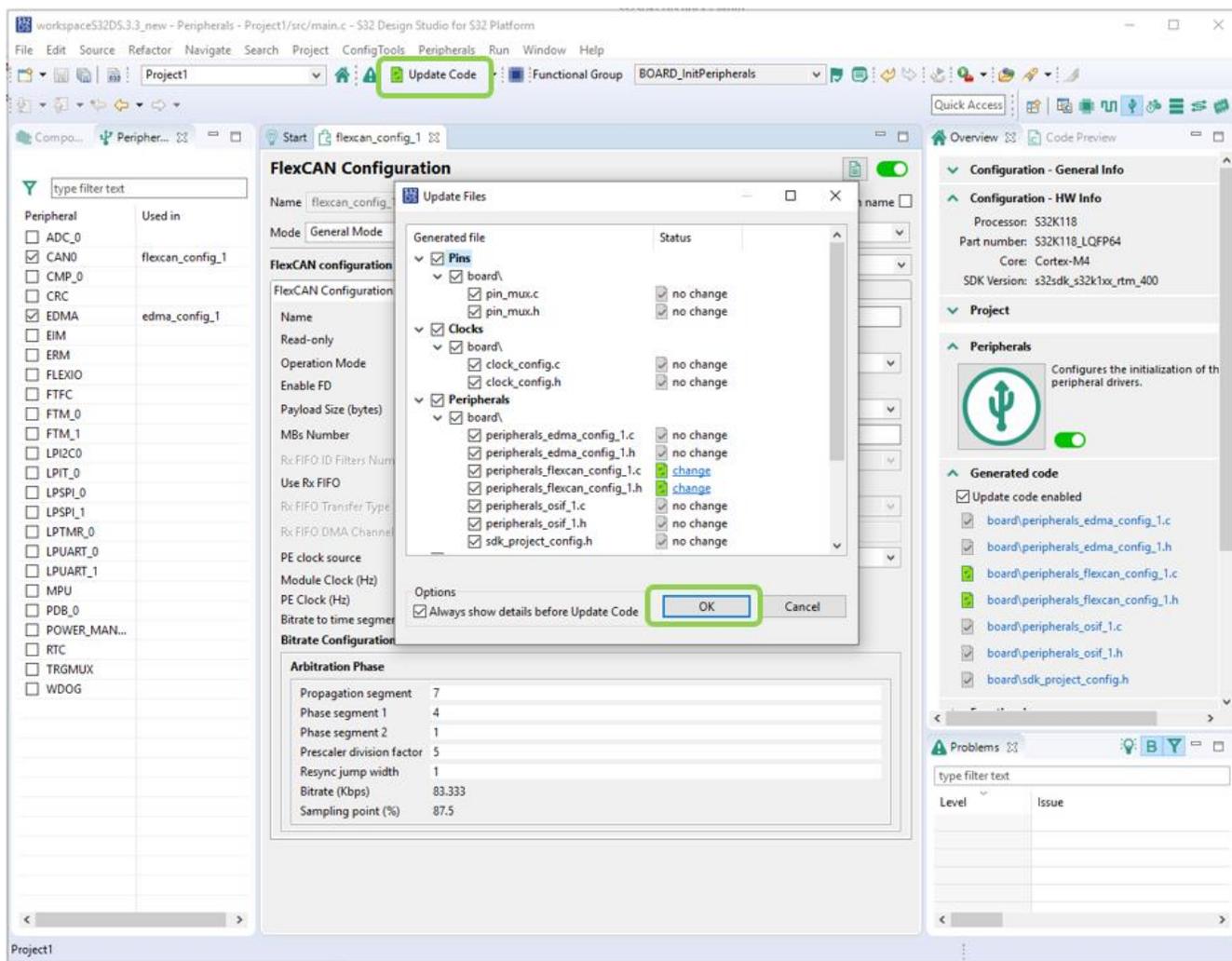
Here are the steps requires to graphically configure a driver:

- Launch the IDE.
- Open Peripherals View (click on S32 Configuration icon or from toolbar ConfigTools -> Peripherals).



