NXP Semiconductors
Application Notes

Document Number: AN13423
Rev. 1, 03/2023

Using CAN2CAN, CAN2ETH and ETH2CAN Features of LLCE on S32G

by: NXP Semiconductors

1. Introduction

This application note is a complementary to the LLCE Getting Started Guide and the LLCE firmware user guide for using CAN2CAN, CAN2ETH and ETH2CAN in S32G.

These three are LLCE's key features to realize offloading CAN gateway tasks. LLCE has the capability to perform CAN frame routing between CAN channels (i.e. CAN2CAN) and between CAN and Ethernet (i.e. CAN2ETH / ETH2CAN) without host core's intervention. These feature reduces the routing latency and host core load. After going through this document, you will be able to understand what are those features and how to play them.

LLCE can perform the CAN frame routing according to the configured routing table without host CPU's load.

CAN2CAN: When the configured frame ID is coming into the configured CAN channels, LLCE routes it to the configured destination CAN channel(s).

CAN2ETH: When the configured frame ID is coming into the configured CAN channels, LLCE encapsulates the CAN frame into the Ethernet frame in IEEE1722 format and UDP packet. PFE sends it to the Ethernet.

ETH2CAN: When the PFE receives the Ethernet frame, LLCE parses it and unpacks the IEEE1722 packet /

Contents

UDP packet and route it to CAN channels. The following figure shows an overview diagram of CAN2CAN, CAN2ETH and ETH2CAN with respect to LLCE in S32G.

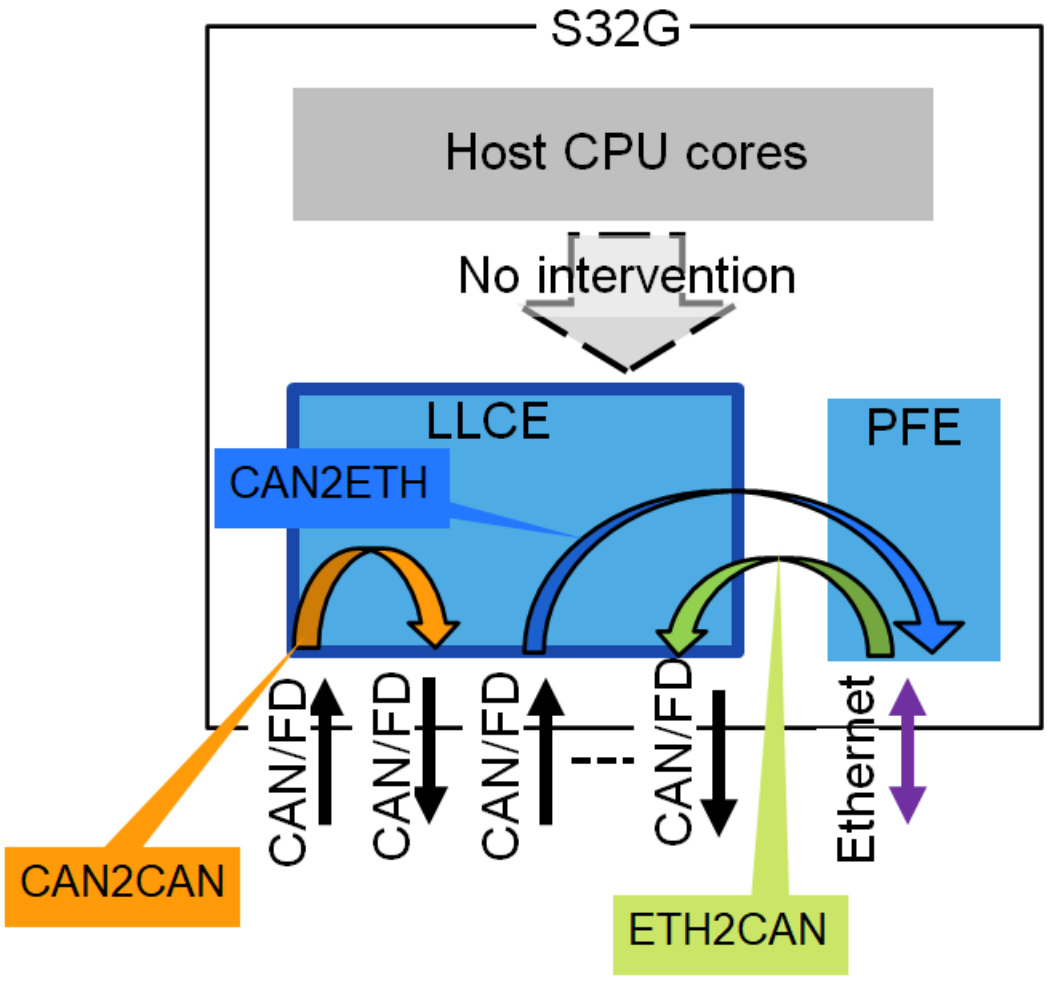


Figure 1. Overview diagram

2. CAN2CAN, CAN2ETH and ETH2CAN features

The CAN2CAN feature performs the off-loaded CAN frame routing according to the configured routing table. The following options are available:

- Multicast/Unicast: Not only single destination channel (i.e. Unicast) but also multiple destination channels (i.e. Multicast) can be configured.
- ID remapping: Remapping CAN frame ID can be configured. Switching Standard & Extended ID is also possible.
- Frame transformation between Classic CAN and CANFD can be configured.

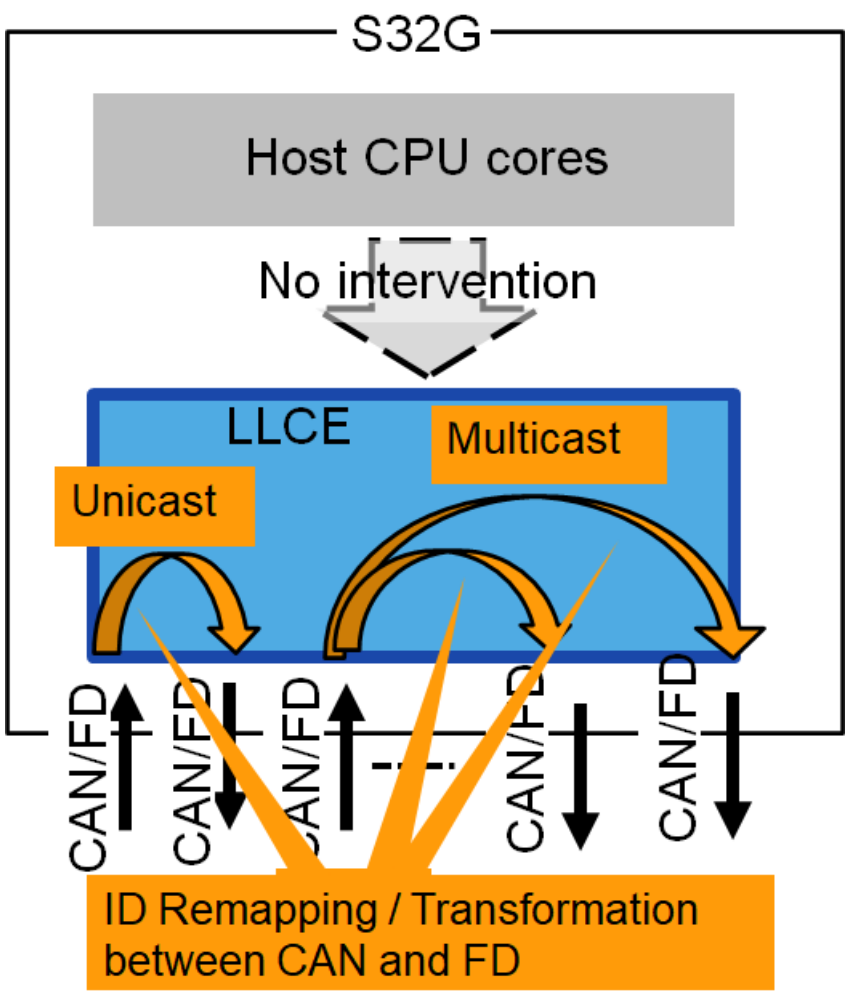


Figure 2. CAN2CAN feature

The CAN2ETH feature performs the off-loaded encapsulation which packs configured CAN frames into Ethernet frames in IEEE1722 format. The following steps are taken by CAN2ETH for encapsulations of the CAN frames:

- CAN frames are packed into IEEE1722 packet (Compliant to AVTP Time-Synchronous / Non time synchronous control format. Packed as ACF CAN Brief /Full messages) or packed into UDP packet.
- LLCE packs the message and put it on the buffers in the SRAM.
- The packet length is controlled by the configuration of the buffer size.
- Stream ID is constant, Not configurable.

