REAL-TIME DRIVERS (RTD) FOR S32K3XX MCUS

OVERVIEW AND INSTALLATION GUIDE

Automotive Systems & Applications Engineering Team
GPIS-BL, Automotive Processing

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

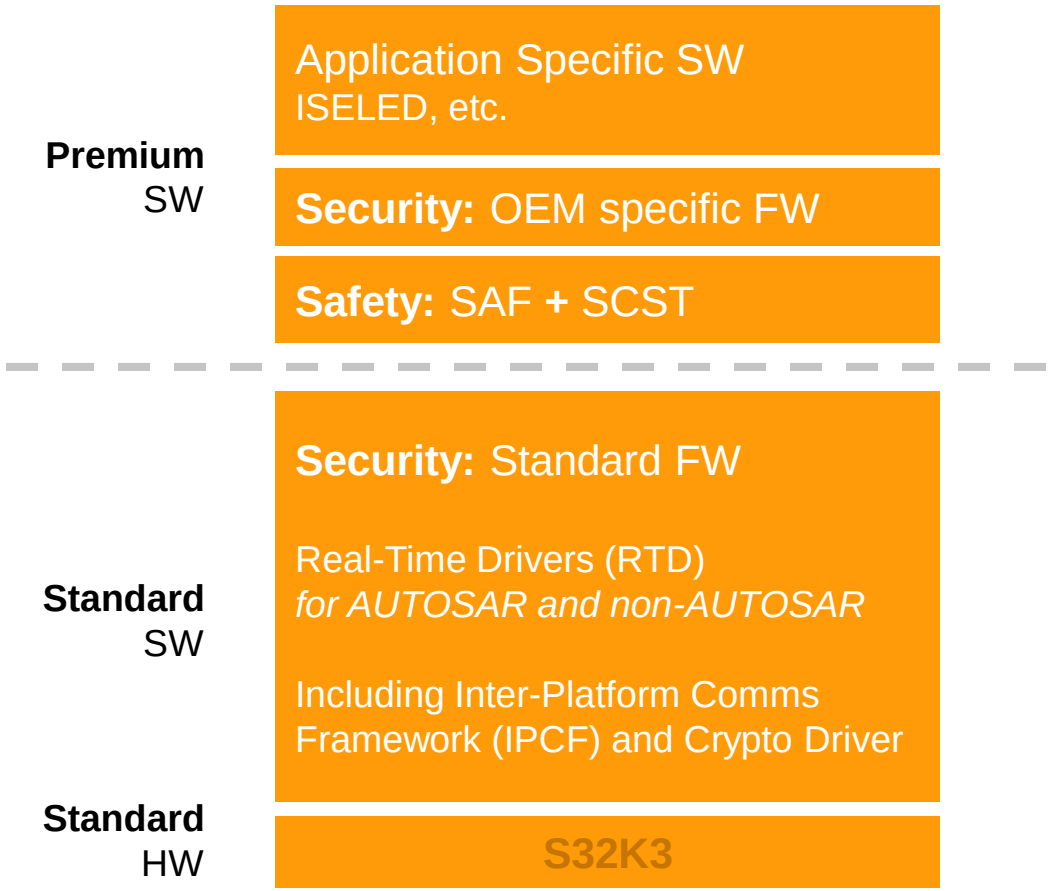
AGENDA

1. S32K3 SW Enablement Overview
2. RTD Architecture Overview
3. RTD Installation
4. RTD Example Projects
5. Create New Projects Based on RTD

S32K3 Software Enablement
OVERVIEW

SECURE CONNECTIONS
FOR A SMARTER WORLD

S32K3 SOFTWARE OFFERING: STANDARD AND PREMIUM



PREMIUM OFFERING:

Application Specific SW: ISELED, etc.
Price adder on top of silicon price for selected PN

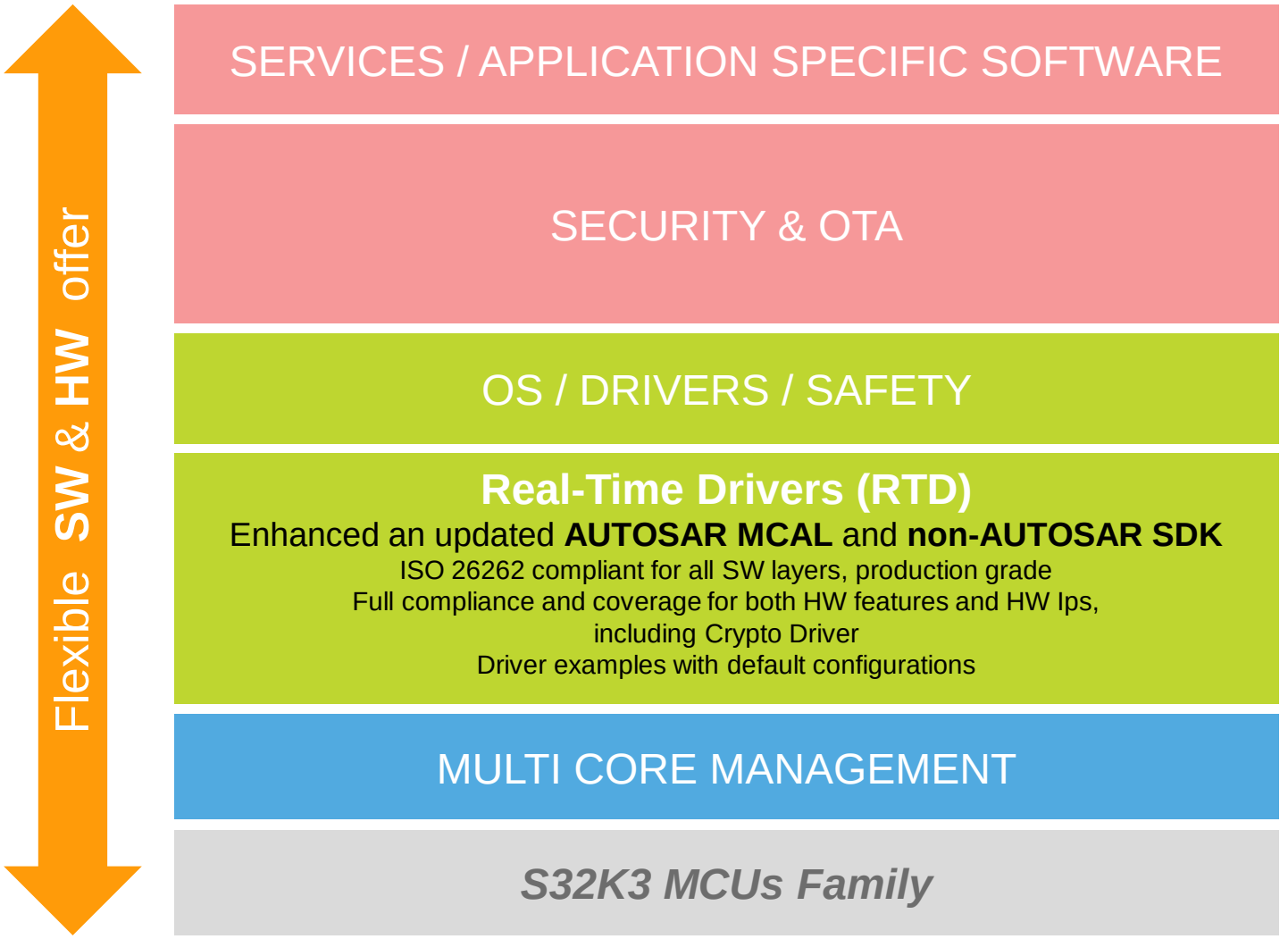
Premium Security: OEM specific firmware
Price adder on top of silicon price for selected PN

Premium Safety:
S32 Safety Software Framework (SAF)
+ Structural Core Self Test (SCST)
One-time license fee for combined SAF + SCST

STANDARD OFFERING:

Included in silicon price

NXP SOFTWARE BASED ON REAL-TIME DRIVERS



Unmatched **HW scalability** across General-Purpose & Integrated Solutions MCUs

combined with

Real-Time Drivers (RTD) flexibility

One SW development environment independently by the project requirements and specifications

One configuration tool and one driver set

MEANING: less time and higher optimization of functionalities

REAL-TIME DRIVERS (RTD)
NEW AND INNOVATIVE DRIVERS SET FOR AUTOSAR AND NON-AUTOSAR SOLUTIONS

Specifically focused on **Real-Time Software**

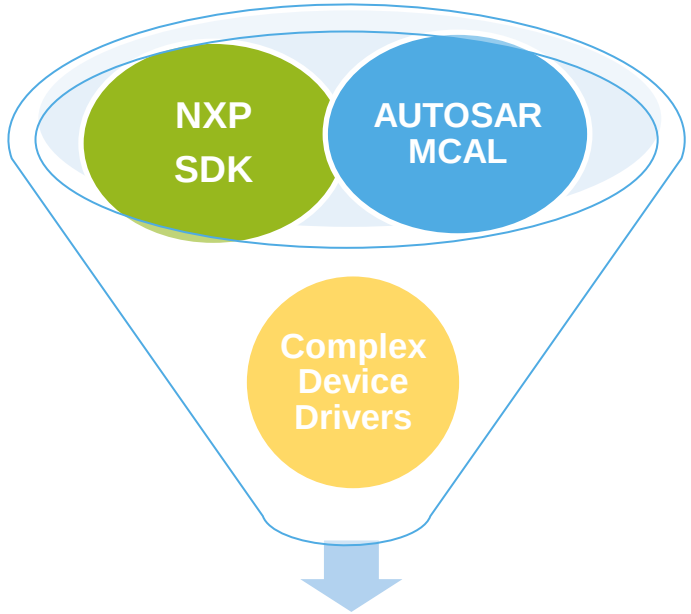
Targeted **for Arm® Cortex®-M core** based MCUs

Single package for each S32 MCU or Processor

For AUTOSAR and NON-AUTOSAR systems

ENHANCEMENTS:

- ISO 26262 Compliance for all SW layers
- AUTOSAR functionalities (e.g. multicore, user mode) are expanded also to non-AUTOSAR environment (previously only available for AUTOSAR)
- Full IP and features coverage for both AUTOSAR and AUTOSAR
- Possible integration on platform level of middleware (FATFS for EEP, FEE for FLS *derived from MCAL)* and stacks (LIN, NFC, TCIP, ..)
- Driver examples with default configurations



Real-Time Drivers (RTD)

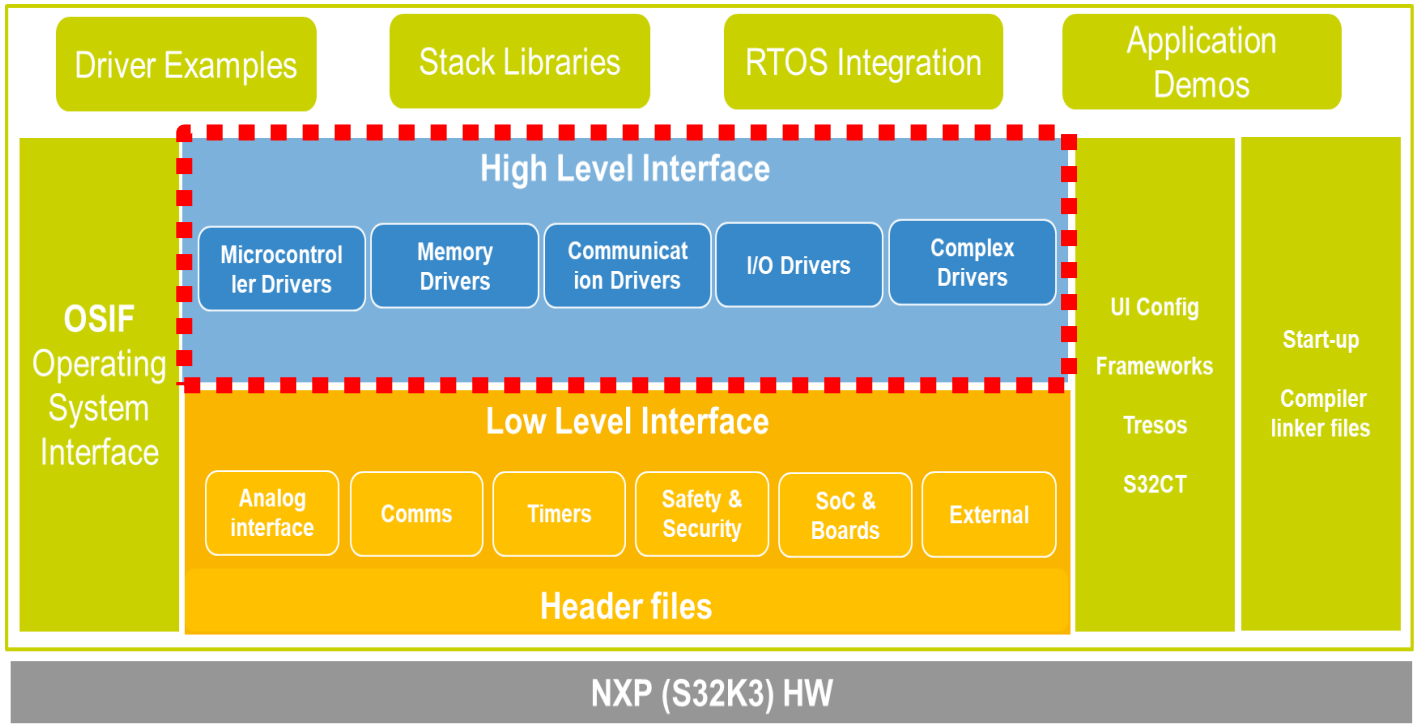
AUTOSAR & non-AUTOSAR

Real-Time Drivers (RTD)
ARCHITECTURE OVERVIEW

SECURE CONNECTIONS
FOR A SMARTER WORLD

REAL-TIME DRIVERS (RTD) SOFTWARE PACKAGE FOR S32 MICROCONTROLLERS AND PROCESSORS

High-Level Interfaces (HLI) based (and enhanced) on former MCAL environment

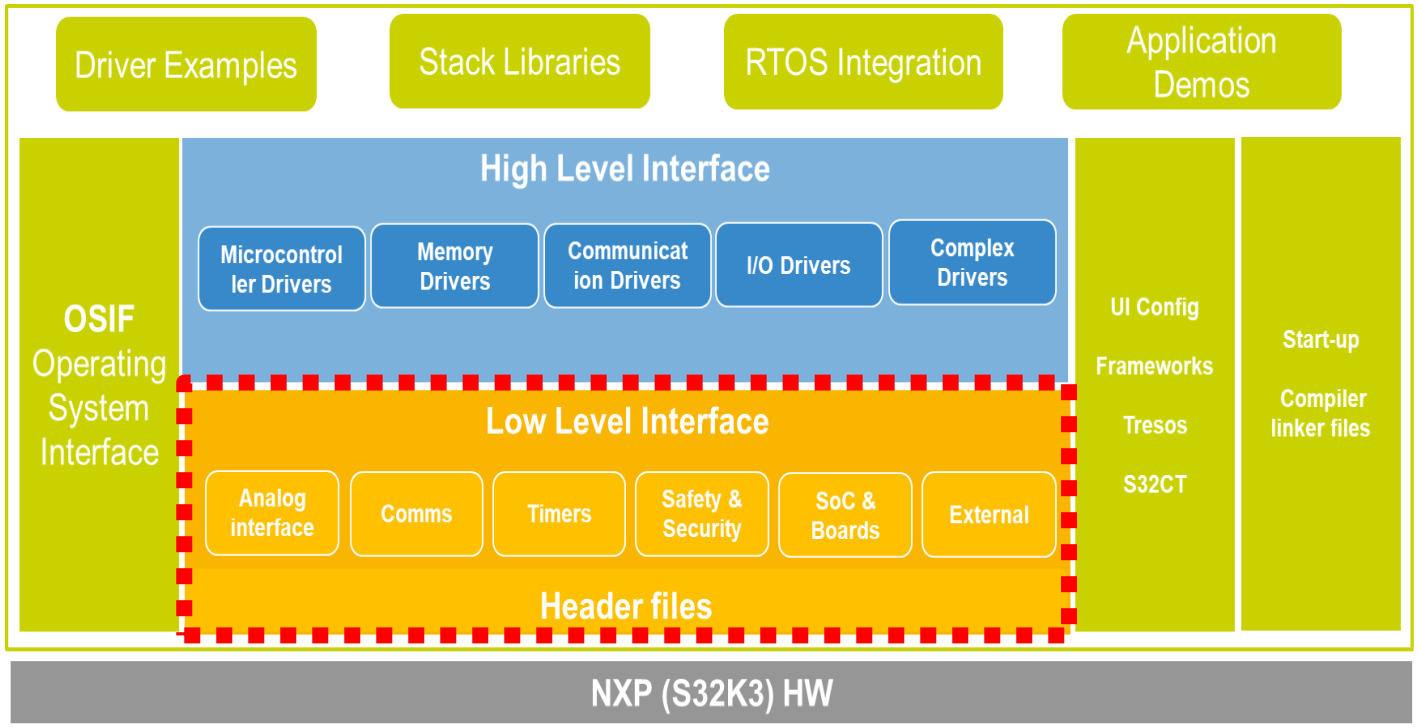


- Production-qualified software abstraction of complex hardware features
- Automotive-grade and production ready: SPICE/CMMI Level 3 compliant, MISRA 2012 tested
- Developed using SPICE Level 3 and ISO 26262 standard compliant process
- Integration with NXP S32 Design Studio (S32DS) IDE
- Supports multiple toolchains: GCC, GHS, IAR
- Full coverage of IPs through *extensions*: extra APIs added to standard ones – e.g. *Adc_EnableCtuControlMode* to support configuration and functions related to CTU control mode of ADC unit
- **AUTOSAR 4.4**:
  - Multicore
  - LIN “follower” support
  - Security: TLS, Key Manager, Security Event Memory
- Documented source code, examples, cookbook & demos for fast application start-up, using drag-drop functionality

*** REAL-TIME DRIVERS to support and improve actual MCAL functionalities**

REAL-TIME DRIVERS (RTD) SOFTWARE PACKAGE FOR S32 MICROCONTROLLERS AND PROCESSORS

Low-Level Interfaces (LLI) based (and enhanced) on former SDK environment



· Automotive-grade and production ready:
SPICE/CMMI Level 3 compliant, MISRA 2012 tested

· Complete drivers offering:
  - Low-level drivers for all MCU peripherals: FlexIO, UART, CAN FD, ISELED, etc.
  - Optional middleware: LIN, TCP/IP, NFC
  - Drivers for complementary NXP ICs: e.g. SBC

· FreeRTOS operating system

· Integration with NXP S32 Design Studio (S32DS) IDE and 3rd party IDEs: KEIL, GHS Multi, IAR