**Figure 16. Addition and opening FlexCan component**

- In the Peripherals view, click on the checkbox of the desired module.
- After configuring the parameters from UI, click on “Update Project” (  Update Code ), the “Update Project Files” window appears and click “OK”. The configuration will be generated in “board”. In the same time the FlexCan driver will be added to project in “SDK/platform/drivers”.



**Figure 17. FlexCan configuration and driver**

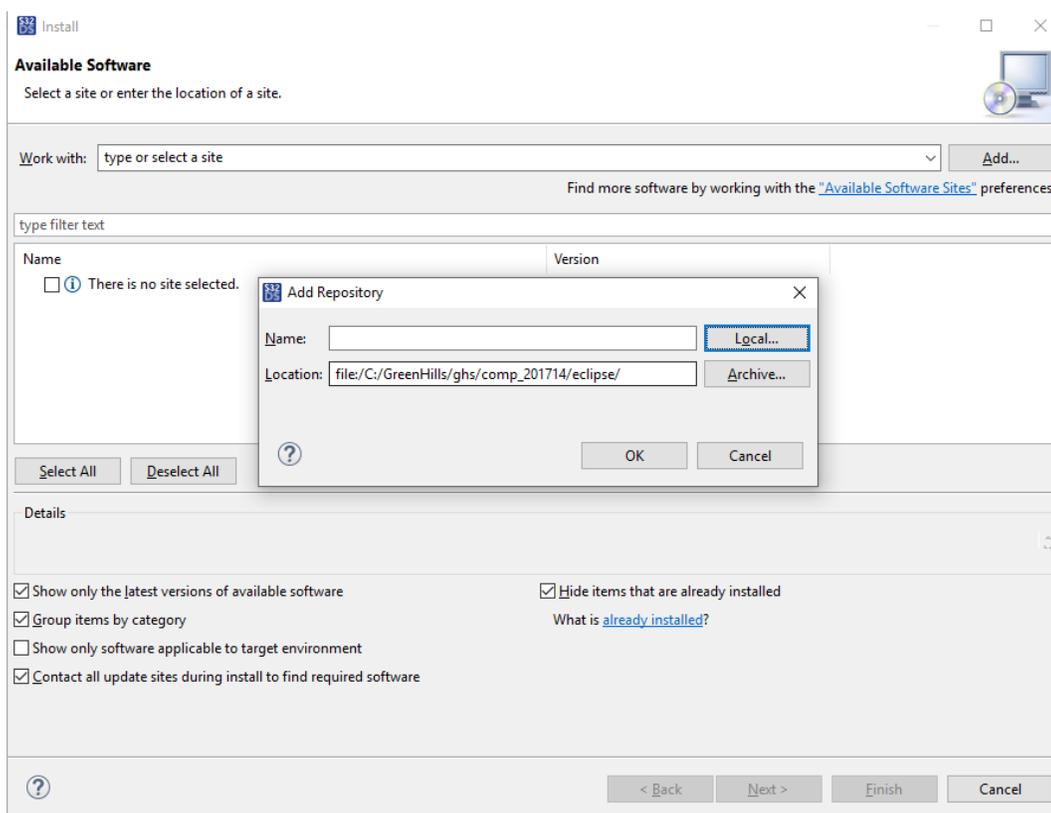
- The generated configuration can be used with driver API

## 2.8 Integrating Green Hills Multi and IAR Compiler into S32DS

The S32 Design Studio supports integrating other compilers into it. In order to integrate them, make sure you already installed desired compiler in your machine with a valid license if required. Otherwise, the integration could not complete