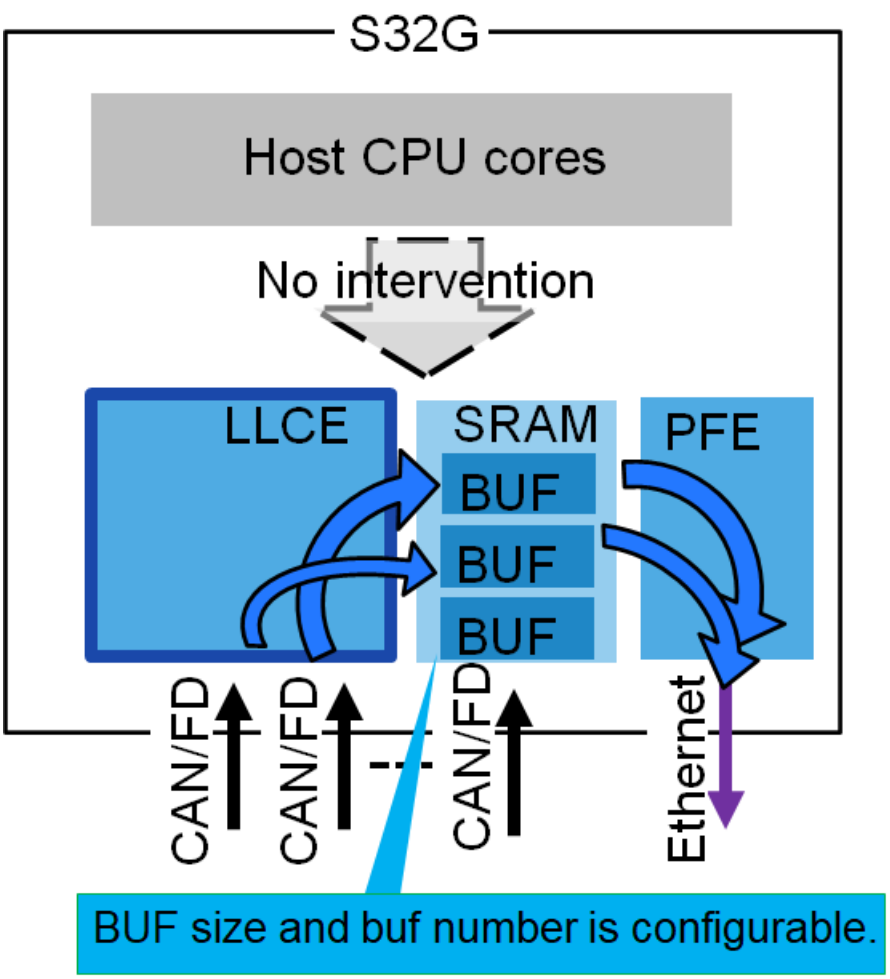


Figure 3. CAN2ETH feature

The ETH2CAN feature performs the off-loaded unpacking IEEE1722 AVTP frames. The following steps are taken for unpacking the frames:

- Any IEEE1722 frames /UDP packets will be parsed and unpacked and routed.
- The maximum number of ACF CAN frames inside one AVTP frame is limited by the number of HTH you configured. 16 frames per one channel is the maximum case.

NOTE

In order to avoid conflict between host application's Ethernet frame handling, be aware the LLCE FW is using PEF_HIF3 for CAN2ETH/ETH2CAN.

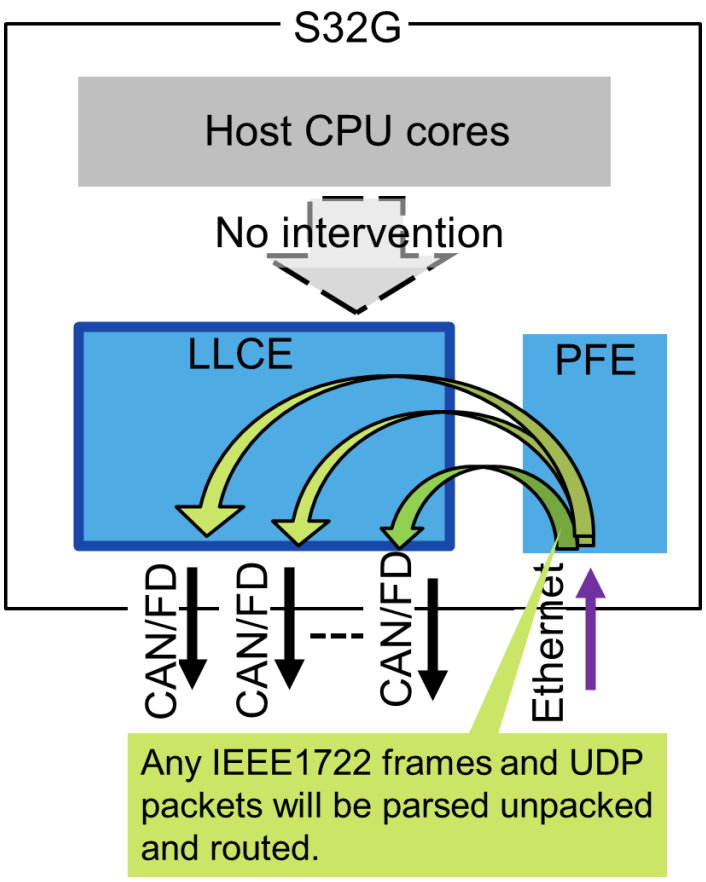


Figure 4. ETH2CAN feature

3. Using sample application

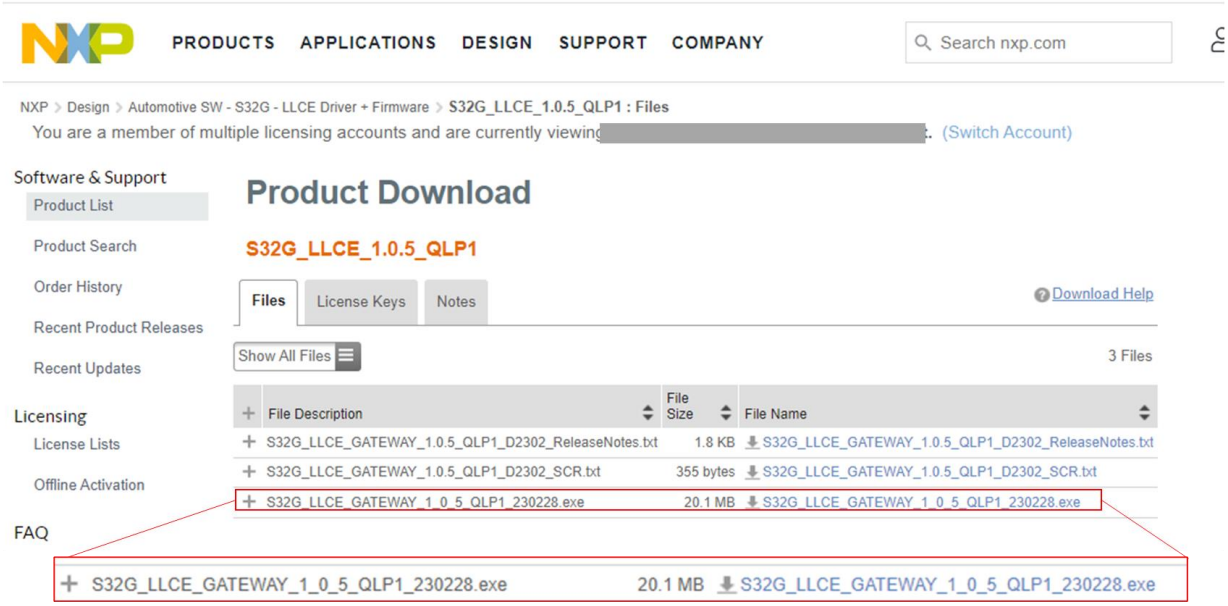
This section and sub sections describes how to use the sample application. The steps that needs to be followed are shown in the section.