· Supports multiple toolchains: GCC, GHS, IAR

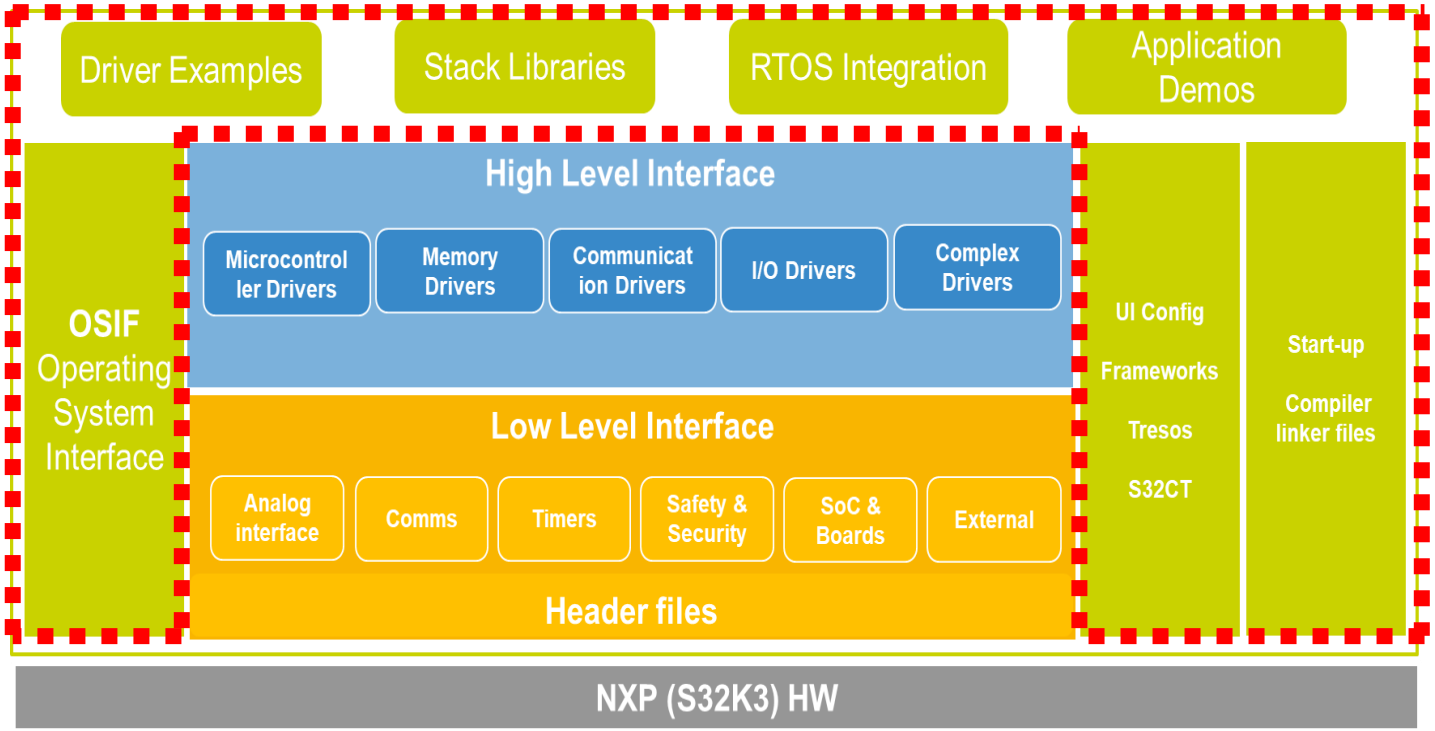
· Documented source code, examples, cookbook & demos for fast application start-up, using drag-drop functionality

*** REAL-TIME DRIVERS to support and improve actual SDK functionalities**

REAL-TIME DRIVERS (RTD) SOFTWARE PACKAGE FOR S32 MICROCONTROLLERS AND PROCESSORS

Additional specific SW packages and Configuration Tools

One configuration tool can be selected for the development: EB tresos or S32 Config Tool (S32CT)
→ *aiming to develop S32CT with AUTOSAR functionalities*



- Stacks and Libraries available in both AUTOSAR and non-AUTOSAR contexts.
  Can be plugged into:
  - High-Level Interface (AUTOSAR compliant)
  - Low-Level Interface

- Demo application code available for:
  - Provided libraries & stacks
  - High-Level Interface (AUTOSAR compliant) layer
  - Low-Level Interfaces layer

*** Additional specific SW package to the REAL-TIME DRIVERS (RTD)**

ERROR MANAGEMENT BETWEEN MCAL/SDK AND RTD

The error detection and reporting mechanism for RTD is tailored for the target application type:

· HL API

- ✓ For the high-level layer, which is mainly intended for usage in AUTOSAR applications, error management follows the standard specifications for **DET** & **DEM**. RTD provides a “stub” implementation of these AUTOSAR modules, which can be used or overwritten by the customer application.
- ✓ Most of the APIs in consisting the AUTOSAR compliant HL API return Std_ReturnType (**E_OK**/**E_NOT_OK**). The specific error can then be retrieved by calling the dedicated APIs in **DEM**/**DET**.

***TIPS**: Development errors are always reported using **DET**;*
*runtime may be reported using **DEM** or **DET**, depending on the impact they have on the application integrity.*

· IP API

The errors reported by the IP layer are still split in two categories:

- **Development errors**: usually parameters checking but not only, these errors are checked using **DevAssert** function; in case an error is detected, this will halt the program execution in the default implementation. The default behavior of **DevAssert** function can also be overwritten by the application. This mechanism is almost identical to the DEV_ASSERT functionality in older SDK, the only improvement being that these statements are now enabled/disabled for each driver separately, as opposed to the SDK approach where this was a global configuration (check the picture below).

- **Runtime errors**: as opposed to the SDK, where all runtime errors reported by drivers were grouped in the generic enumeration called ***status_t***, the RTD define a set of runtime errors per driver. The naming convention for these errors is ***<IP_Name>_Ip_StatusType***, as shown is the example below:
  - ✓ Each driver defines the set of errors that can be reported by the controlled IP; these errors can either be used by the non-AUTOSAR application implemented on top of the IP layer for retrieving the status of the driver, or further fed into the high-level state machine of the layers on top.

CONFIGURATION FILES DIFF BETWEEN MCAL/SDK AND RTD

The configuration data files are now split following a more granular approach to ensure the possibility of using the IP drivers stand-alone.

From a functional point of view, all the data that is needed in an AUTOSAR application will be exported through the **HLD** files, so nothing changes in the application flow.

MCAL S32K1/S32K2	RTD S32K3	Comments
<Mdl>_Cfg.h	**<Mdl>_Cfg.h** **<Mdl>_Ipw_Cfg.h** **<Ip>_Cfg.h**	Contains precompile parameters used in the driver, usually defines and constants, extern declarations and data types
<Mdl>_Cfg.c	**<Mdl>_Cfg.c** **<Mdl>_Ipw_Cfg.c** **<Ip>_Cfg.c**	Static configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by *<Mdl>_Init* function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for Precompile variant.
<Mdl>_PBcfg_<Variant>.c	**<Mdl>_PBcfg _<Variant>.c** **<Mdl>_Ipw_PBcfg _<Variant>.c** **<Ip>_PBcfg_<Variant>.c**	There is one file for each variant. The name of the file contains the name of the variant, as defined in the EcuC. This file contains the configuration structure used by the driver that have variant aware members. Each file contains the configuration parameters for its corresponding variant. All parameters and/or structures that are not variant aware and were generated once in the <Mdl>_Cfg.c file are referenced in the structures from <Mdl>_PBcfg_<Variant>.c files if needed. The configuration structures are used in all variants.
<Mdl>_PBcfg_<Variant>.h	**<Mdl>_PBcfg _<Variant>.h** **<Mdl>_Ipw_PBcfg _<Variant>.h** **<Ip>_PBcfg_<Variant>.h**	It was created to export the extern declaration of each configuration structure, to be used when calling *<Mdl>_Init* in the application. There is one file for each variant. The name of the file contains the name of the variant, as defined in the EcuC.

MAPPING BETWEEN RTD DRIVERS AND S32K3XX PERIPHERALS

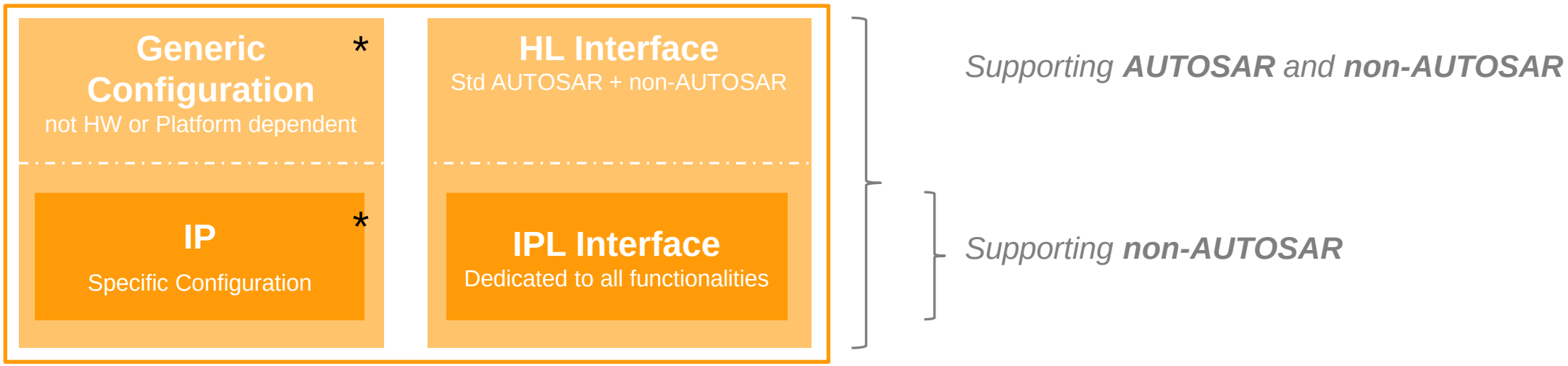
- Add **UART** support as complex driver for MCAL 4.4
- Add **Flexio_SENT** to support **SENT** communication
- **OsIf** can support FreeRTOS and AUTOSAR OS as well as bare-metal timer, but no semaphore, mutex and queue support
- **REG_PROT** is included in **BASE**
- Functional safety related driver, such as **MPU** and **XRDC** are included in Resource Manager as complex driver
- **FLS** also includes **QuadSPI** external Flash memory, **EEP** is a standard MCAL for D-Flash emulated EEPROM implementation.
- All RTD drivers have **timeout** and **multicore support** per AUTOSAR 4.4 standard and S32K3xx multi-core architecture required.

Real Time Driver	S32K3XX IP	Comments
DEM	-	Diagnostics Event Manager Reference code provided by NXP to be used in Non Autosar applications. To be replaced to Autosar Standard Implementation for Autosar applications
DET	-	Default Error Tracer Reference code provided by NXP to be used in Non Autosar applications. To be replaced to Autosar Standard Implementation for Autosar applications
ECUC	-	Ecu Configuration – add support for multicore Reference code provided by NXP to be used in Non Autosar applications. To be replaced to Autosar Standard Implementation for Autosar applications
ECUM	-	Ecu Manager Reference code provided by NXP to be used in Non Autosar applications. To be replaced to Autosar Standard Implementation for Autosar applications
RTE	-	Run Time Environment – implements exclusive areas Reference code provided by NXP to be used in Non Autosar applications To be replaced to Autosar Standard Implementation for Autosar applications
OsIf	-	OS Abstraction layer Integrates support for FreeRTOS and Autosar OS
Resource	-	Resource driver – collection for all derivatives features
BASE	-	Base driver Collection of header files
	REG_PROT	A subset of driver's files provided by NXP can be used in Non Autosar applications A subset of driver's files provided by NXP can be replaced to Autosar Standard Implementation for Autosar applications
MCU	MC_CGM	Microcontroler Unit Driver Provides services for basic microcontroller initialization, mode management and clock management
	FIRC	
	SIRC	
	PLLDIG	
	FXOSC	
	MC_RGM	
	MC_ME	
	SXOSC	
	RAM Controller	
	MC_PCU	
	PMC	
PLATFORM	MSCM	Platform – Complex Device Driver Integrates functionalities specific to platform and interrupt management
	NVIC	
	INTM	
	MCM	
RM	MPU	Resource Manager – Complex Device Driver
	XRDC	
	SEMA42	
	PRAMC	
	Pflash	
	VIRT_Wrapper	
	XBIC - MCR reg	
	XBAR	
MCL	eDMA	Microcontroller Library – Complex Device Driver Offers DMA and Cache functionalities
	LCU	
	DMAMUX	
	TRGMUX	
	Cache_M7	
PORT	SIUL2	PORT Driver Provides the services for initializing the whole structure of PORT driver
DIO		Digital Input Output Driver Provides services for accessing the microcontroller's hardware pins
EEP		EEPROM Driver Provides services for reading, writing, erasing to/from an EEPRM
FLS	QuadSPI	Flash Driver
	C40	
	Pflash	
ICU	eMIOS	Input Capture Unit Driver
	SIUL2	
	CMP	
	WKPU	
OCU	eMIOS	Output Capture Unit Driver
GPT	eMIOS	General Purpose Timer Driver
	PIT	
	RTC	
	STM	
PWM	eMIOS	Pulse Width Modulation Driver
	FlexIO_PWM	
QD	eMIOS	Quadrature Complex Device Driver
I2C	LPI2C	I2C Complex Device Driver
	FlexIO_I2C	
ETH	ENET	Ethernet Driver
UART	LPUART	UART Complex Device Driver
	FlexIO_UART	
LIN	LPUART	Lin Driver
SPI	LPSPI	Serial Parallel Interface Driver
	FlexIO_SPI	
CAN	FlexCan	Can Driver
ADC	ADC	Analog Digital Comparator Driver
	BCTU	
Crypto	HSE_M	Crypto Driver
	MU	
WDG	SWT	Watchdog Driver
SENT	FlexIO_SENT	SENT driver
CRC	CRCU	CRC Complex Device Driver

Table 1 Mapping between Drivers and S32K3XX Drivers

CUSTOMER FACING:
ONE PACKAGE FOR AUTOSAR & NON-AUTOSAR ENVIRONMENTS

RTD offers both abstracted/standardized interfaces and HW specific interfaces (exported by the IPL Interface)
These two interface types are exclusive, cannot be used at the same time



IPL (IP Layer):

- Peripheral specific layer implementing support for all IP features
- Constant for the same IP across platforms.
- Dedicated to export all hardware functionalities
- IP layer to come with standalone ISO 26262 compliance

HL (High Layer):

- Implements the standard APIs described into the AUTOSAR specifications
- Implements the APIs extensions based on:
  - Specific customer requirements
  - IP features exposed from layer below

*** as part of S32 Configuration Tool (S32CT) and EB tresos**

Depending on the context wanted to be used by the upper layer, a specific interface must be used:

AUTOSAR context:

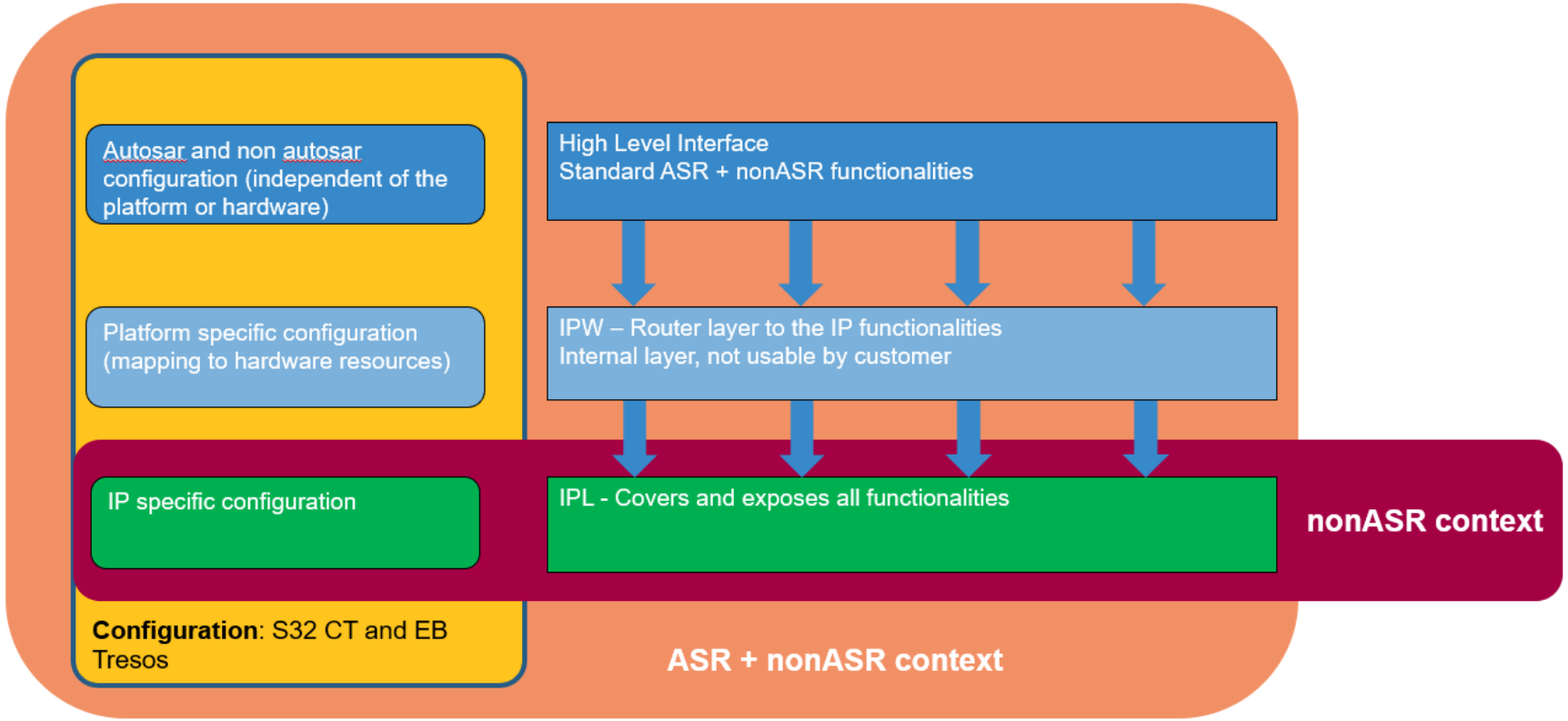
- High Level Interface usage

non-AUTOSAR context:

- High Level Interface usage

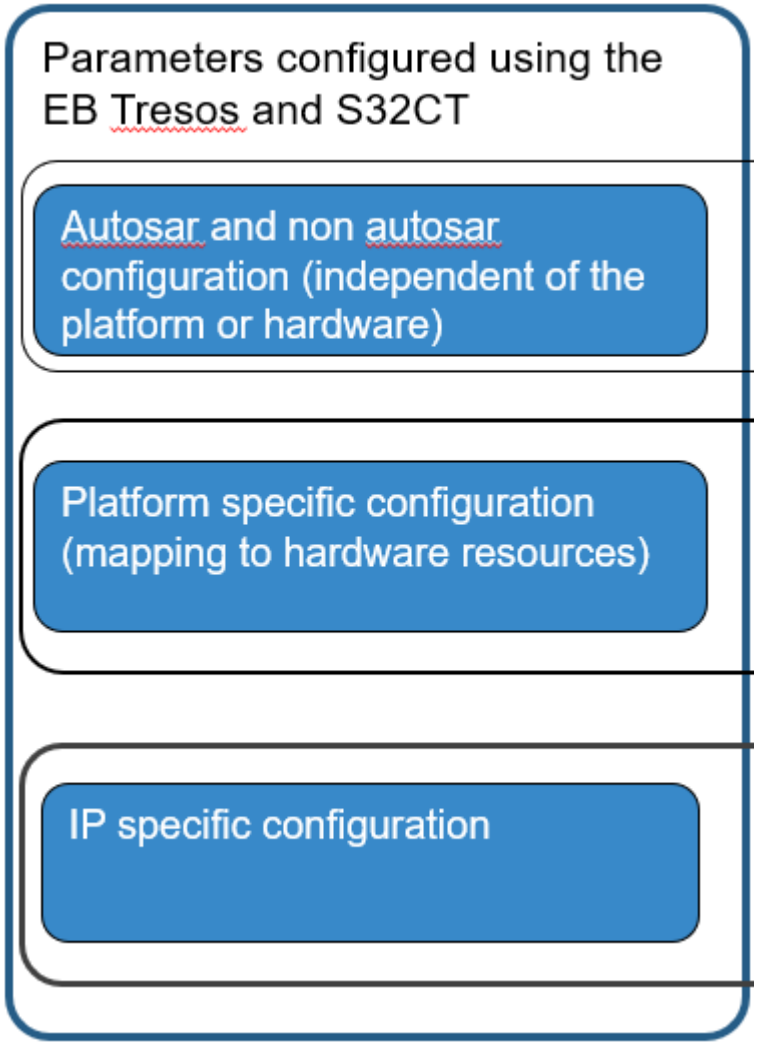
or

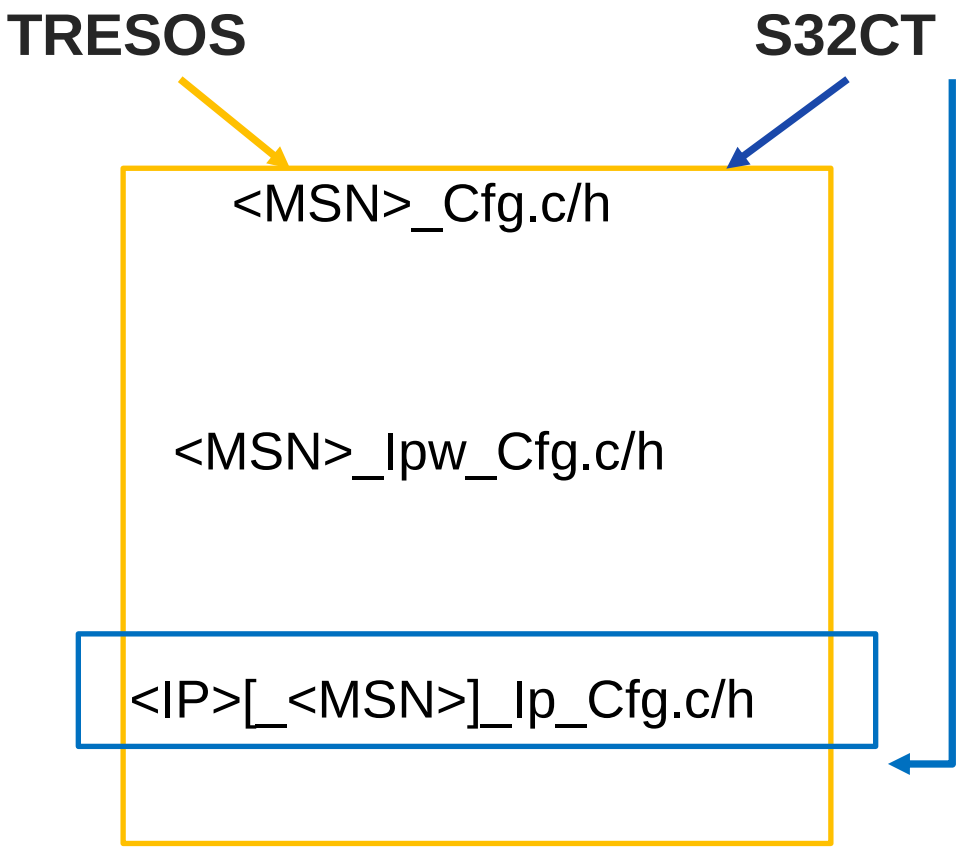
- IP Layer (ex-SDK)



AUTOSAR = ASR
non-AUTOSAR = nonASR

REAL-TIME DRIVERS (RTD) – CONFIGURATION SCHEME





The configuration output (c/h files) shall be identical with both tools*

If the HL is used in the application, the initialization must be done with the config from <MSN>_Cfg.c/h

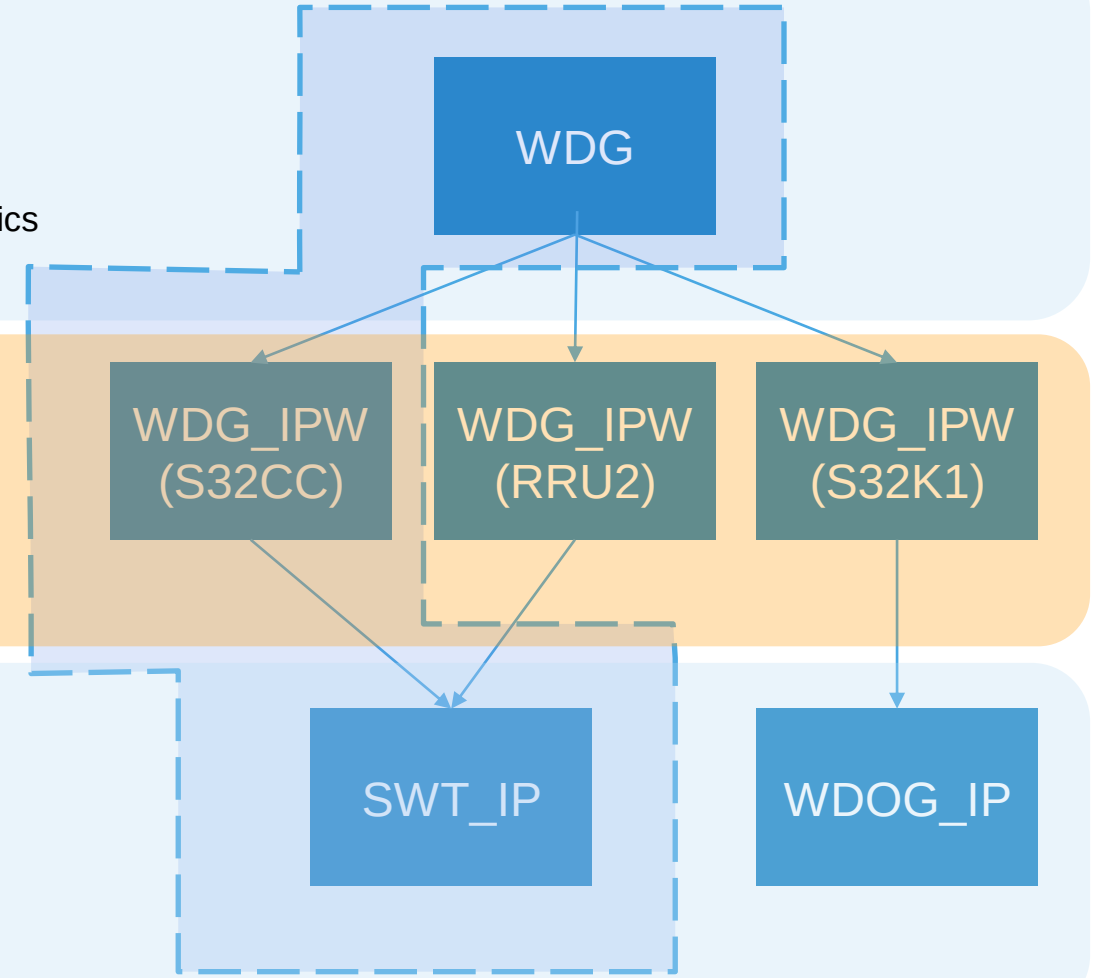
If the IP layer is used in the application, the initialization must be done with the config from <IP>[_<MSN>]_Ip_Cfg.c/h

REAL-TIME DRIVERS (RTD) – LAYERED ARCHITECTURE

RTD High-Level Interface (HLI) Layer:
- Generic cross-NPIs (up to 100% reuse)
- Implements APIs called by the RTE/App Level:
- Implements standard AUTOSAR APIs
- Implements AUTOSAR API extensions & non-AUTOSAR APIs for NXP HW specifics

RTD IP Wrapper Layer:
- This is the code that changes between NPIs.
- Maps the generic HLI APIs to the IP specific ones (LLI APIs).

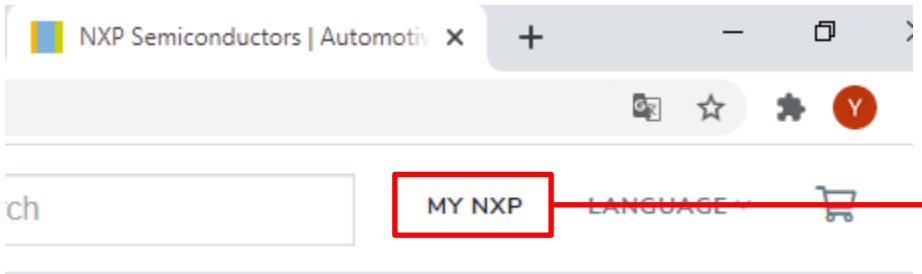
RTD Low-Level Interface (LLI) Layer:
- IP Specific Software Layer that implements IP (µC) specific functionalities
- Constant (full reuse) for the same IP across NPIs and the same AUTOSAR versions (up to 100% reuse)



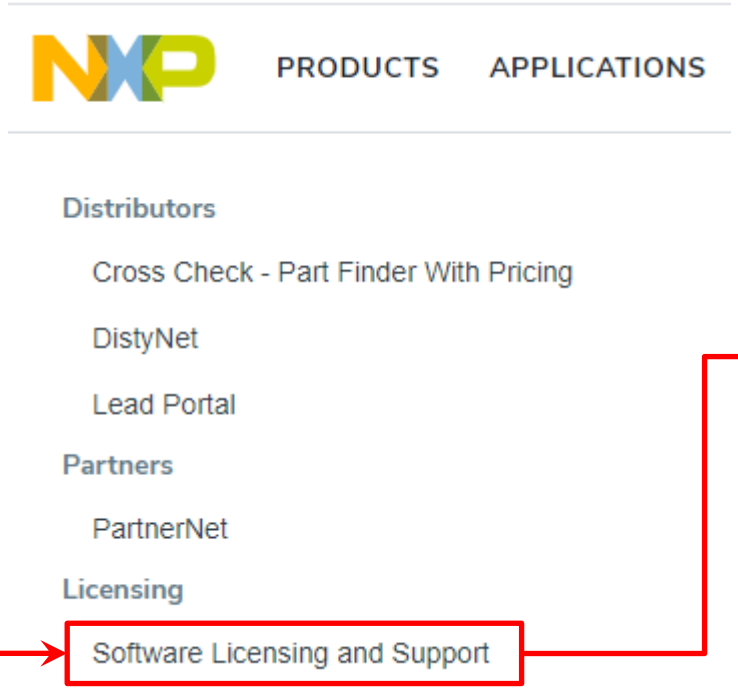
S32DS and RTD for S32K3
INSTALL GUIDE

SECURE CONNECTIONS
FOR A SMARTER WORLD

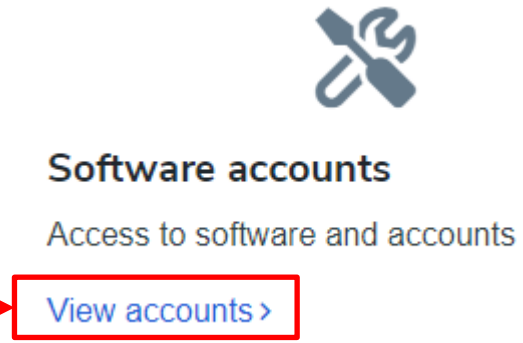
1) LOG IN WITH YOUR CREDENTIALS



Login at NXP.com with your credentials



Select
Software Licensing and Support

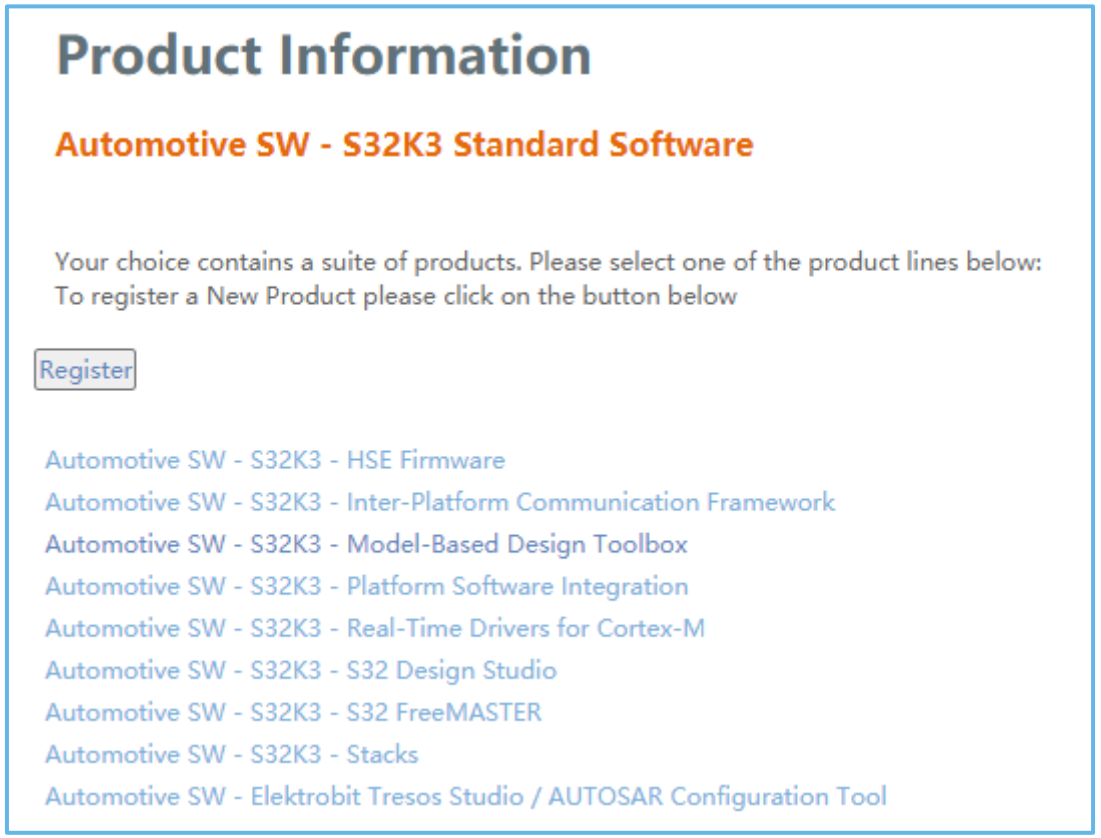


View your account

2) DOWNLOAD: S32K3 STANDARD SOFTWARE PACKAGE

Login your account on NXP website, and download the **S32K3 Standard Software** from:

https://www.nxp.com/webapp/swlicensing/sso/downloadSoftware.sp?catid=SW32K3-STDSW-D

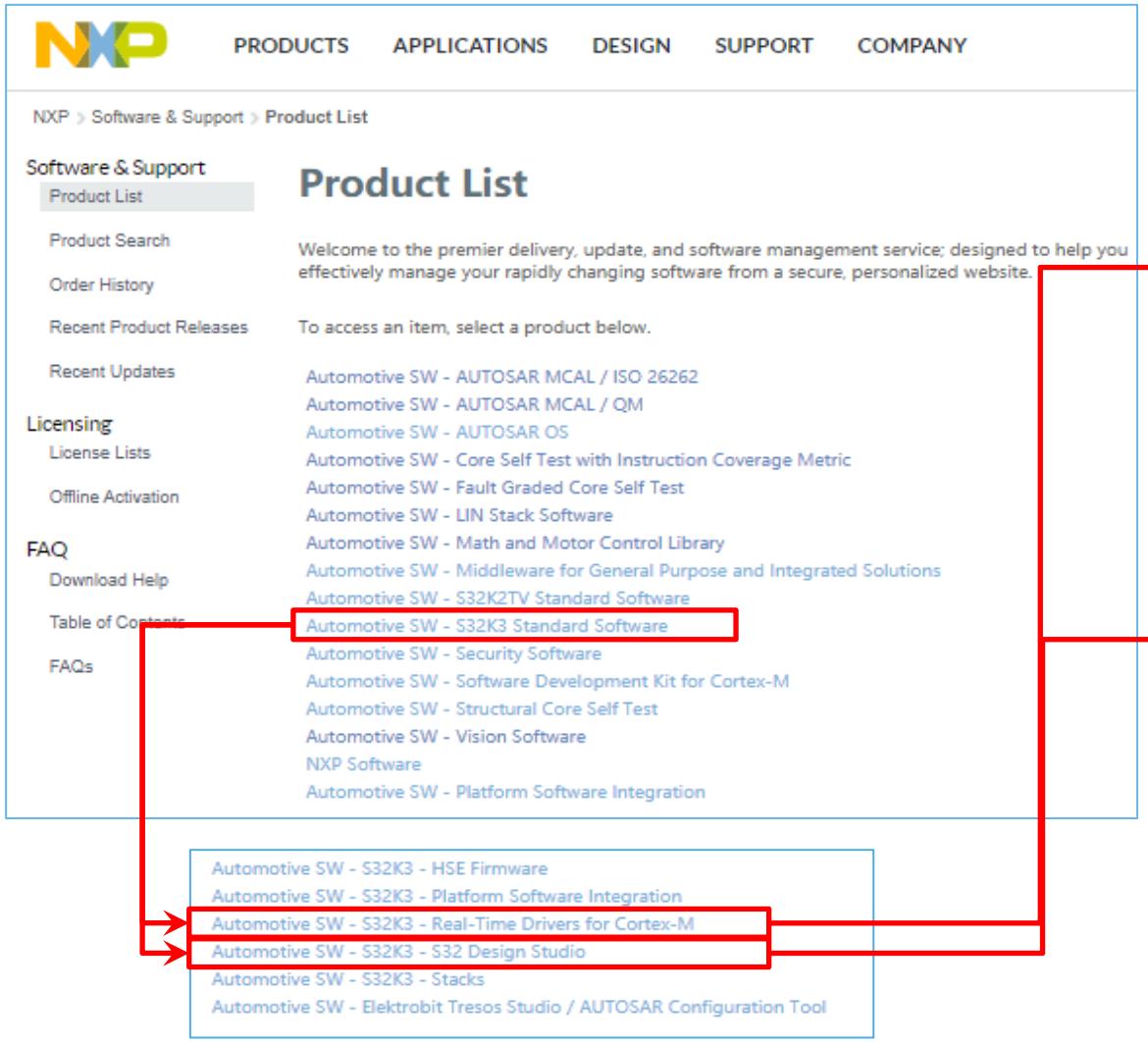


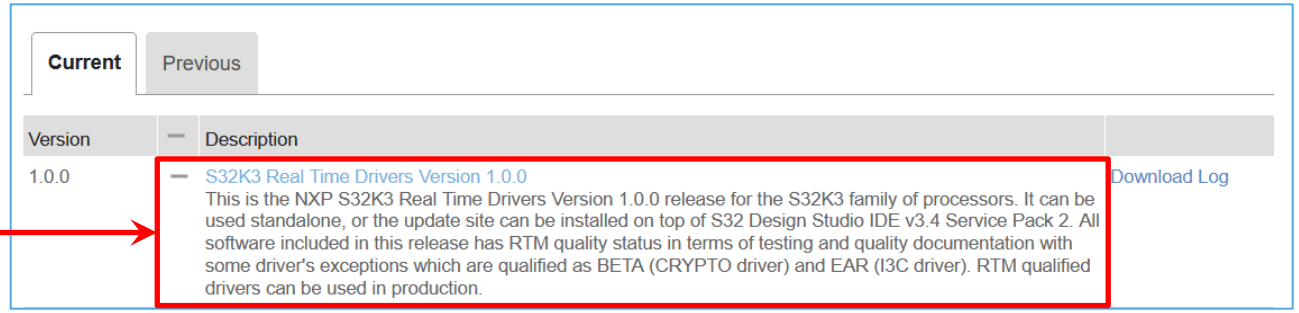
Make sure to login your account first.