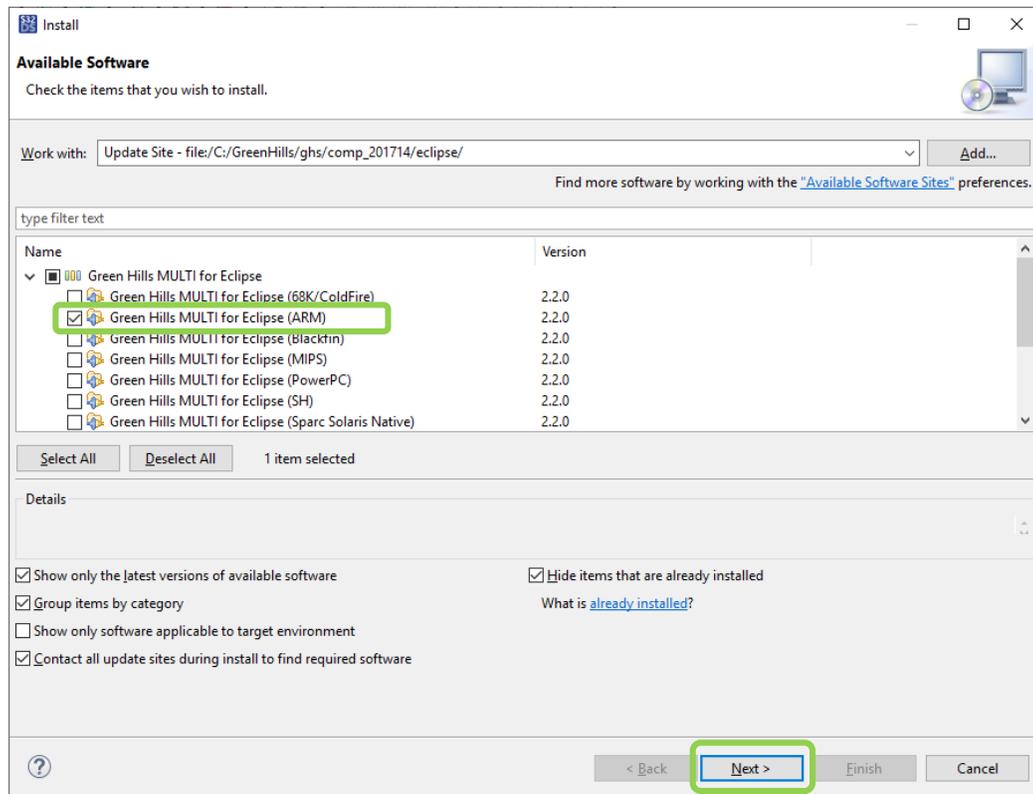
- **Green Hills Multi Compiler**

1. Launch the S32 Design Studio. Go to **Help > Install New Software**
2. In the **Install** dialog box, click to **Add > Local** and select the **eclipse** folder in which GHS was installed to install plug-in for GHS



**Figure 18. Select GHS eclipse plugin location**

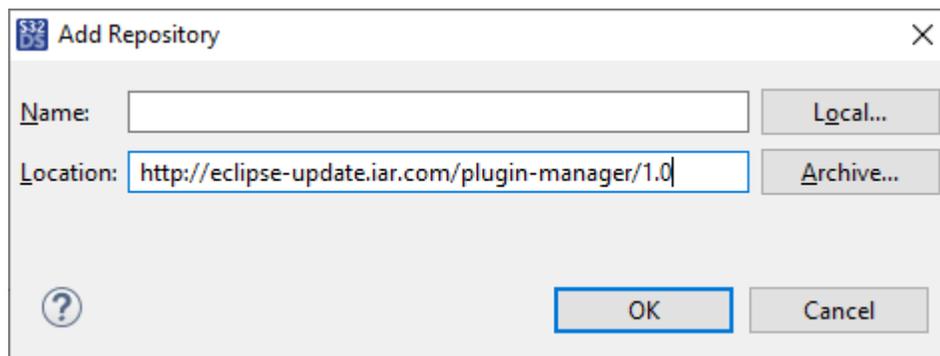
3. Tick to the desired eclipse. In this case, it should be **Green Hills MULTI for Eclipse (ARM)**. Select **Next** and install, then Restart S32DS after completion