NOTE

This section is based on the latest release as of February 2023. (i.e. S32G_LLCE_GATEWAY_1_0_5_QLP1_D230228.exe). If you are using newer version, the contents described in this chapter may be different.

3.1. Downloading and installing the LLCE package

Go to FLEXERA and download the latest LLCE software package. After download install the package. Refer to the following screenshot.



In this document, assuming you are installing the LLCE SW into the default location (i.e. C:/NXP/S32G_LLCE_1_0_5_QLP1)

Figure 5. Downloading and installing LLCE

After installation of LLCE SW package, put the bundled plugins folders and files into the tresos/plugins as shown below.

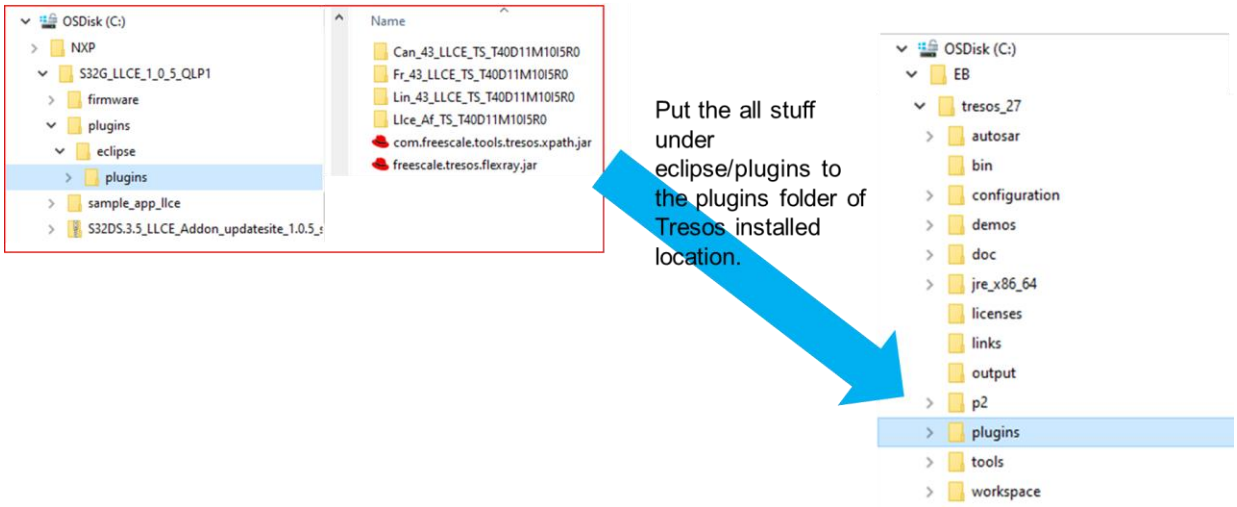


Figure 6. LLCE SW package in TREOS

If you do not have PFE MCAL 4.4 driver 1.0.0 and RTD package, download both of them.

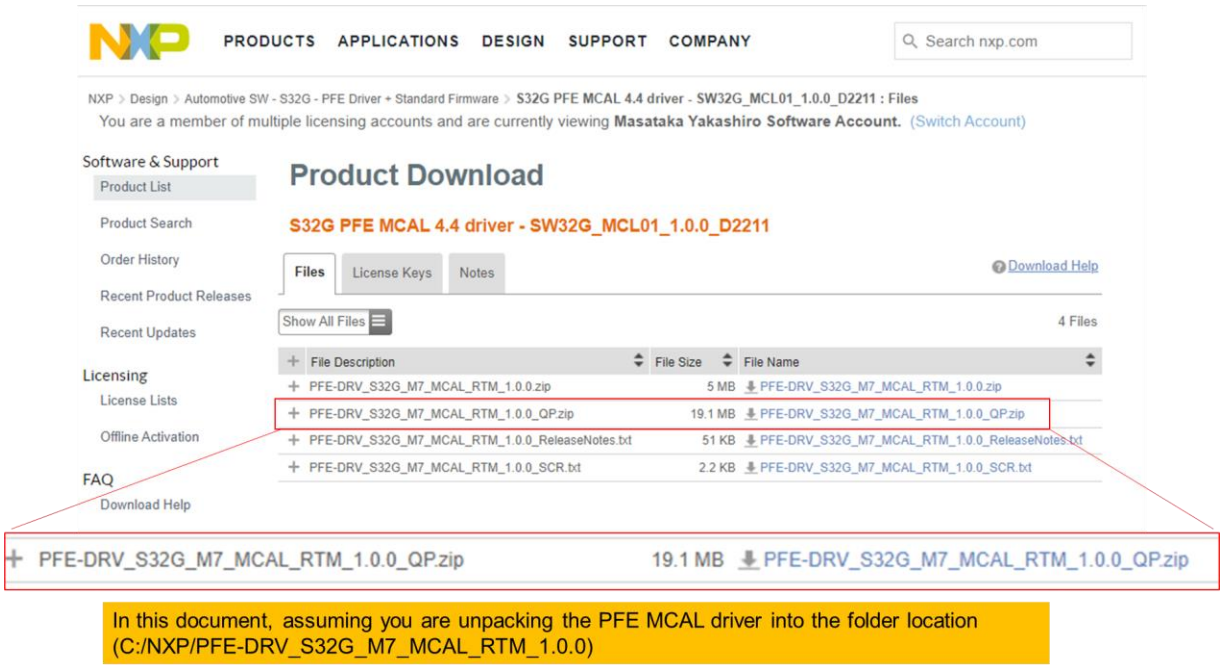


Figure 7. PFE MCAL 4.4 driver

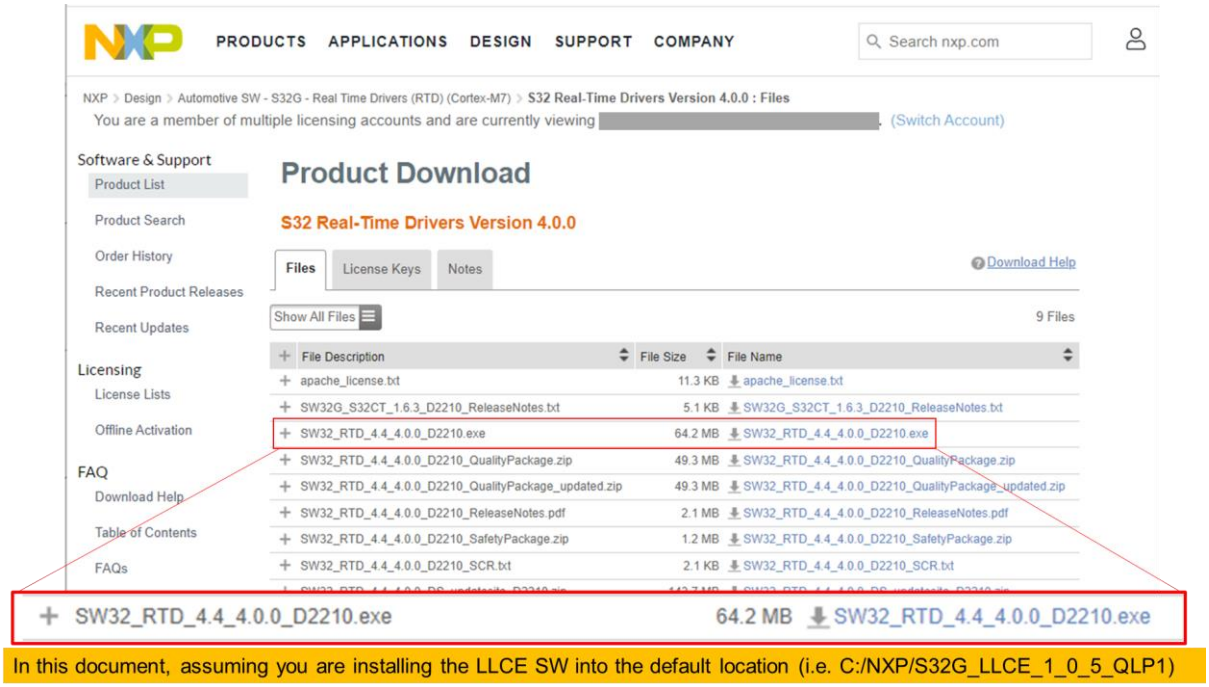


Figure 8. RTD package

After the installation of PFE MCAL and RTD, put the folders and files into the tresos/plugins as shown below.