Then click link to access the available ***Automotive SW – S32K3 Standard Software***

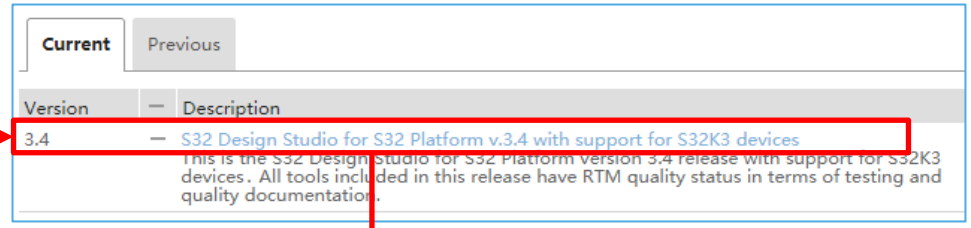
3) DOWNLOAD: S32DS (S32DS 3.4) AND K3 RTD

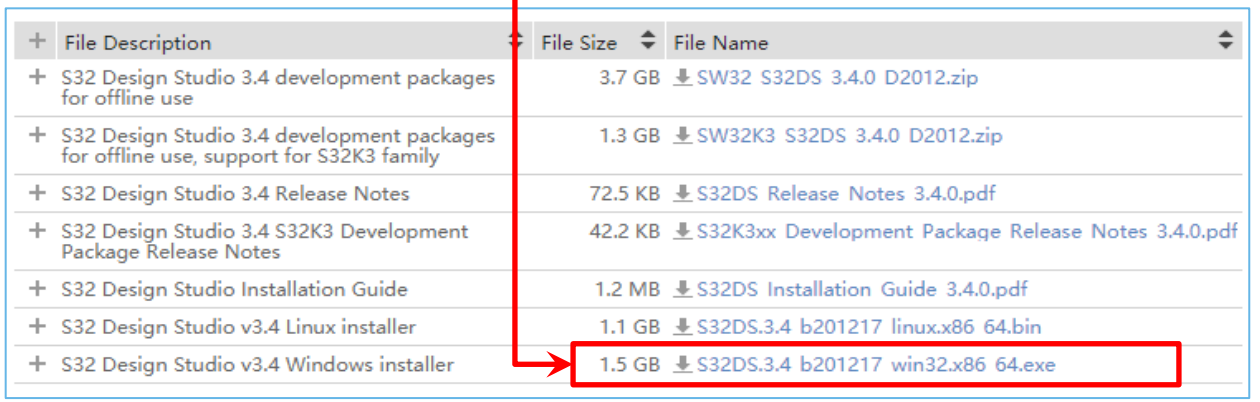
*Download latest version of **RTD***





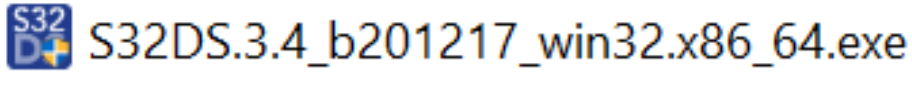
*Download **S32 Design Studio Installer***



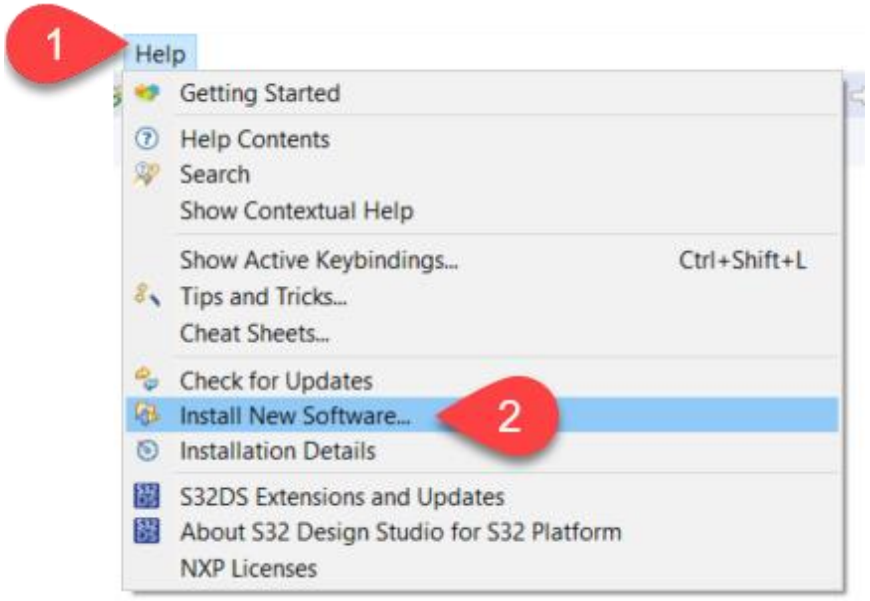


– "Automotive SW - S32K3 - S32 Design Studio"

- Step 1: Run "S32 Design Studio v3.4 Windows installer"
  - **"S32DS.3.4_b201217_win32.x86_64.exe"**
  - License Keys => Activation Code.

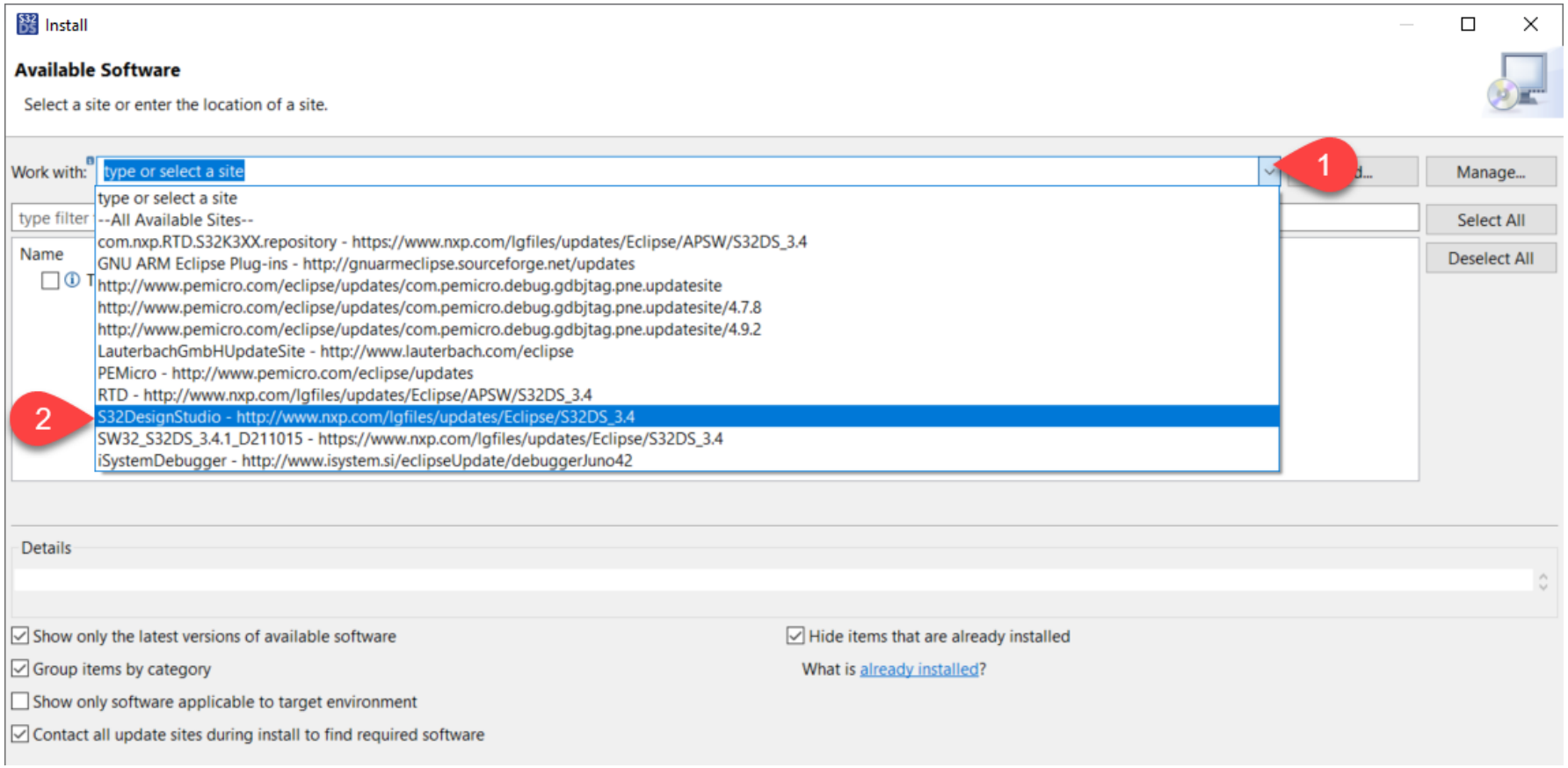


- Step 2: : Install "S32 Design Studio 3.4 development packages for offline use, support for S32K3 family" in S32DS3.4:
  - After the installation was completed, open the S32DS3.4 and go to Help => Install New Software



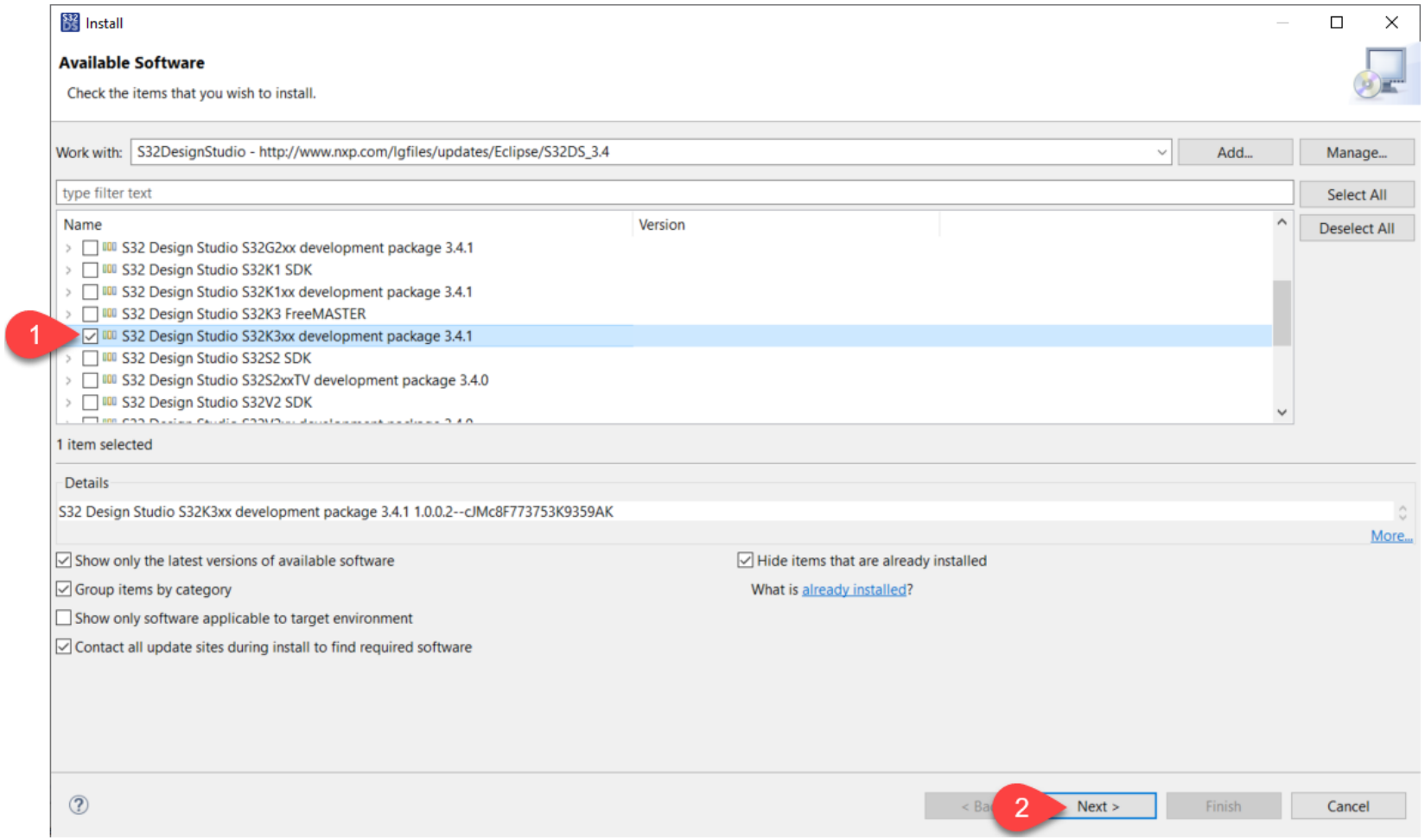
4) INSTALL: S32DS 3.4 AND S32K3 SUPPORT PACKAGE

- Step 3: Look for the available software
  - Drop down on the menu
  - Select “S32DesignStudio - http://www.nxp.com/lgfiles/updates/Eclipse/S32DS_3.4”



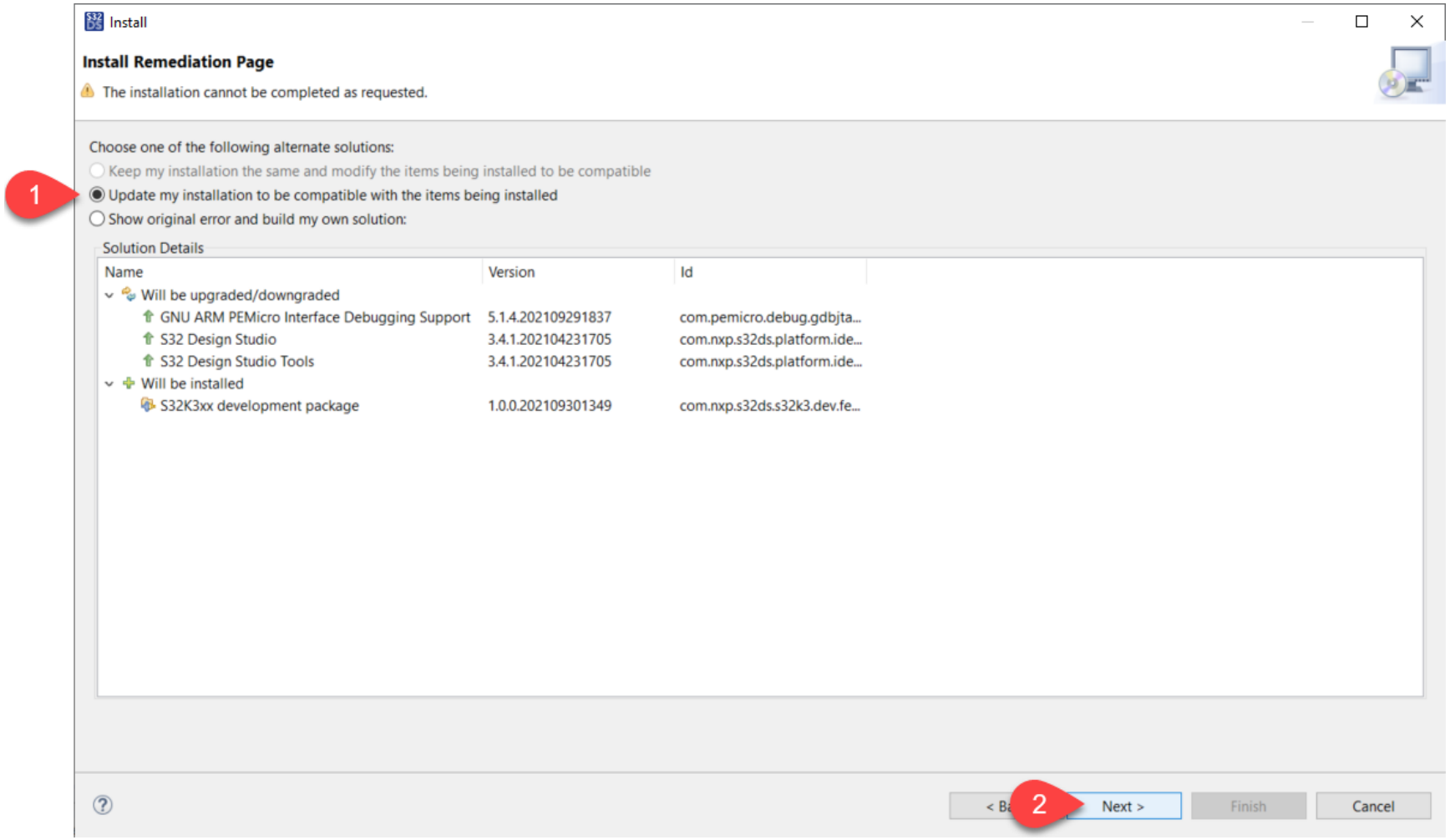
4) INSTALL: S32DS 3.4 AND S32K3 SUPPORT PACKAGE

- Step 4: Choose the S32DS S32K3xx development package
  - Select the S32DS S32K3 development package 3.4.1 and click Next