**Figure 19. Install GHS eclipse plugin**

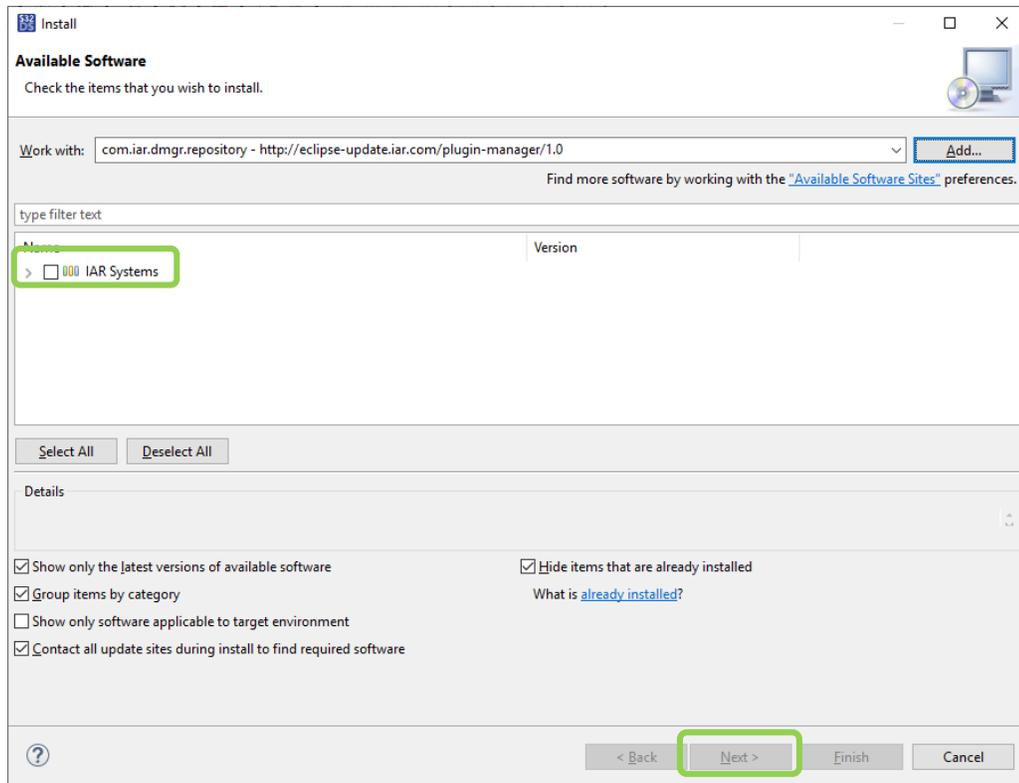
- **IAR Embedded Workbench**

1. Launch the S32 Design Studio. Go to **Help > Install New Software**
2. In the **Install** dialog box, click to **Add** and copy the following link into Location <http://eclipse-update.iar.com/plugin-manager/1.0> and then **OK**



**Figure 20. Add IAR Repository**

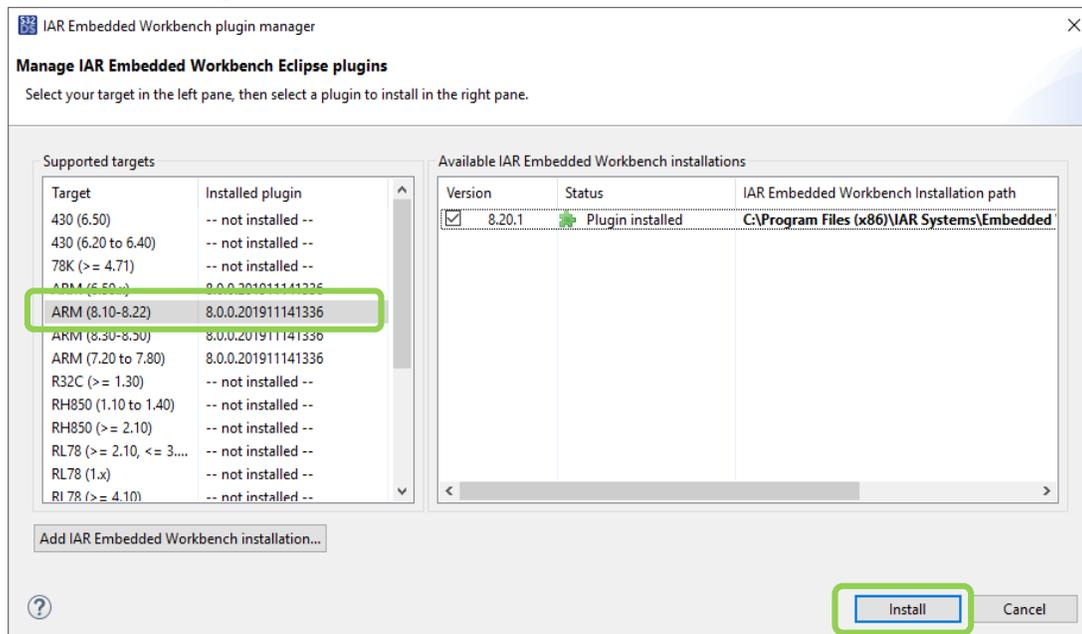
3. Tick to **IAR Systems** and select **Next**, install and Restart S32DS after completion