Figure 9. PFE MCAL and RTD in tresos

For llce_sample_app_pfe, modify the sample_app_initialization.c. as shown in the following screenshot.

3.2. Modifying the files and make

Modify the config.mk for your environment. For CAN2CAN, modify the file: C:\NXP\S32G_LLCE_1_0_5_QLP1\sample_app_llce\llce_sample_app_af\config.mk.

Please refer to the following screenshot.

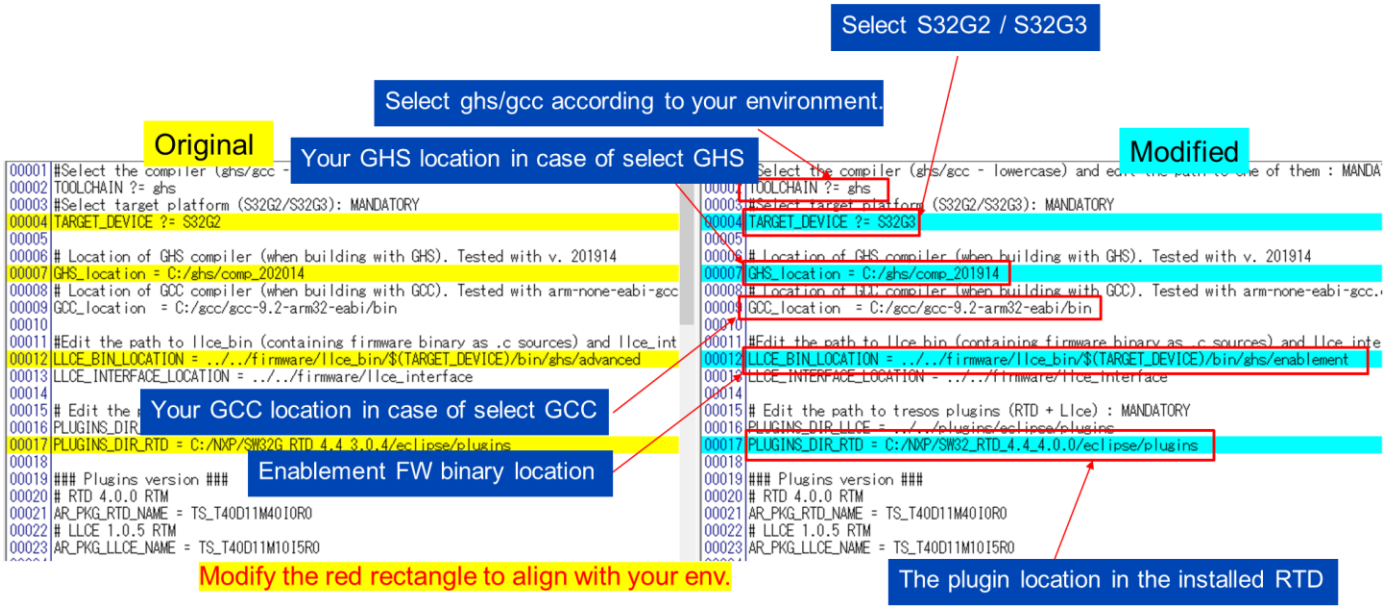


Figure 10. Modifying config.mk

In the S32G_LLCE_1_0_5_QLP1 release, there is one known-issue in S32G3's EB-Tresos config of the CAN2CAN sample app, which is needed to fix. In default, the interrupt is disabled in the S32G3's CAN2CAN sample app config. So, change it to Enabled as following flow.

Run EB-Tresos Studio and import sample config.