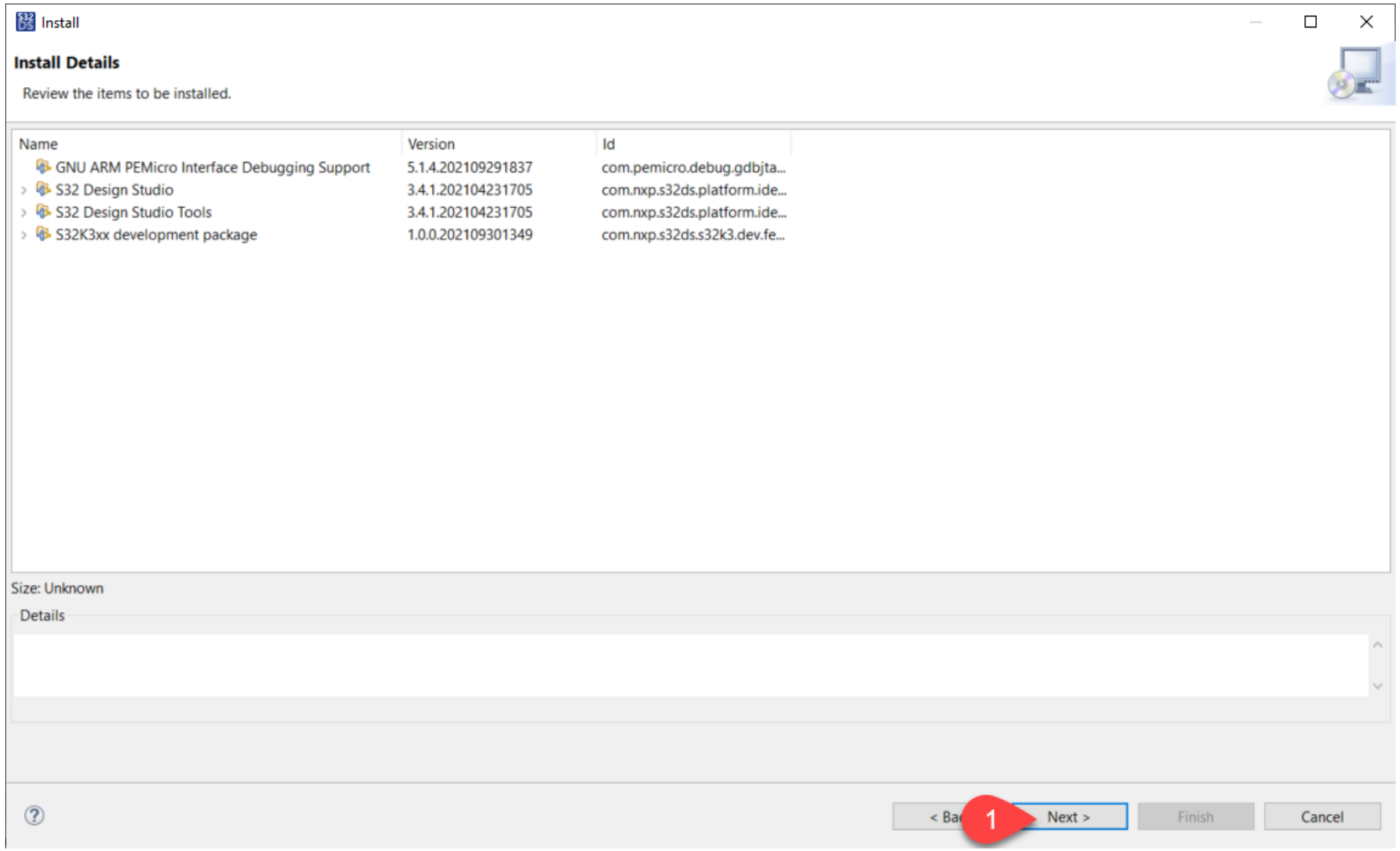
4) INSTALL: S32DS 3.4 AND S32K3 SUPPORT PACKAGE

- Step 5: Install the S32DS S32K3xx development package
  - Select the option “Update my installation to be compatible with the items being installed” and click Next



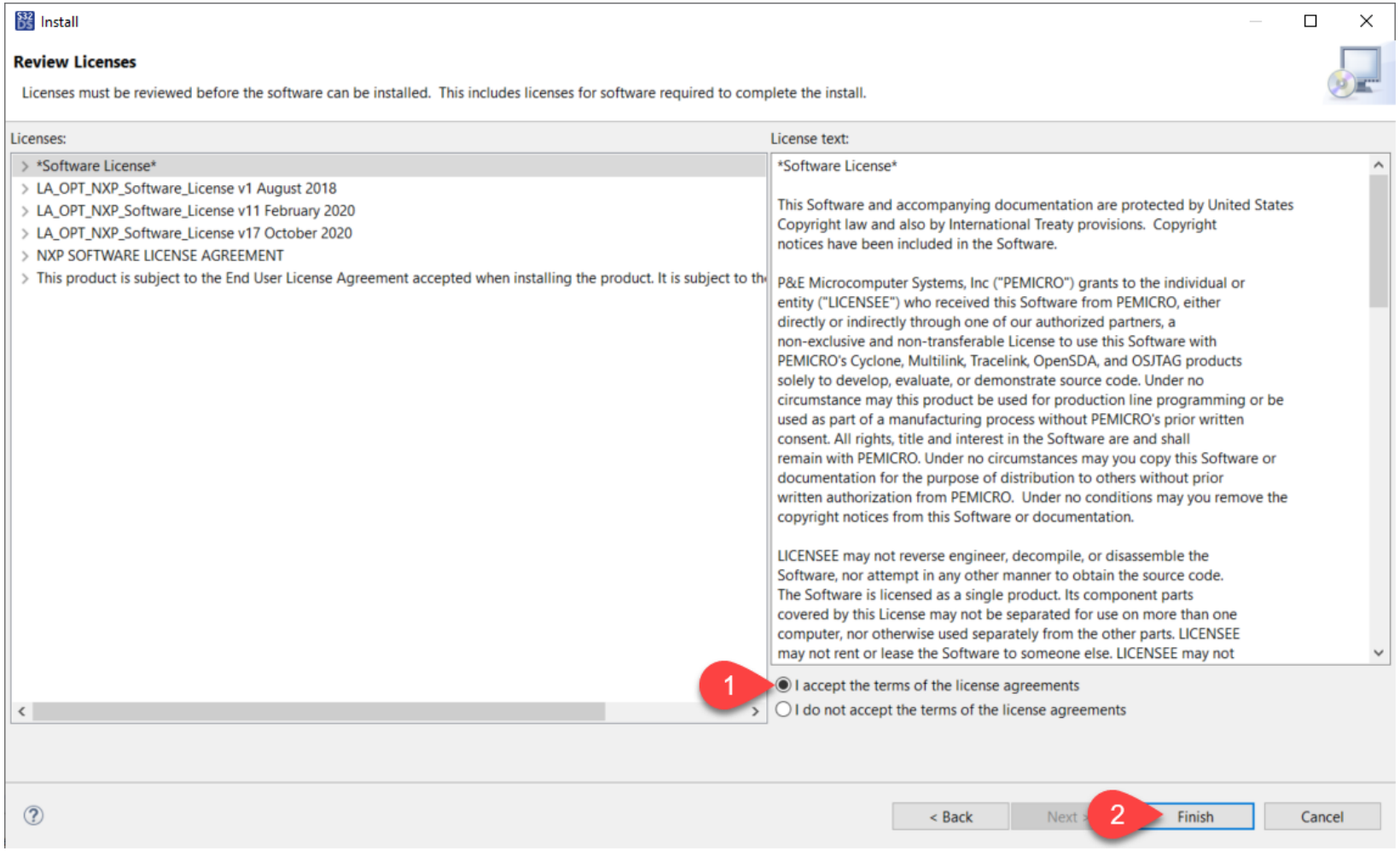
4) INSTALL: S32DS 3.4 AND S32K3 SUPPORT PACKAGE

- Step 5: Install the S32DS S32K3xx development package
  - Now you can review the installation details. Click Next



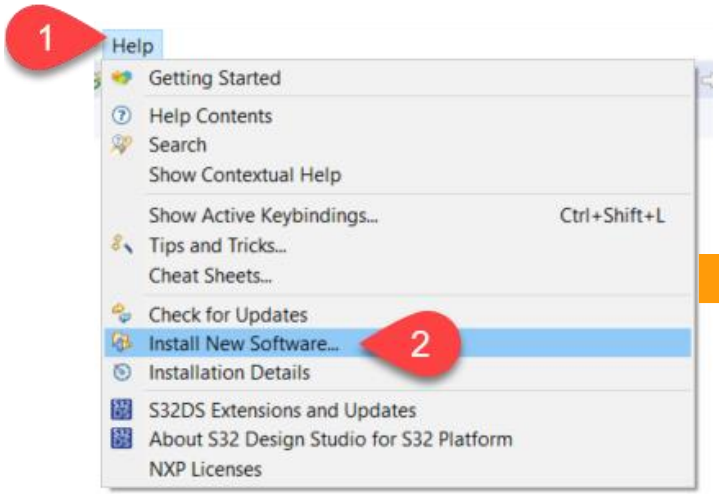
4) INSTALL: S32DS 3.4 AND S32K3 SUPPORT PACKAGE

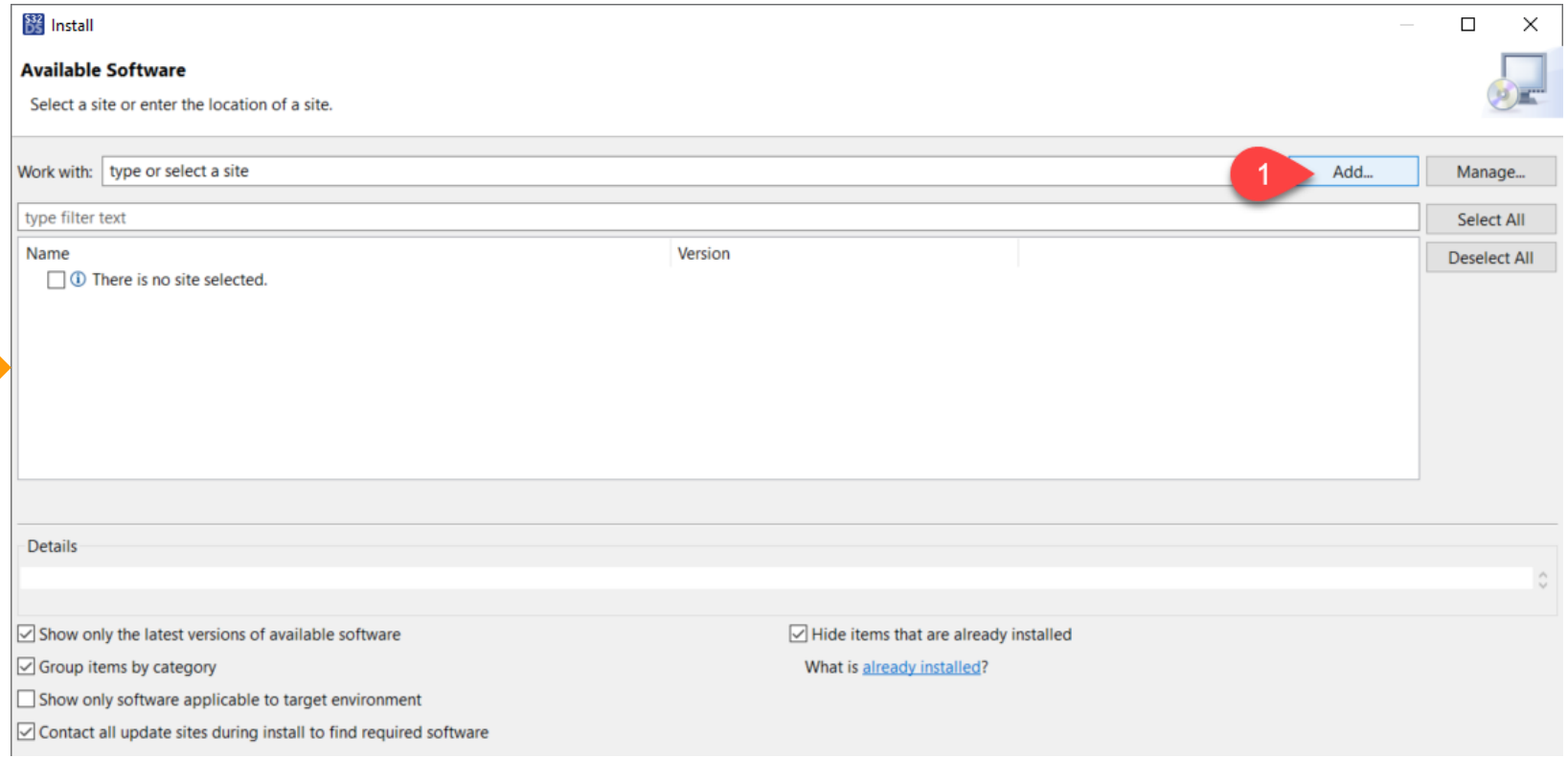
- Step 5: Install the S32DS S32K3xx development package
  - Accept the license and click Finish



5) INSTALL: RTD FOR S32K3 IN S32DS 3.4

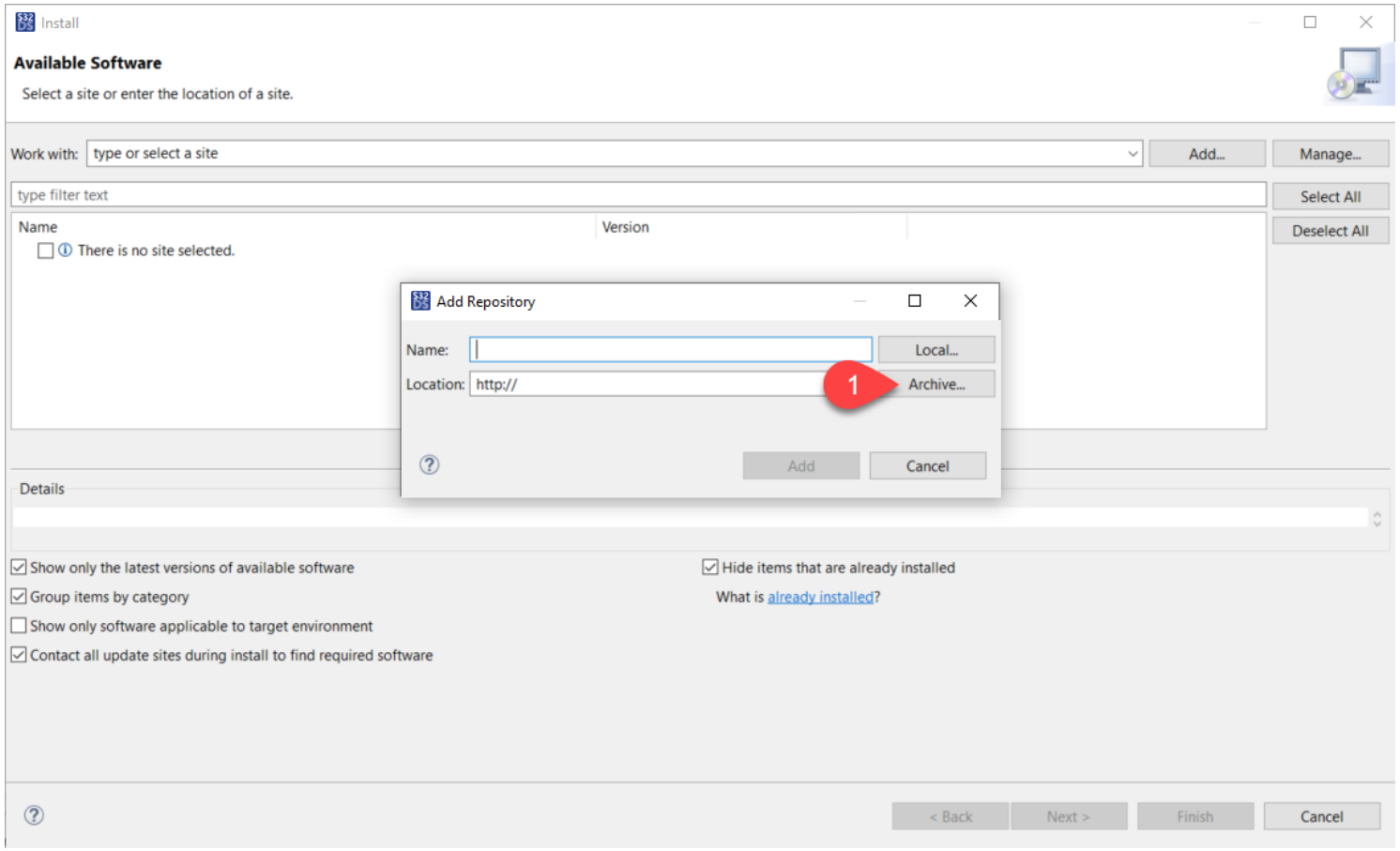
- Step 1: Install “S32K3 Real Time Drivers (RTD)” in S32DS3.4:
  - Open the S32DS3.4 and go to Help => Install New Software
  - Click on the “Add” button.