**Figure 21. Install IAR eclipse plugin manager**

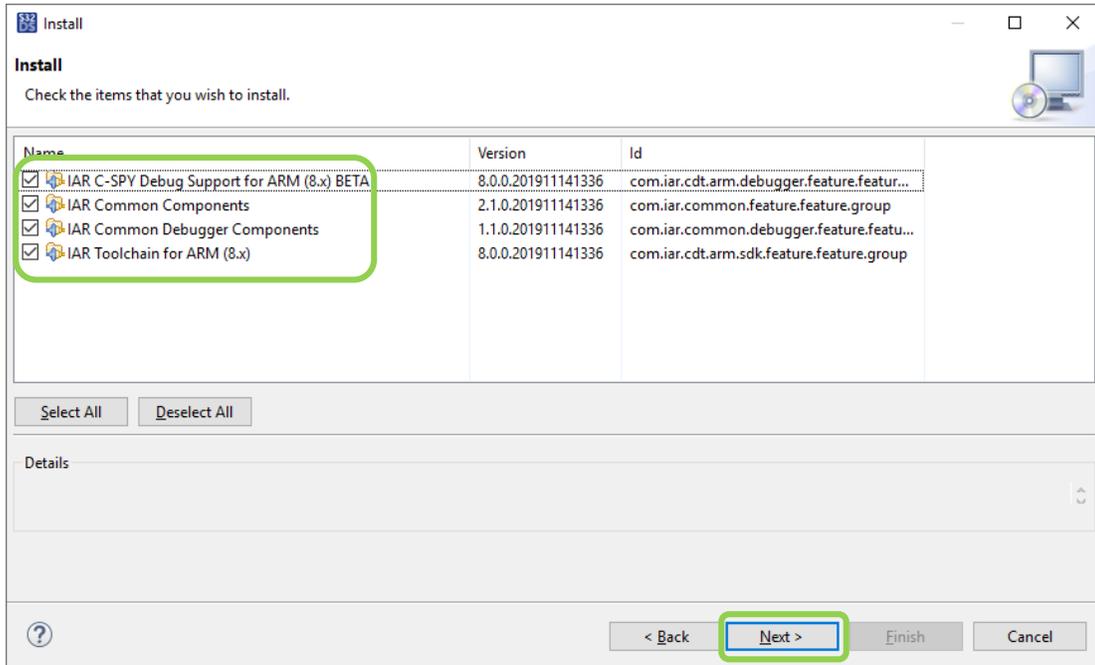
4. Go to **Help > IAR Embedded Workbench plugin manager**

5. In the **IAR Embedded Workbench** dialog box. Select **ARM (8.10-8.22)**. Tick to the IAR version in the right side and then click to **Install**.



**Figure 22. Select IAR Embedded Workbench**

6. After taking a while, an **Install** dialog box will show as below. Select All and Next. Install and Restart S32DS after completion



**Figure 23. Install IAR eclipse plugin**

**NOTE**

Path to IAR should not contain spaces to avoid compiling error, for example: default path of IAR (*:\Program Files (x86)\IAR Systems\Embedded Workbench 8.0*) will not work with eclipse.

***How to Reach Us:***

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals”, must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/about/about-nxp/about-nxp/our-terms-and-conditions-of-commercial-sale:TERMSCONDITIONSSALE](http://nxp.com/about/about-nxp/about-nxp/our-terms-and-conditions-of-commercial-sale:TERMSCONDITIONSSALE).

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorIQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. and Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SafeAssure logo, SMARTMOS, Tower, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2016 Freescale Semiconductor, Inc

© 2017 - 2021 NXP Semiconductors