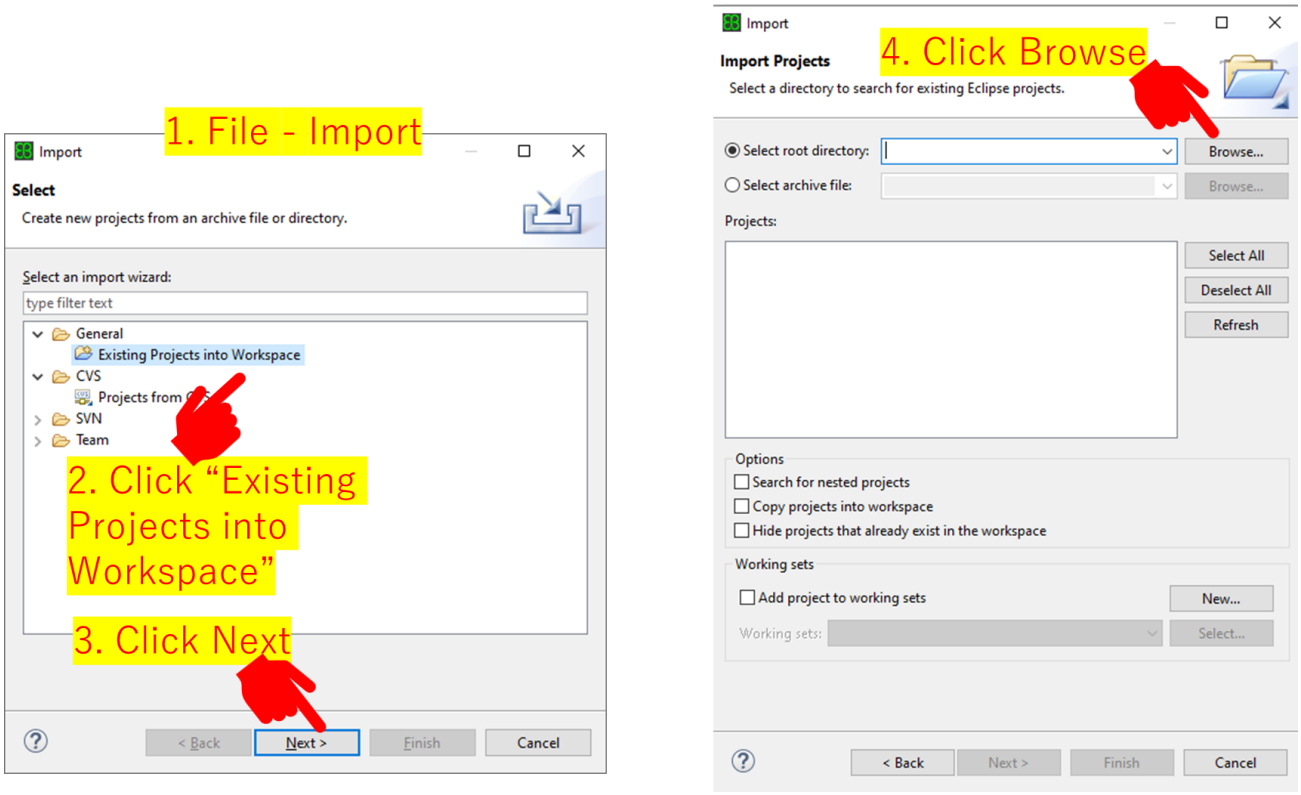


Figure 11. Import Sample config of CAN2CAN-G3

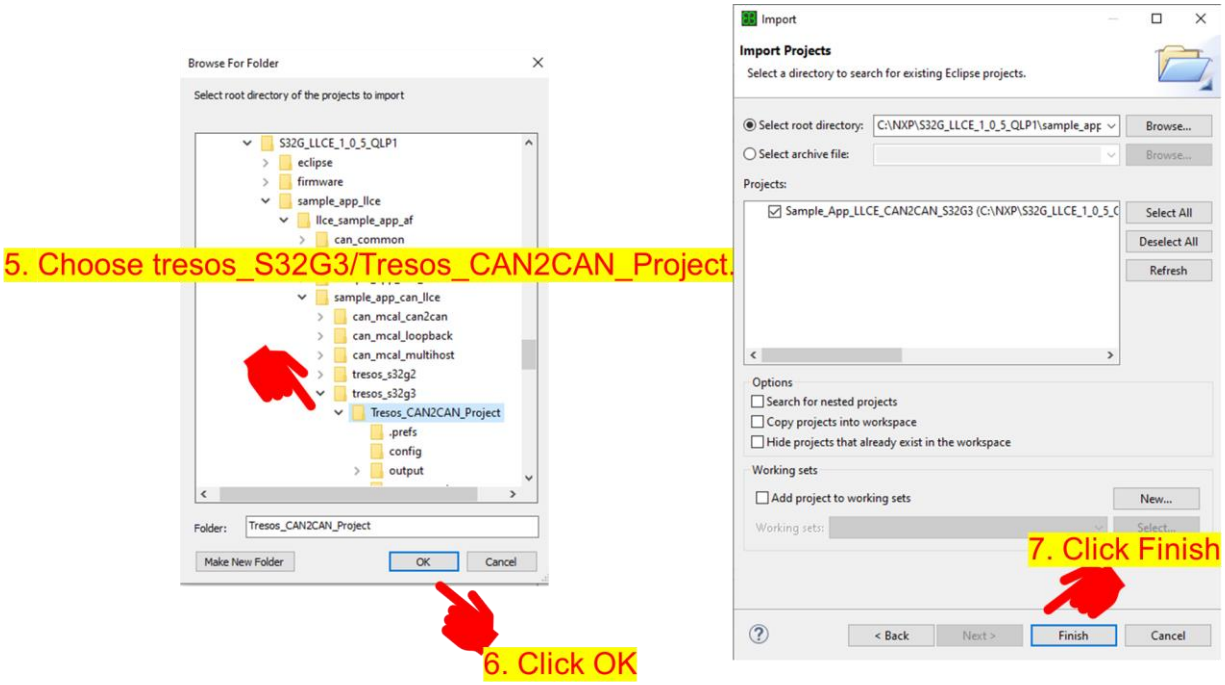


Figure 12. Import Sample config of CAN2CAN-G3

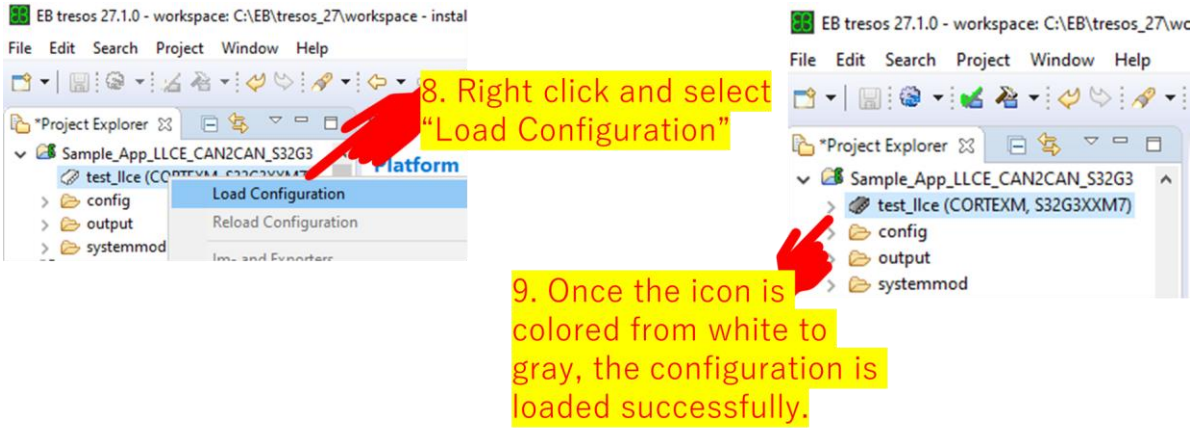


Figure 13. Load Config

Run EB-Tresos Studio and import sample config.