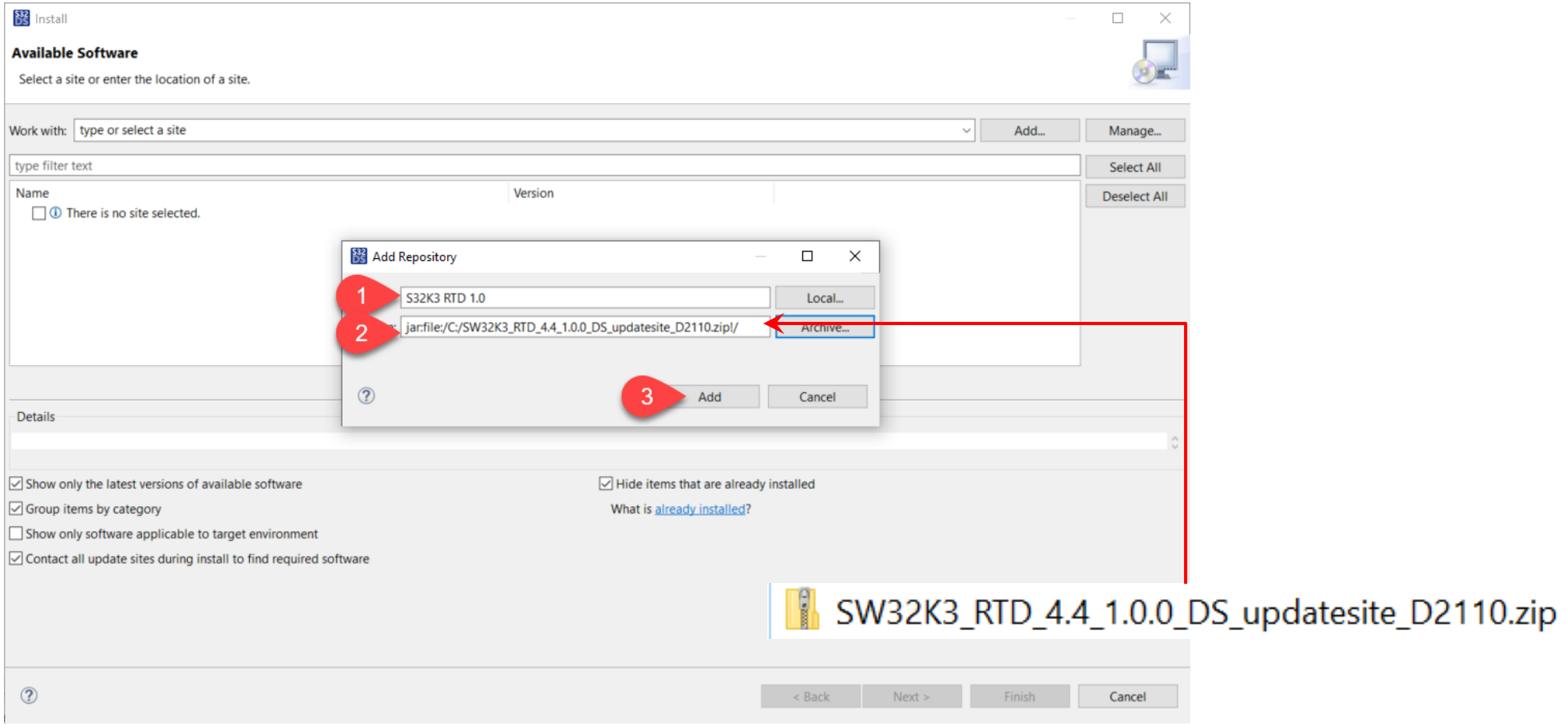
5) INSTALL: RTD FOR S32K3 IN S32DS 3.4

- Step 2: Look for the available software
  - Click on the “Archive” button and look for the **SW32K3_RTD_4.4_1.0.0_DS_updatesite_D2110** file inside your files. Remember that we downloaded the S32K3 RTD 1.0 package from the NXP official website (Slide 20)



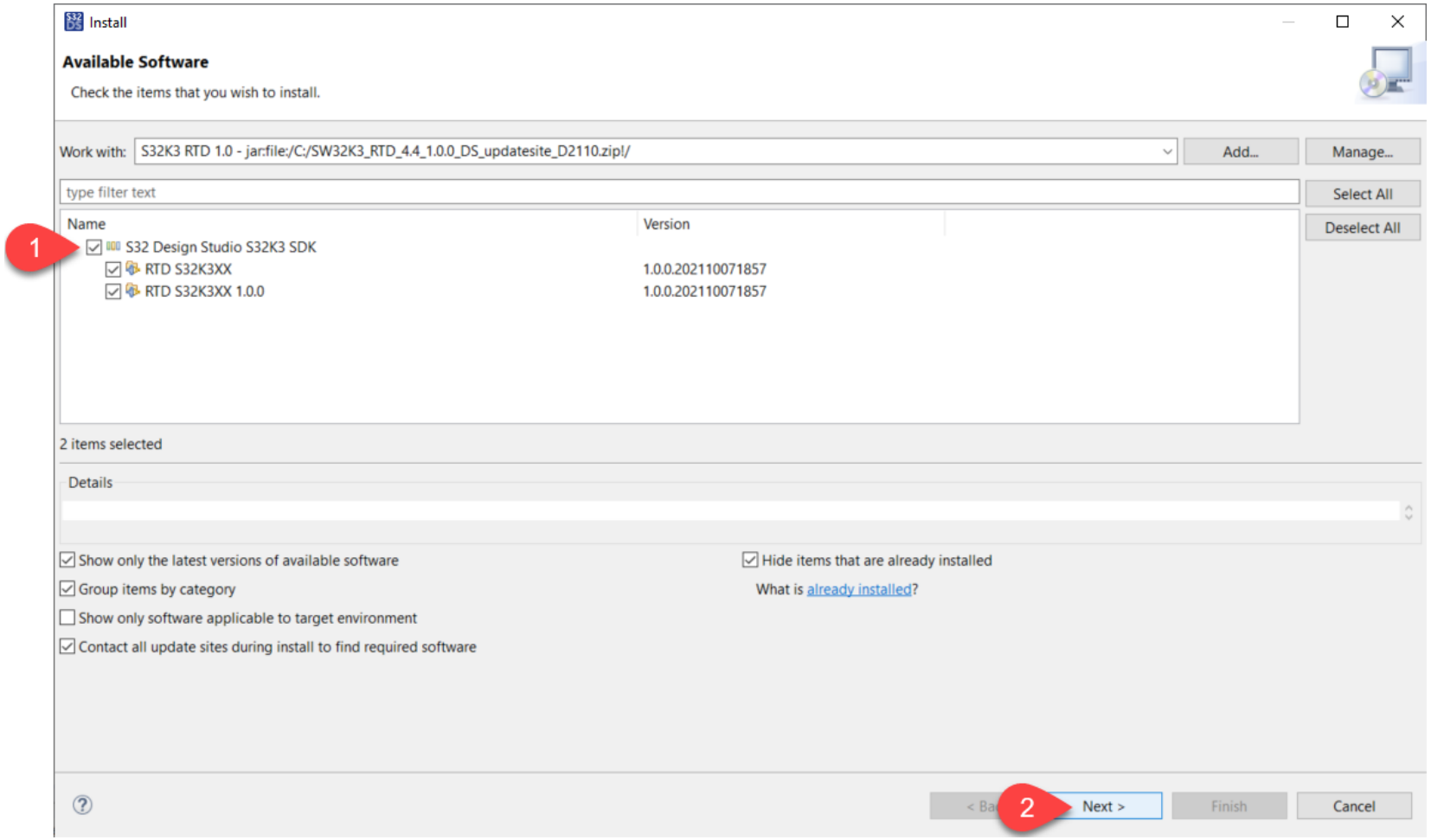
5) INSTALL: RTD FOR S32K3 IN S32DS 3.4

- Step 2: Look for the available software
  - Select a name to identify this new package. For example, “S32K3 RTD 1.0”
  - Remember to select the **SW32K3_RTD_4.4_1.0.0_DS_updatesite_D2110** file and click “Add”



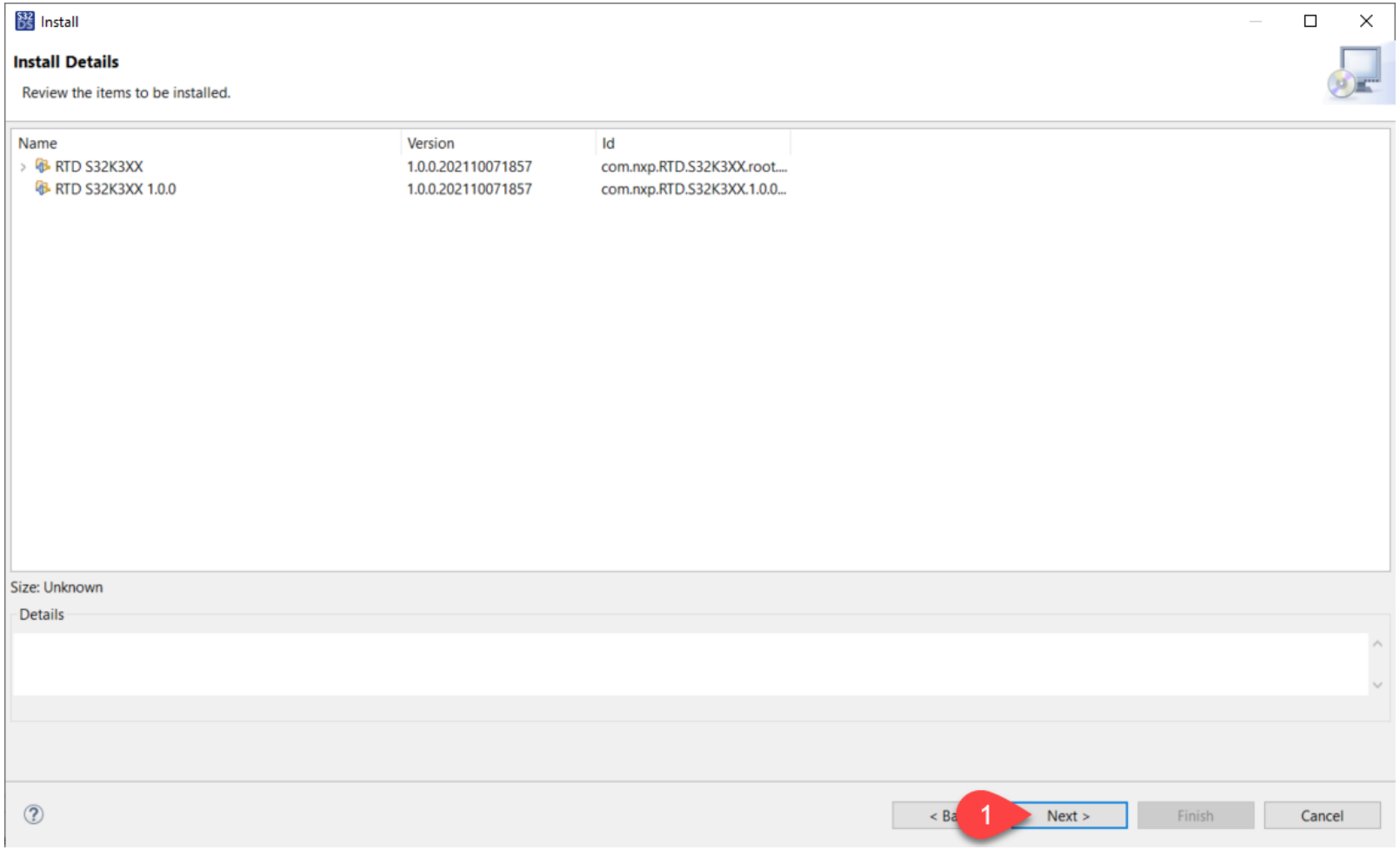
5) INSTALL: RTD FOR S32K3 IN S32DS 3.4

- Step 3: Choose the S32DS S32K3xx RTD package
  - Select the S32 Design Studio S32K3 RTD package and click Next



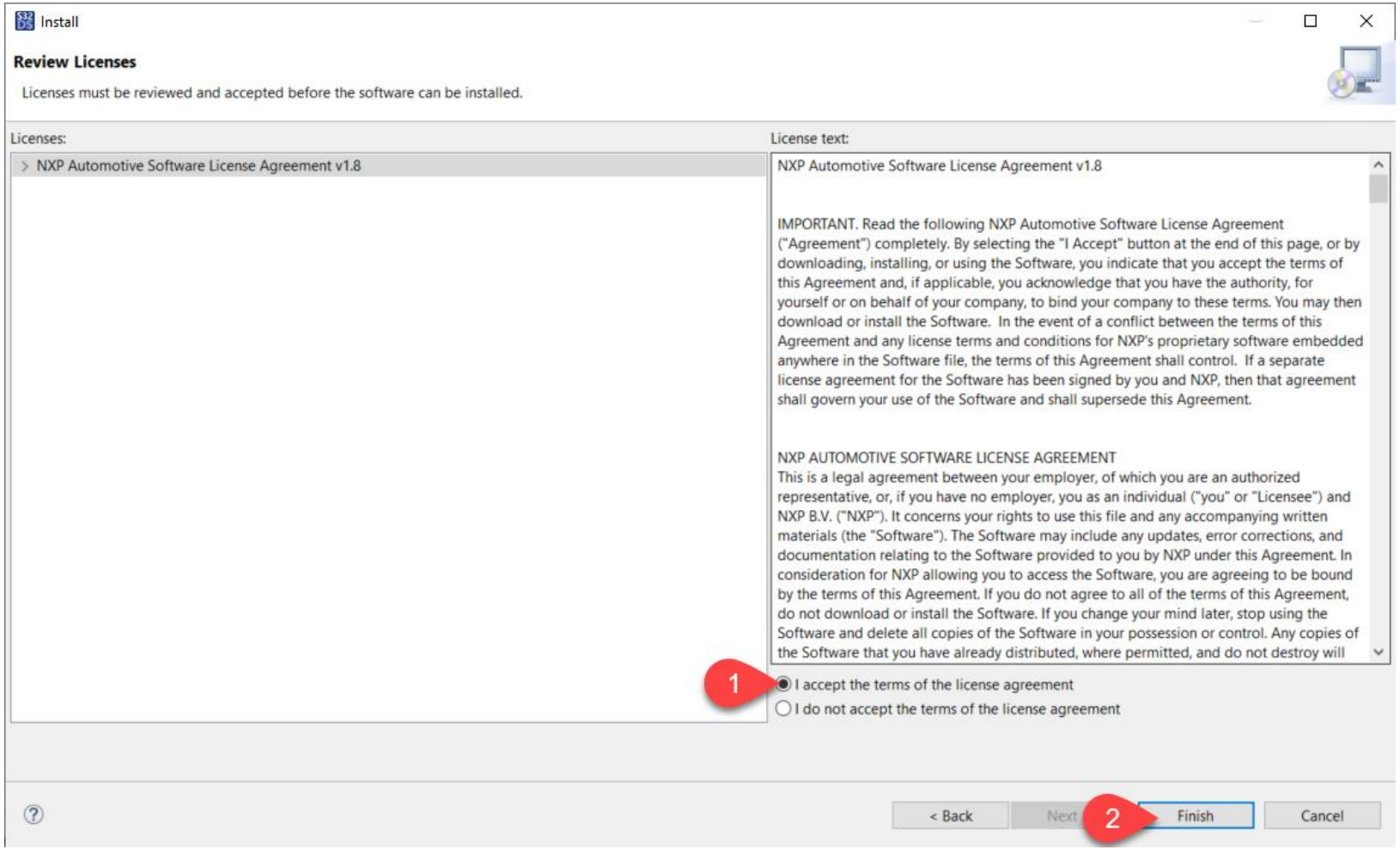
5) INSTALL: RTD FOR S32K3 IN S32DS 3.4

- Step 4: Install the S32DS S32K3xx RTD package
  - Now you can review the installation details. Click Next



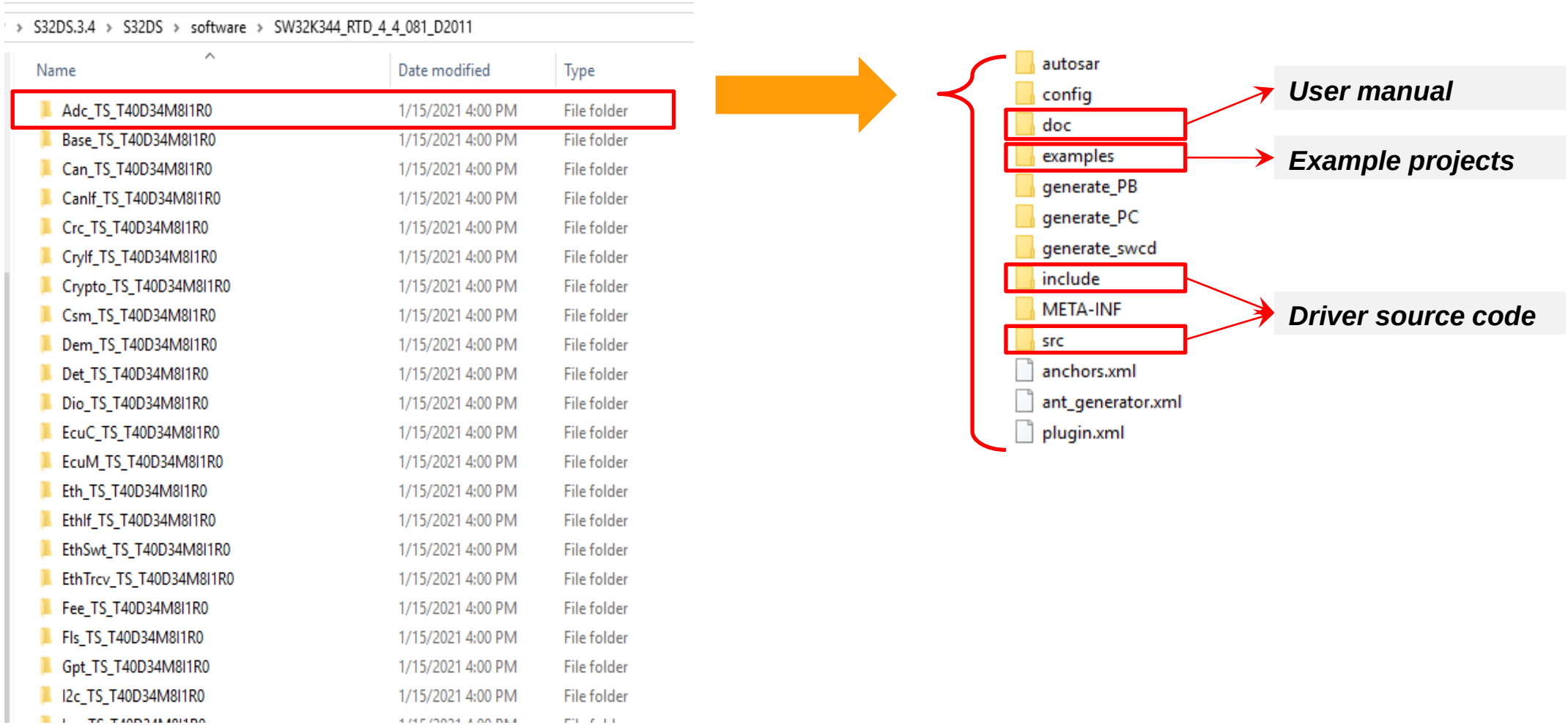
5) INSTALL: RTD FOR S32K3 IN S32DS 3.4

- Step 4: Install the S32DS S32K3xx RTD package
  - Accept the license and click Finish



6) CONFIRM: RTD FILES

After installation, the **RTD for S32K3** files can be found in the following path:
C:\NXP\S32DS.3.4\S32DS\software\PlatformSDK_S32K3_2021_10