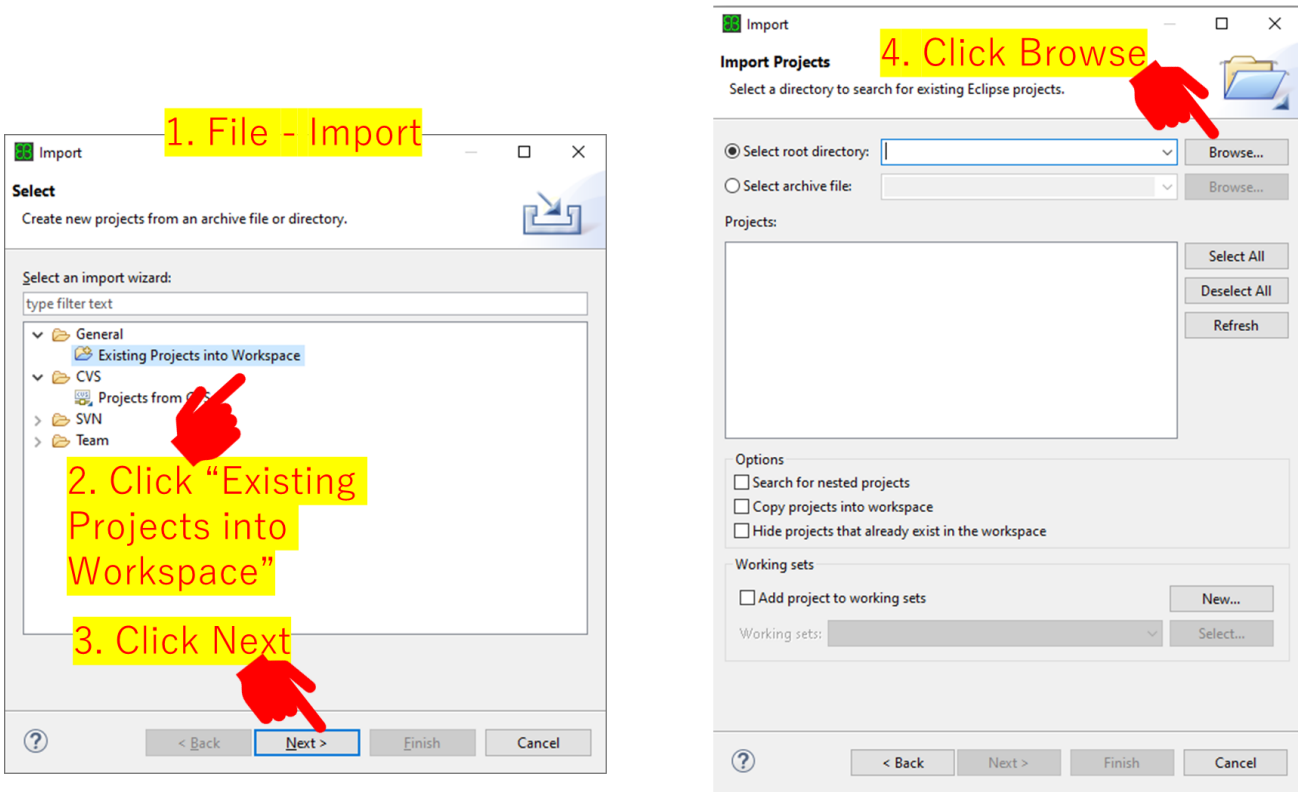


Figure 14. Import sample config

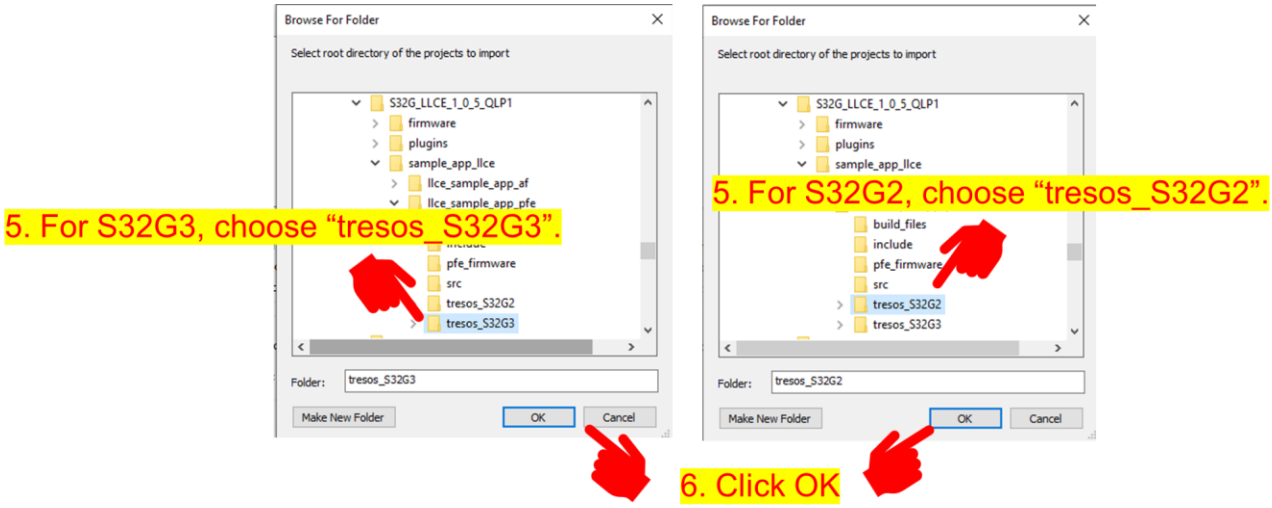


Figure 15. Import Sample Config

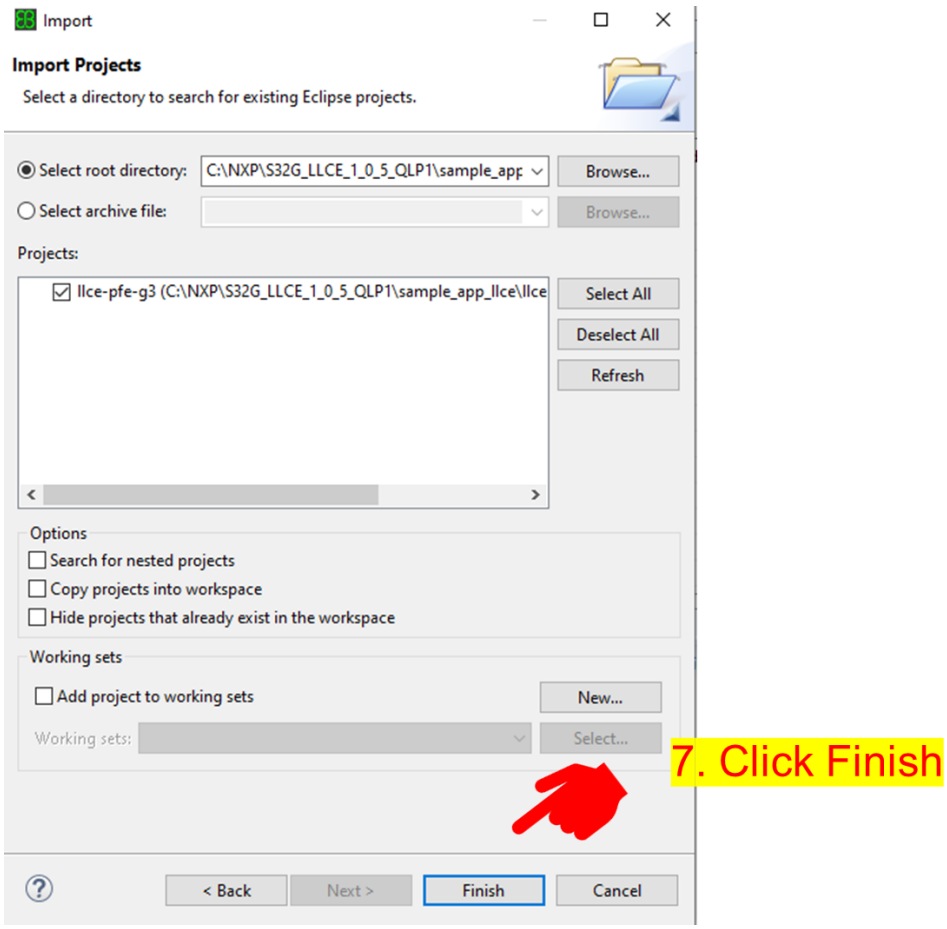


Figure 16. Import Sample Config

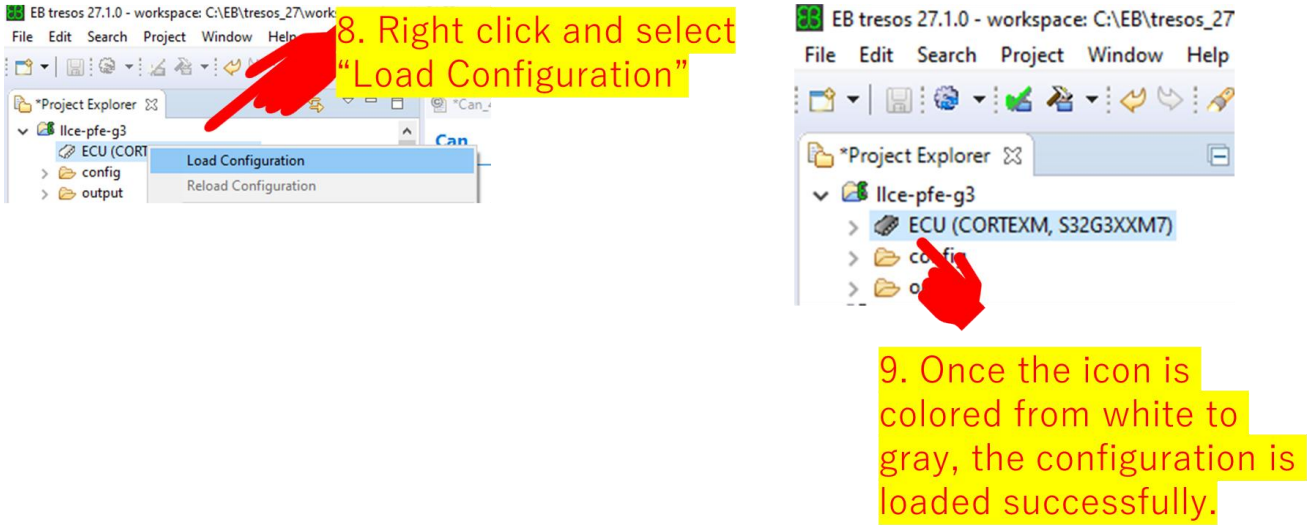


Figure 17. Load config of CAN2CAN-G3

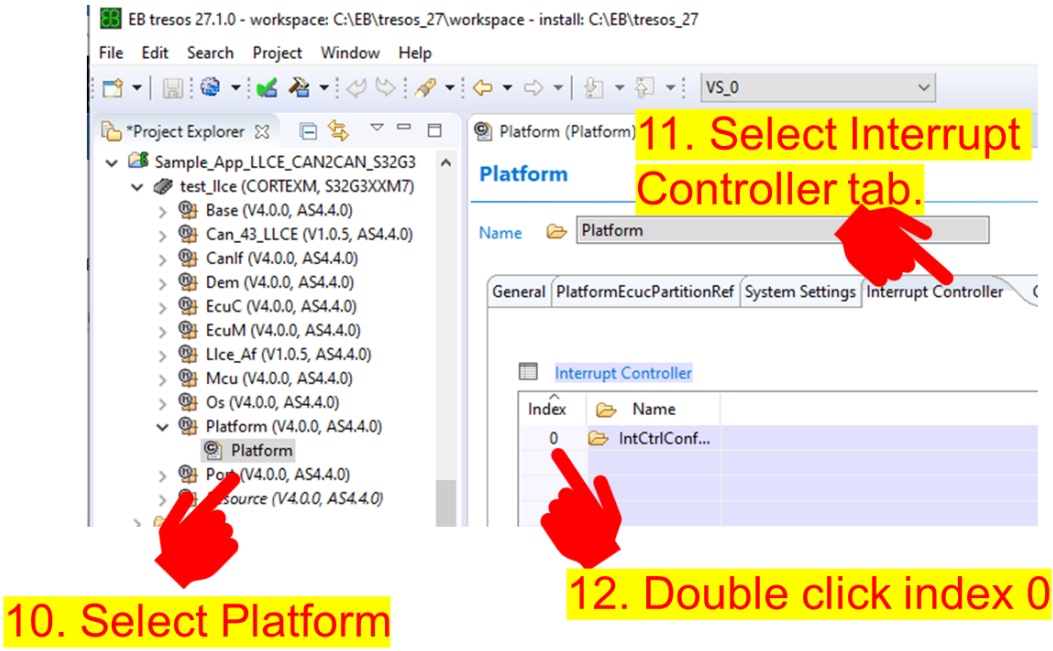


Figure 18. Enable the interrupt on Tresos

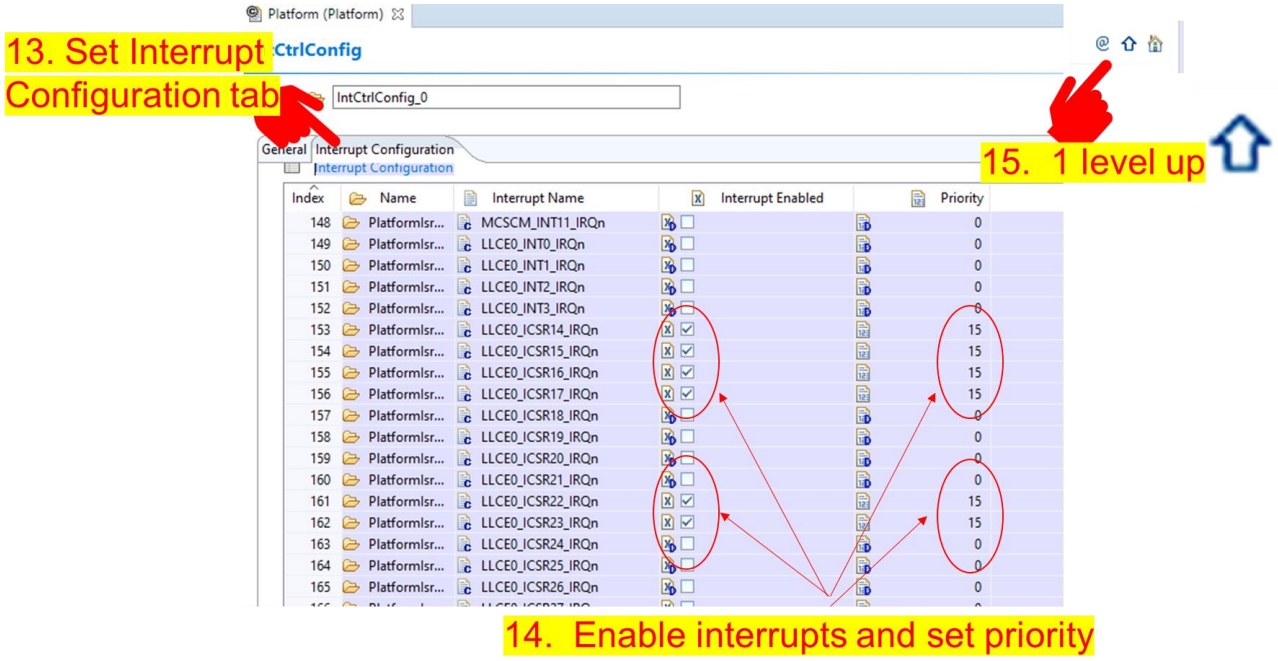


Figure 19. Enable the interrupt on Tresos

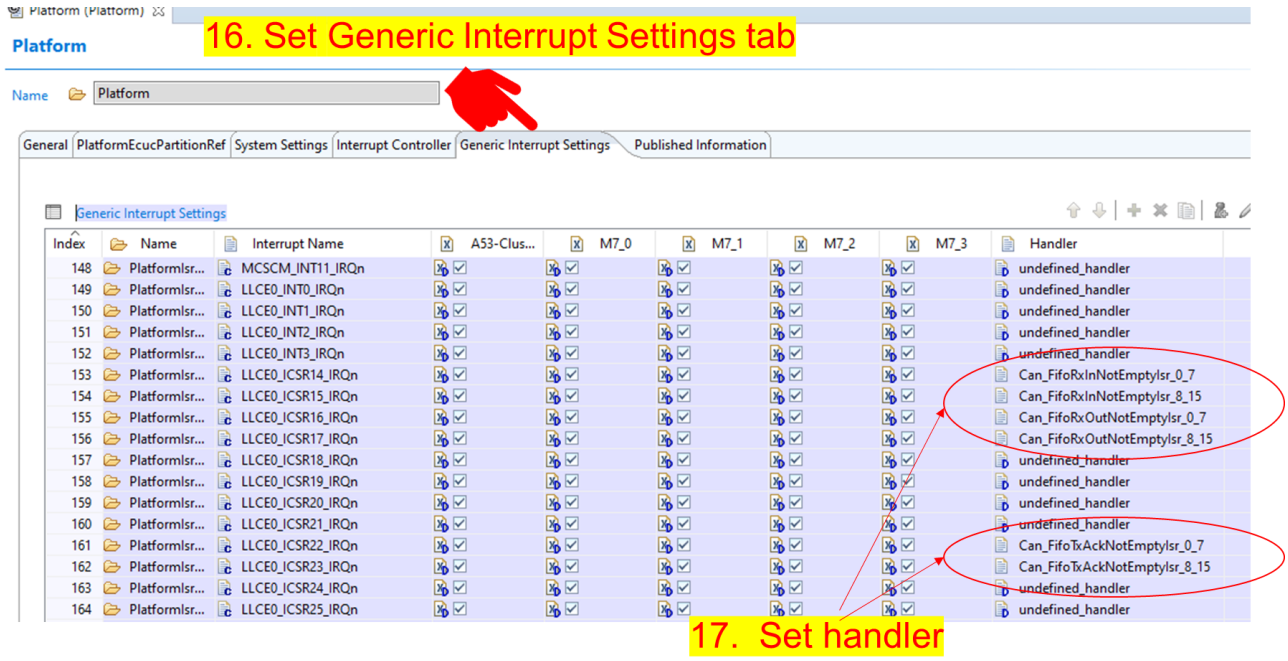


Figure 20. Enable the interrupt on Tresos

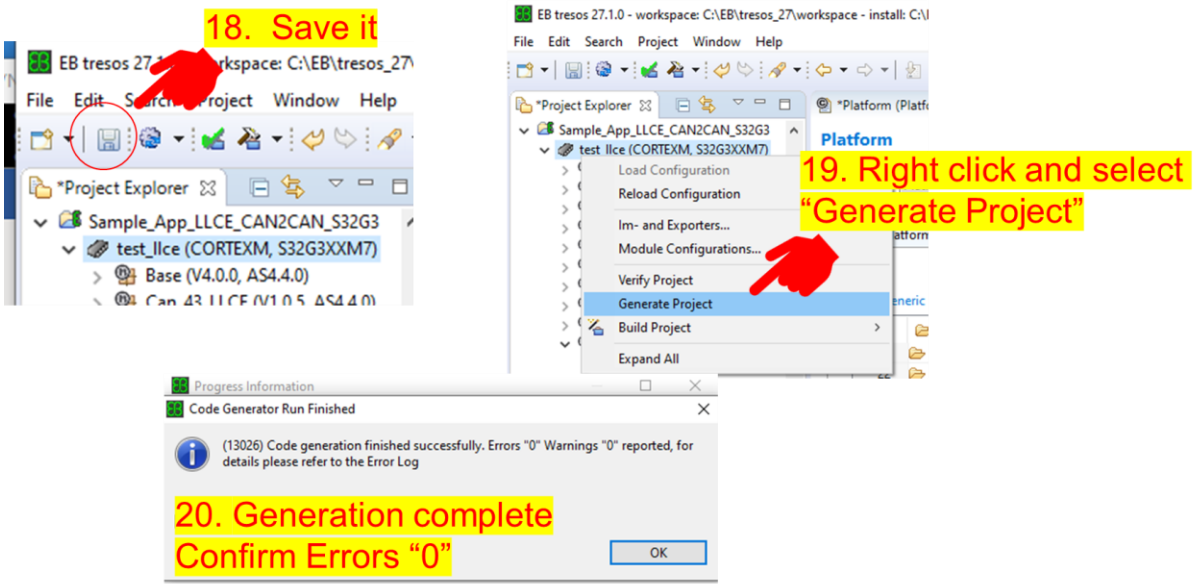


Figure 21. Generate Code

Now CAN2CAN sample app is ready to build.