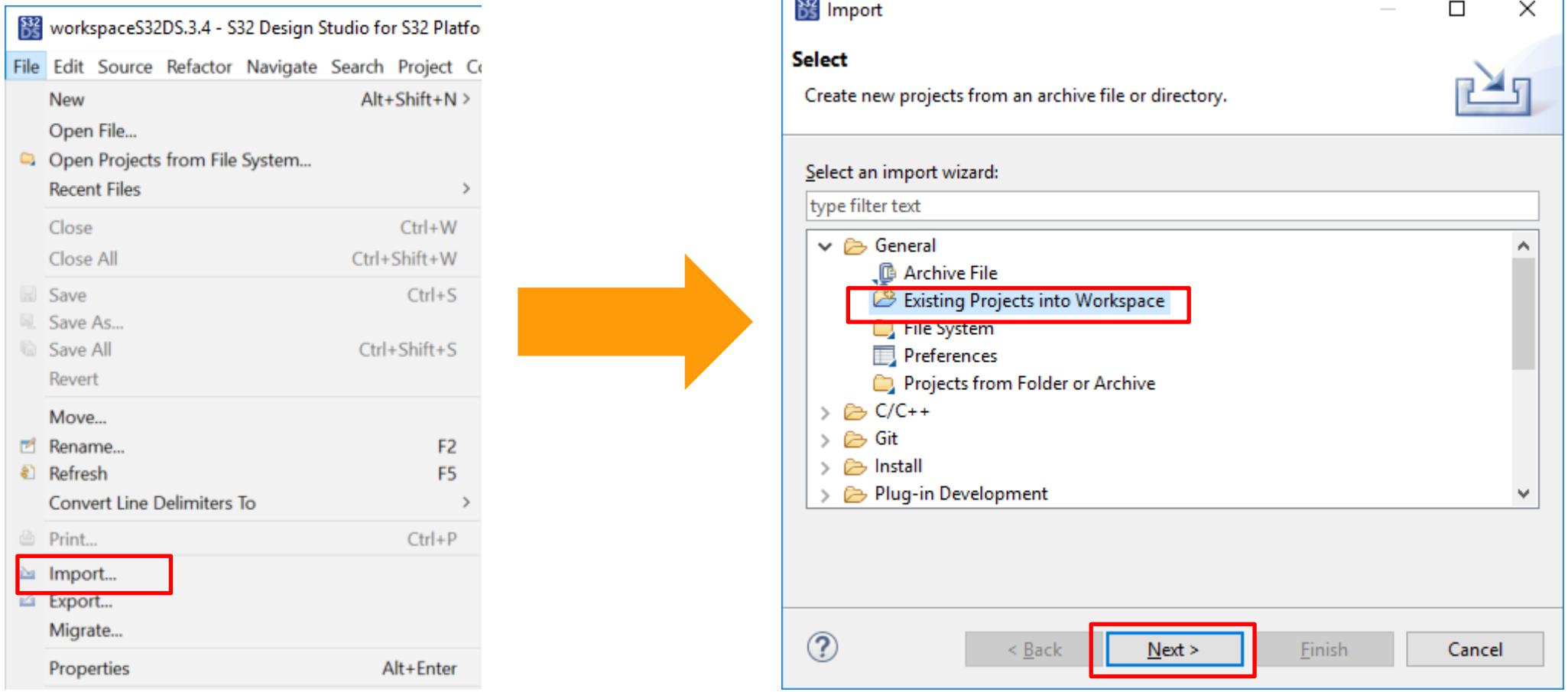
REAL-TIME DRIVERS (RTD) for S32K3

EXAMPLE PROJECTS

SECURE CONNECTIONS
FOR A SMARTER WORLD

IMPORT EXISTING EXAMPLE PROJECTS

- File → Import → General → Existing Projects into Workspace → Next



IMPORT EXISTING EXAMPLE PROJECTS

Example projects path:

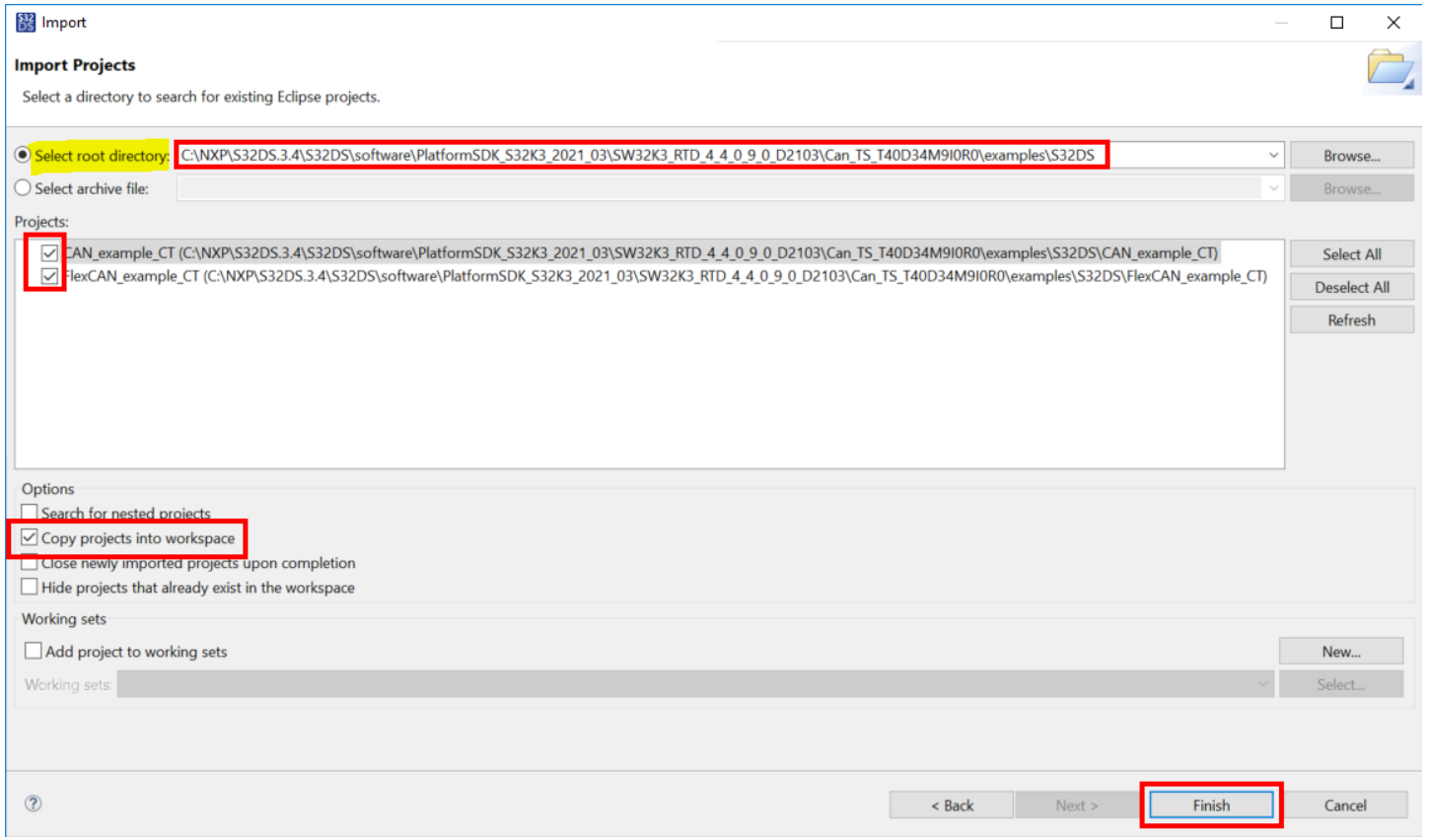
C:\NXP\S32DS.3.4\S32DS\software\PlatformSDK_S32K3_2021_10\SW32K3_RTD_4_4_1_0_0_D2110\Can_TS_T40D34M10I0R0\examples\S32DS

"examples\EBT" is for **EB tresos** example project and "examples\S32DS" is for **S32DS** example

Recommend to select "Copy projects into workspace" to save original example project for reference

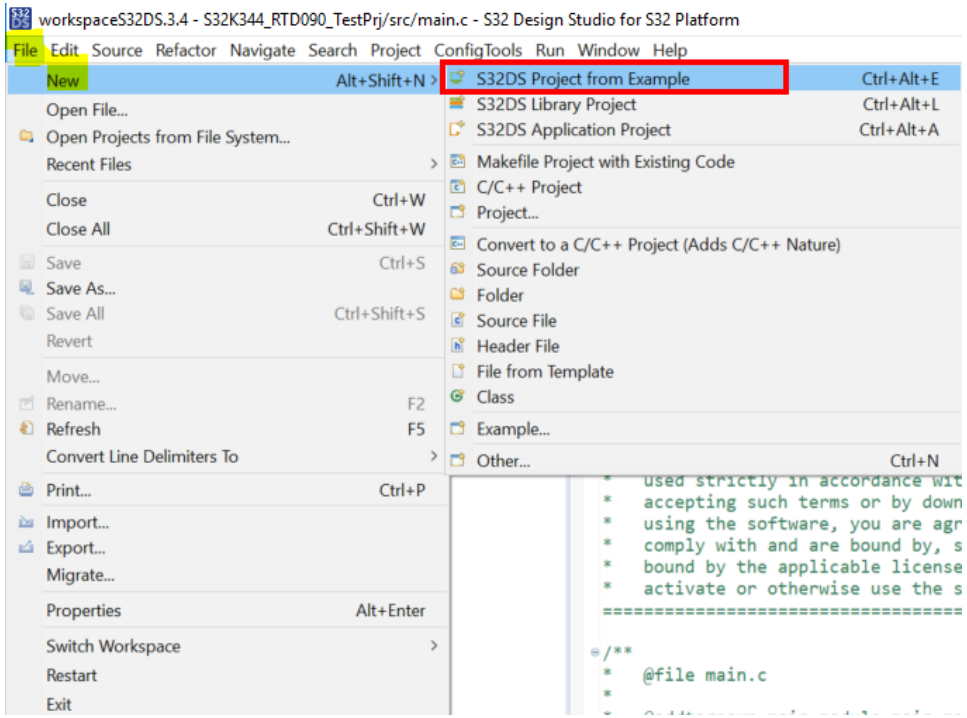
Available SDK examples:

- *adc_ip_example*
- *Can*
- *Dio_example_DS*
- *Eth_Example_DS_001*
- *FLS_IP_C40_Example_001*
- *FLS_IP_QSPI_Example_001*
- *Gpt_example_DS*
- *I2c_CodeDrop_example_DS*
- *Icu_example_DS*
- *dma_ip_transfer*
- *Power_Ip_Example_CT*
- *Clock_Ip_Example_CT*
- *Port_example_DS*
- *Ip_Lpspi_example_DS*
- *swt_ip_interrupt*

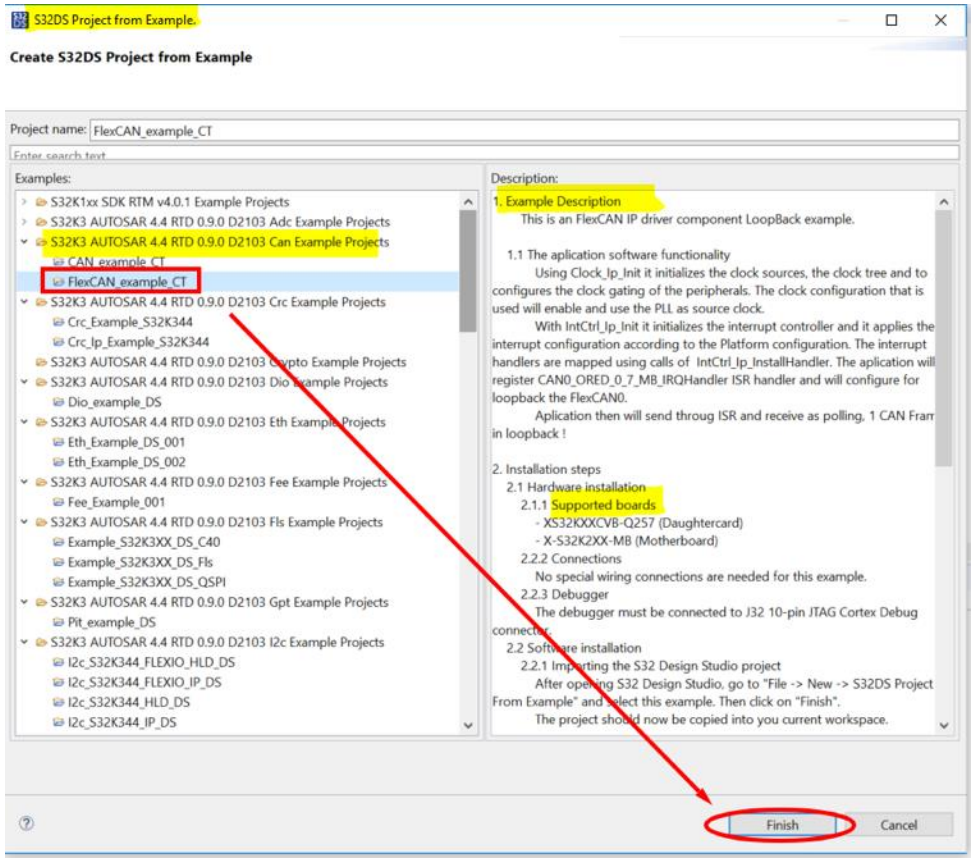


CREATE AN S32DS PROJECT FROM EXAMPLE

File → New → S32DS Project from Example

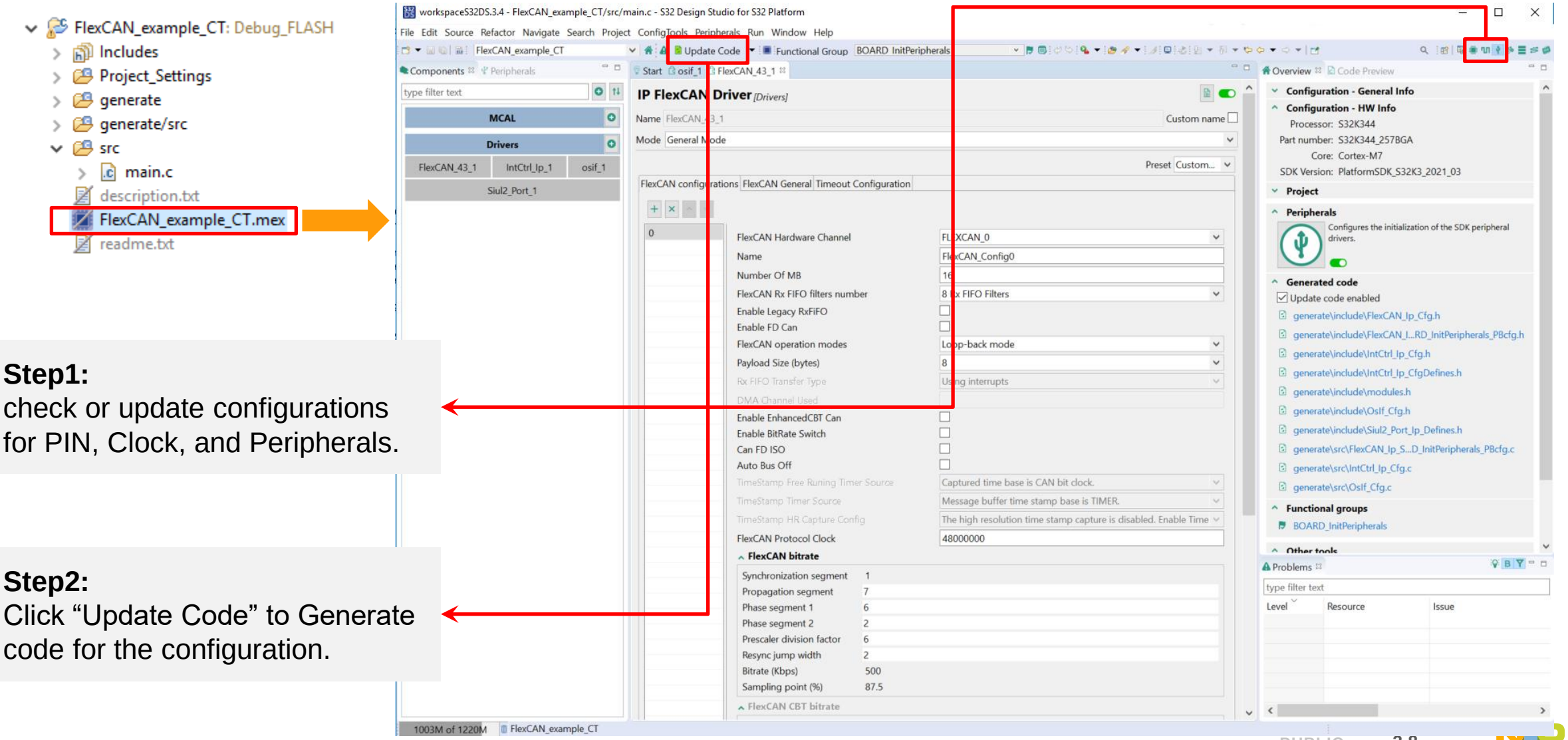


Select the RTD version and example project



GENERATE CODE FOR EXAMPLE PROJECT

Double click the “mex” file to open SDK configuration tool.

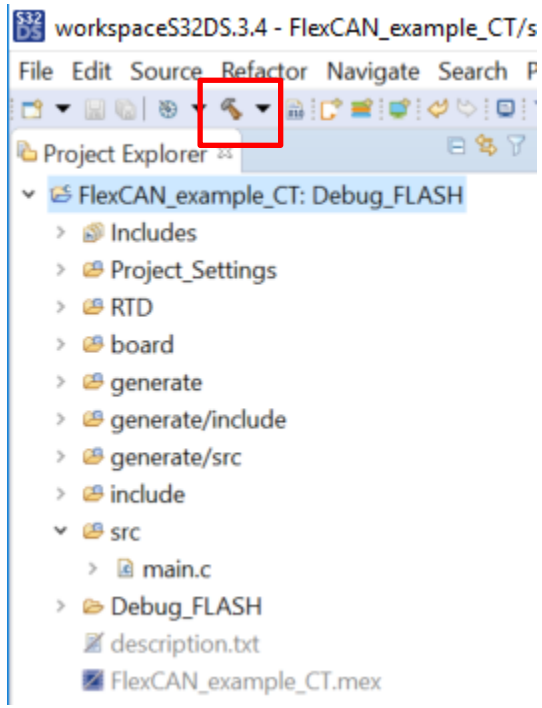


Step1:
check or update configurations for PIN, Clock, and Peripherals.

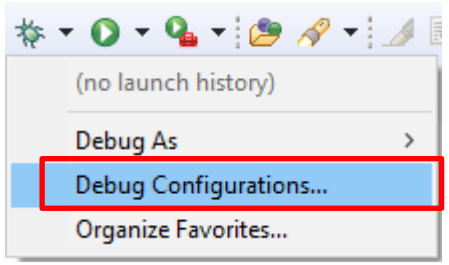
Step2:
Click “Update Code” to Generate code for the configuration.

BUILD AND DEBUG THE EXAMPLE PROJECT

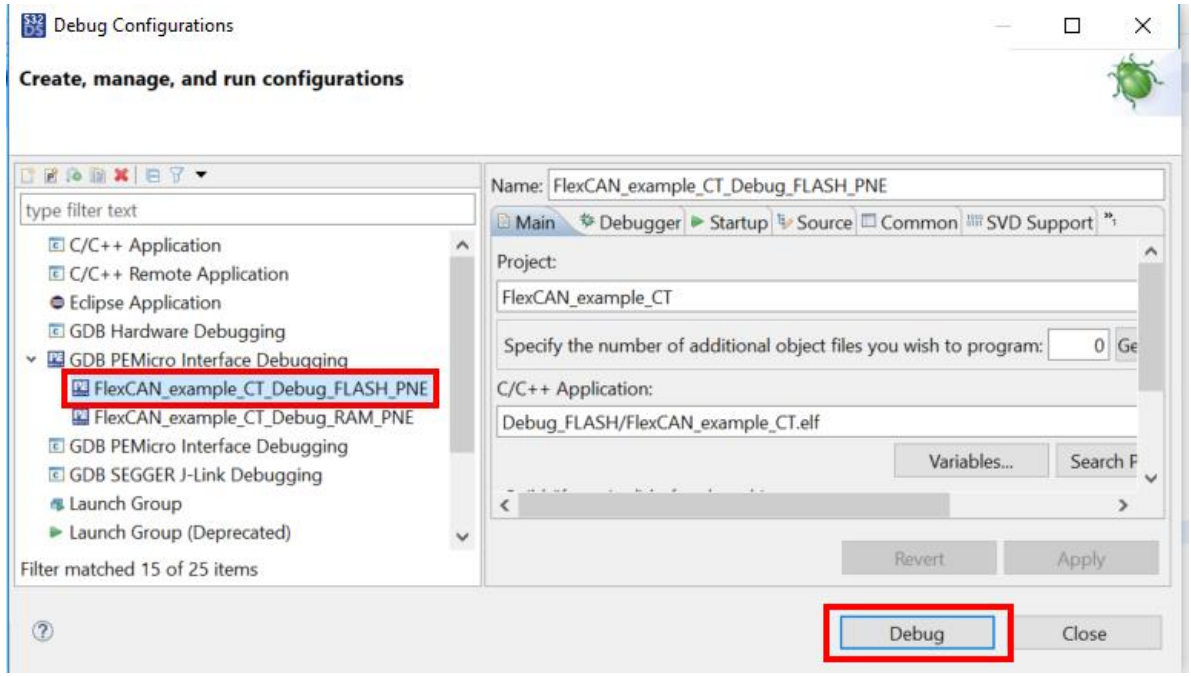
Build the project



Configure Debugging



Download and Debug the example project



REAL-TIME DRIVERS (RTD) for S32K3

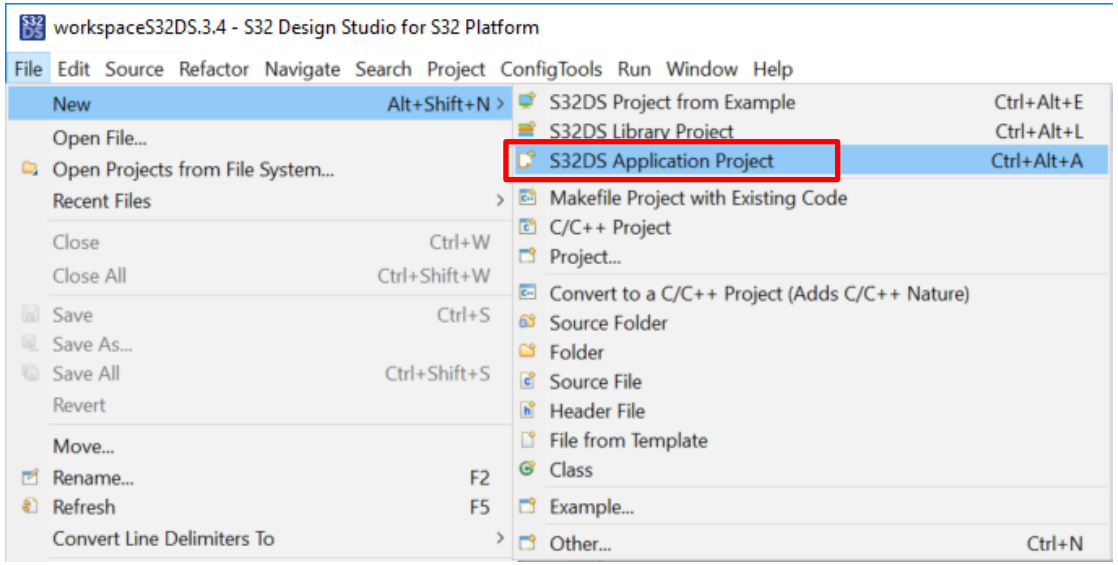
CREATING NEW PROJECTS

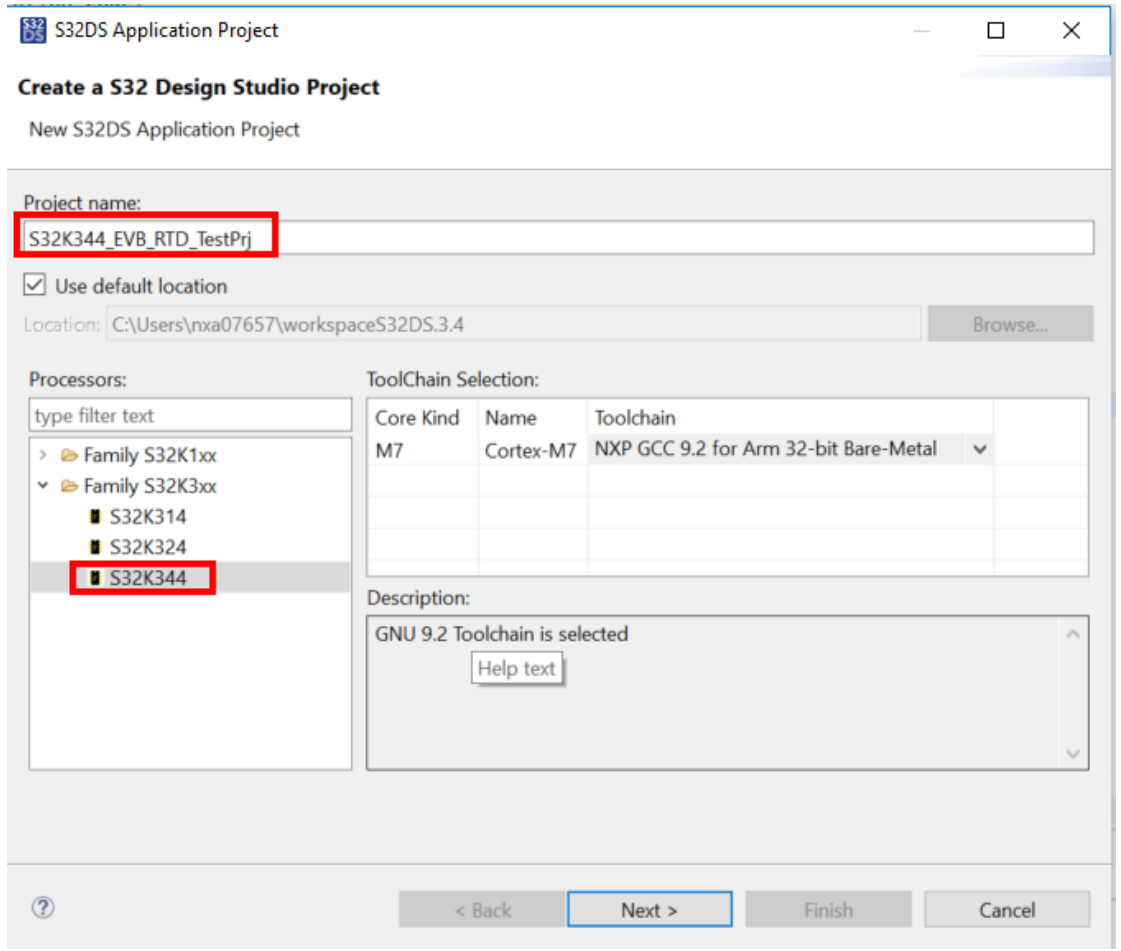
SECURE CONNECTIONS
FOR A SMARTER WORLD

CREATE NEW PROJECT

Select MCU S32K344

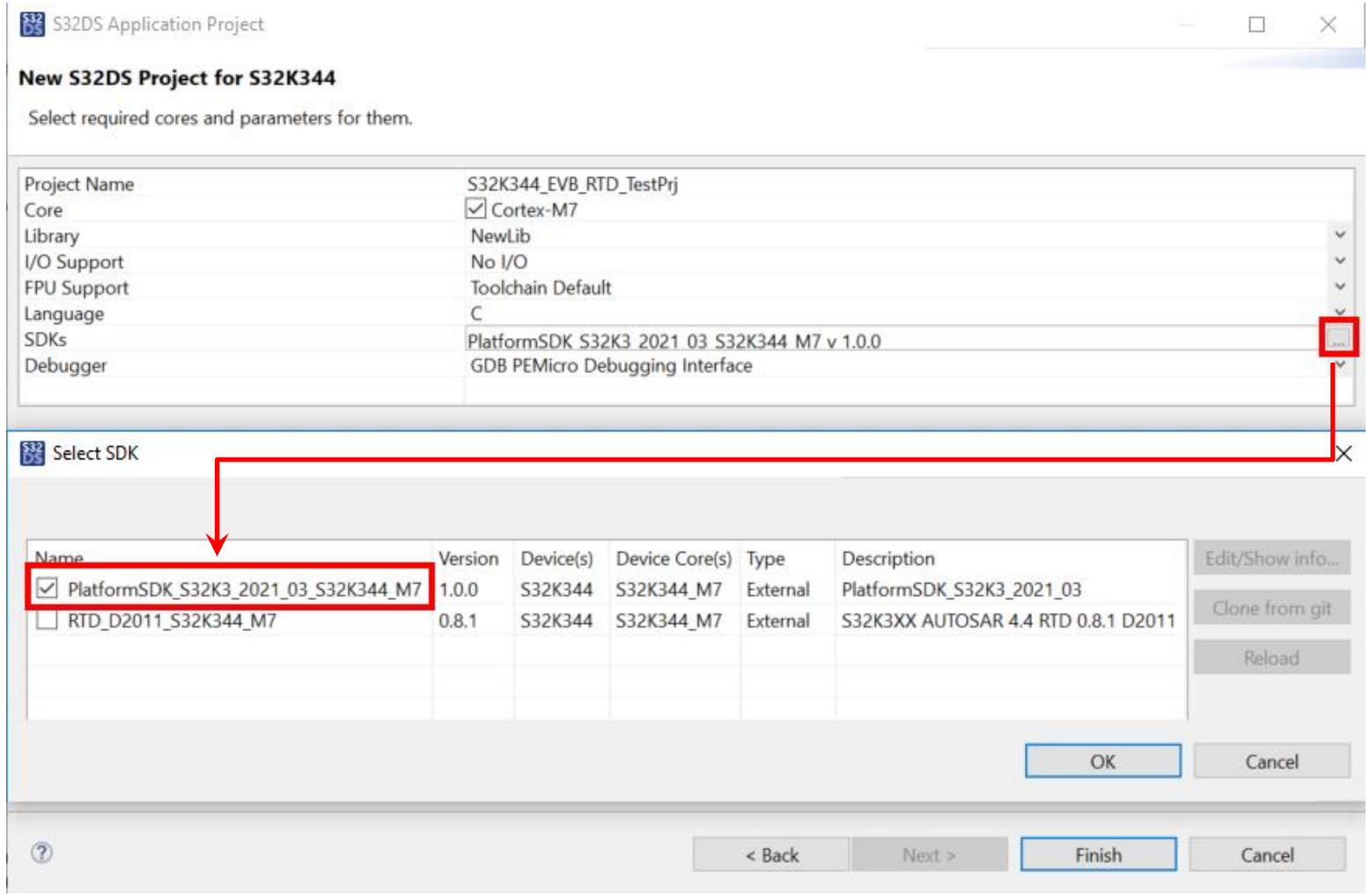
Set project name.



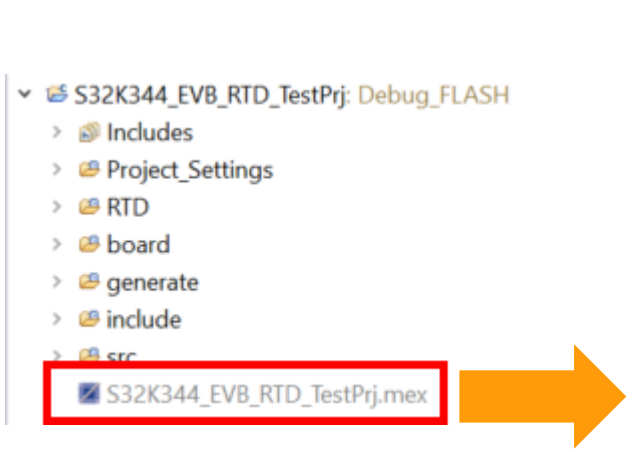


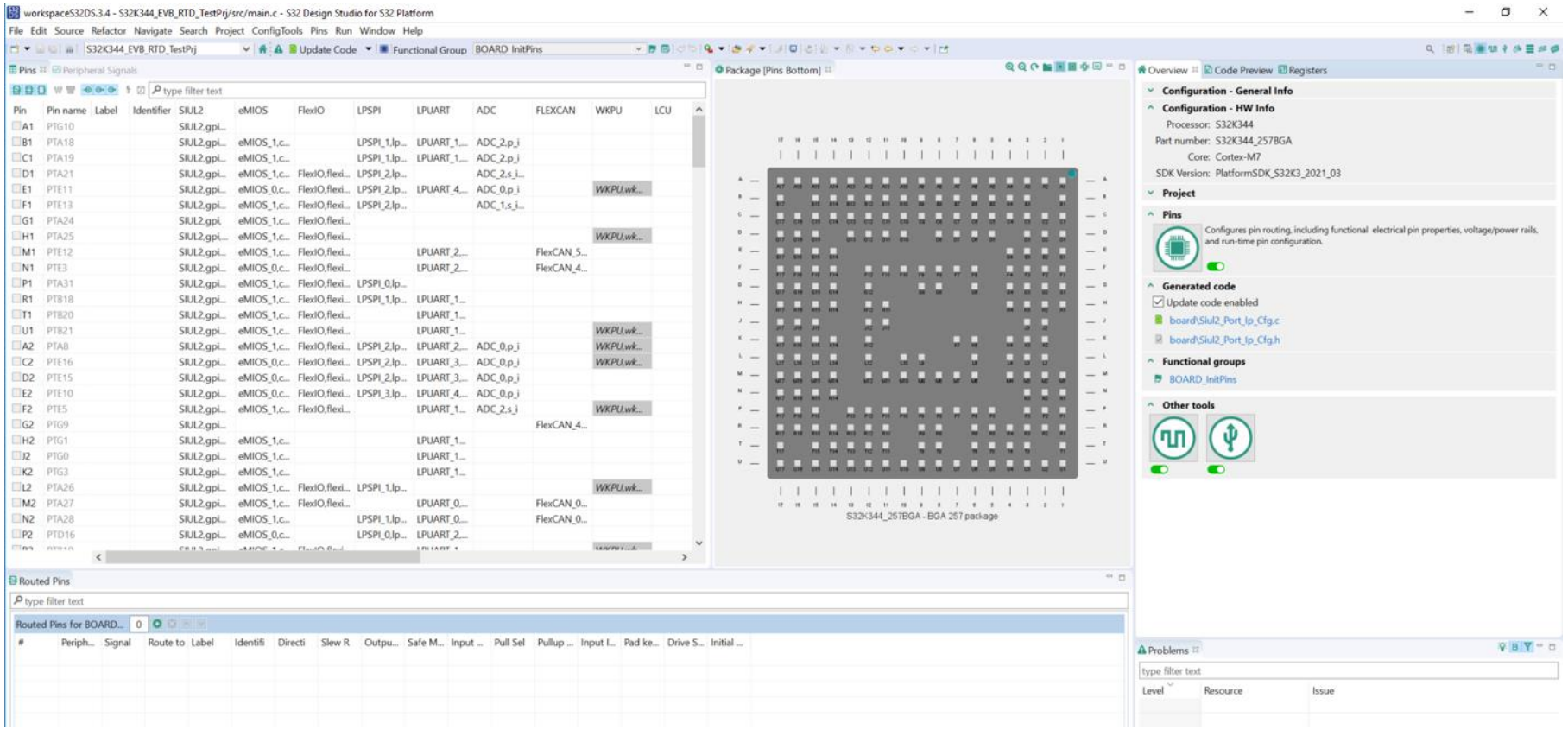
SELECT SDK FOR S32K3XX NEW PROJECT

Select the desired RTD version (“RTD_Dxxxx_xxxx”) for the new project.

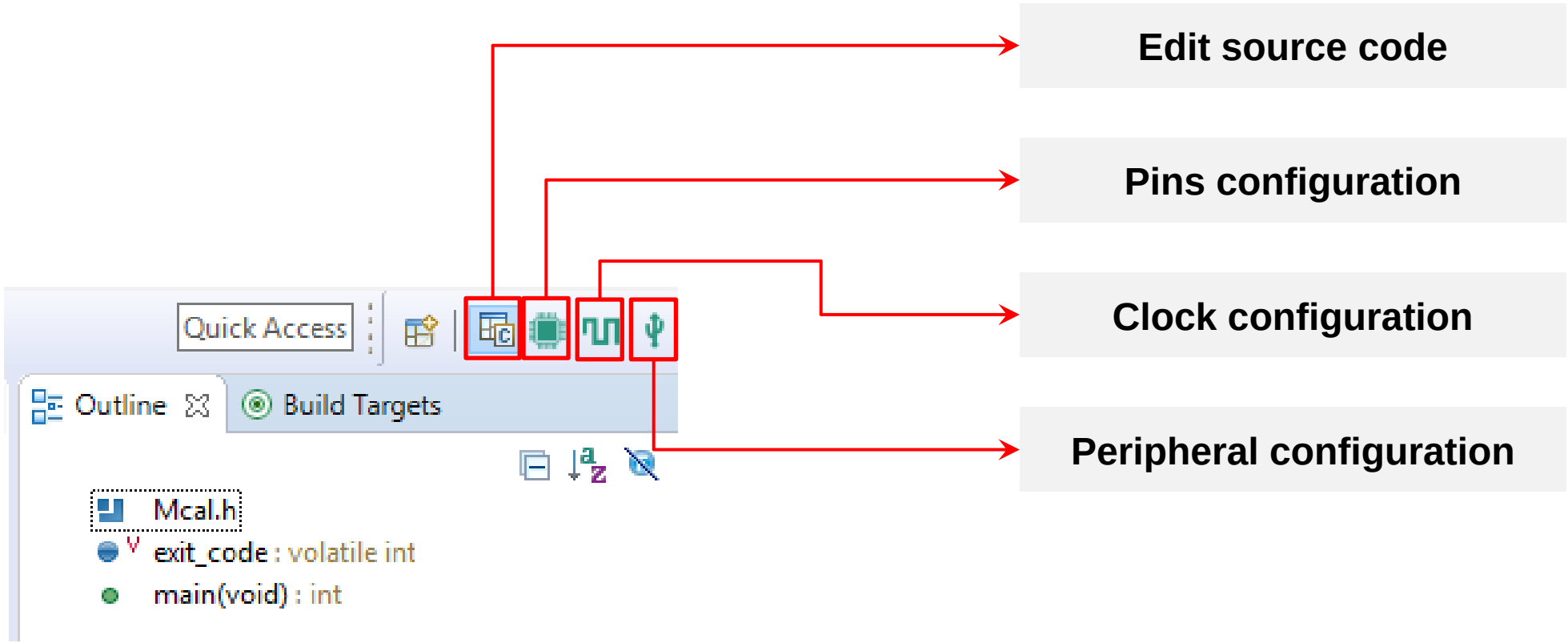


Double click the ".mex" file to open SDK config tool





Click the button on top right to switch between different configuration tools and source code editor.



REAL-TIME DRIVERS (RTD)

Learn more at nxp.com/RTD

Join us at NXP Connects!
Registration now open.

- NXP Connects EMEA: Nov 9-10, 2021
- NXP Connects AMEC: Nov 10-11, 2021
- NXP Connects APAC: Nov 16-17, 2021

SECURE CONNECTIONS
FOR A SMARTER WORLD

SHOWROOM.NXP.COM