Under llce_sample_app_af folder, you can build as following.

- $make clean
- $make can_routing

Then, you can see the elf file "can_routing.elf" under llce_sample_app_af/build.

As for the CAN2ETH/ETH2CAN sample app, modify the config.mak for your environment. For CAN2ETH/ETH2CAN, modify C:\NXP\S32G_LLCE_1_0_5_QLP1\sample_app_llce\llce_sample_app_pfe\config.mak

Please refer to the following screenshot.

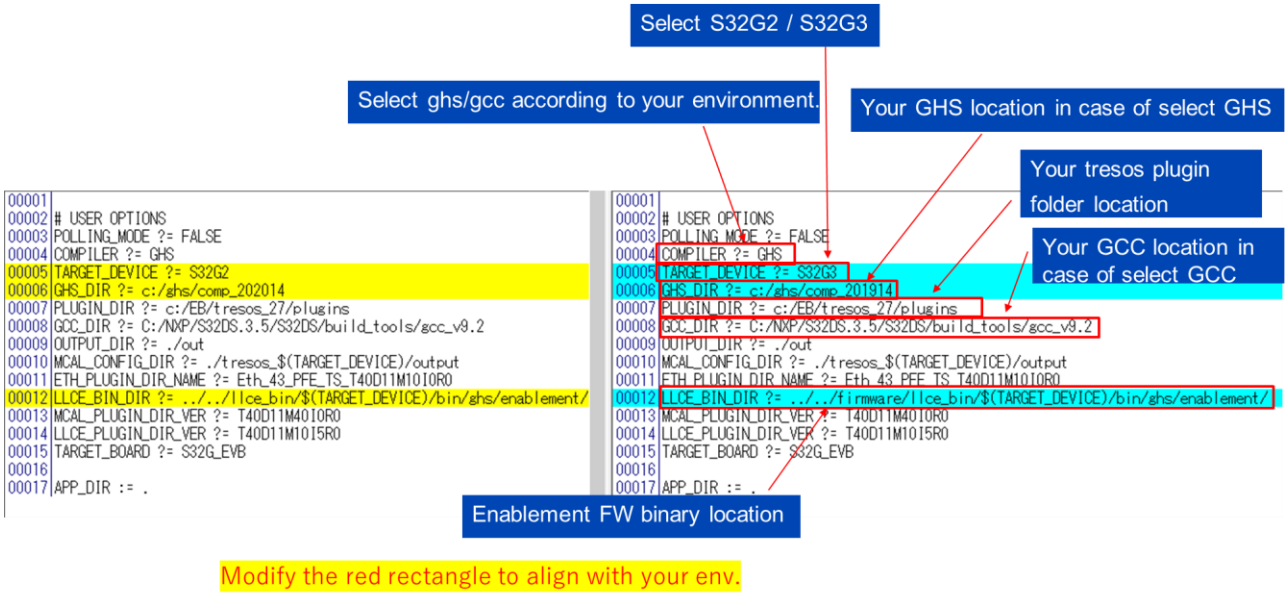


Figure 22. Modifying config.mak

In the S32G_LLCE_1_0_5_QLP1 release, there is one known-issue in the EB-Tresos config of the llce_sample_app_pfe, which is needed to fix. As for S32G3, you should modify C:\NXP\S32G_LLCE_1_0_5_QLP1\sample_app_llce\llce_sample_app_pfe\tresos_S32G3\config\Eth.xdm. As for S32G2, you should modify C:\NXP\S32G_LLCE_1_0_5_QLP1\sample_app_llce\llce_sample_app_pfe\tresos_S32G2\config\Eth.xdm.

As for S32G3, change value from 2048 to 1522 on line 223, 429 and 618 as below. For S32G3, on line 269,521,and 756.

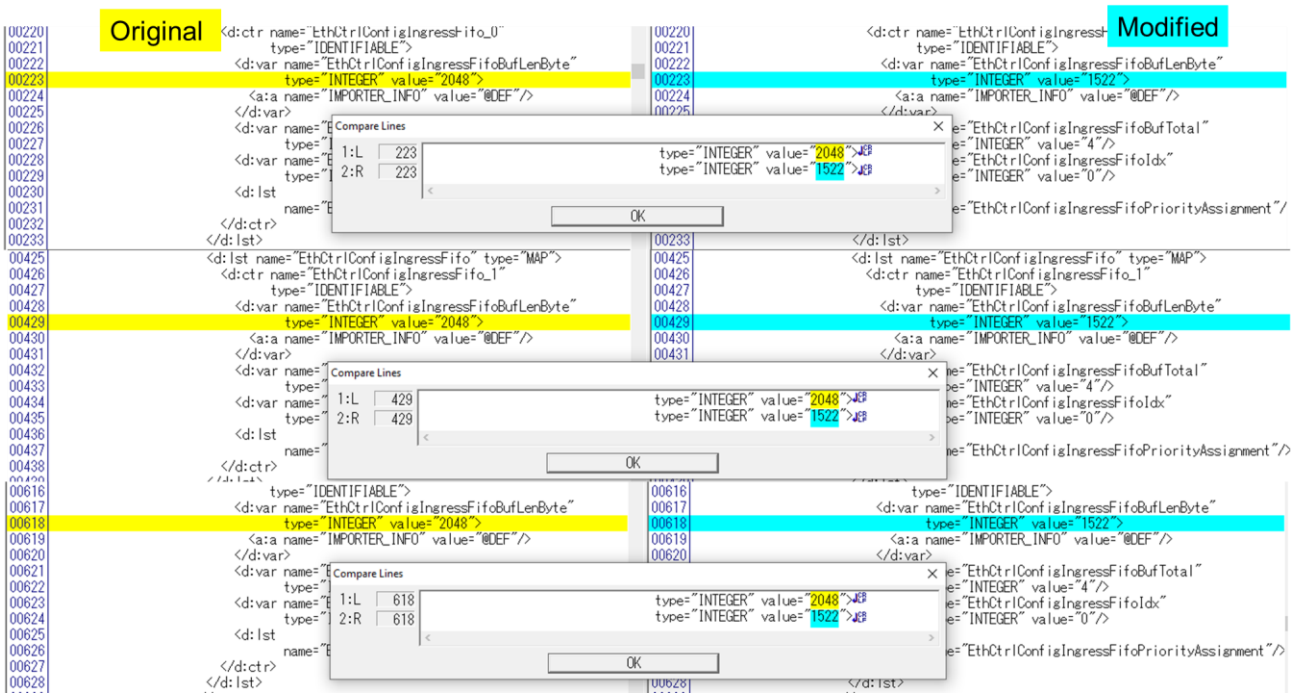


Figure 23. Modifying Eth.xdm

In PFE MCAL 4.4 driver 1.0.0, there is a known-issue in the Eth_43_PFE_TS_T40D11M10I0R0\include\hal.h. So fix the copied file under tresos/plugins as follows. Change symbol from “GHS” to “__ghs__”.

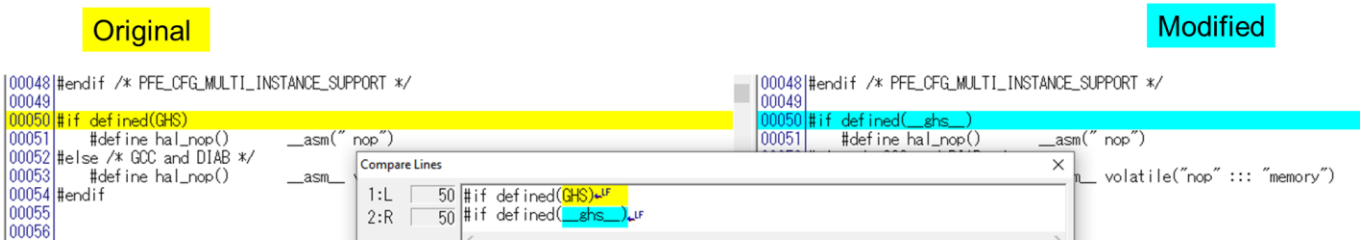


Figure 24. Fixing hal.h

In case of S32G2, follow below steps 1) – 11) before generating the config code.

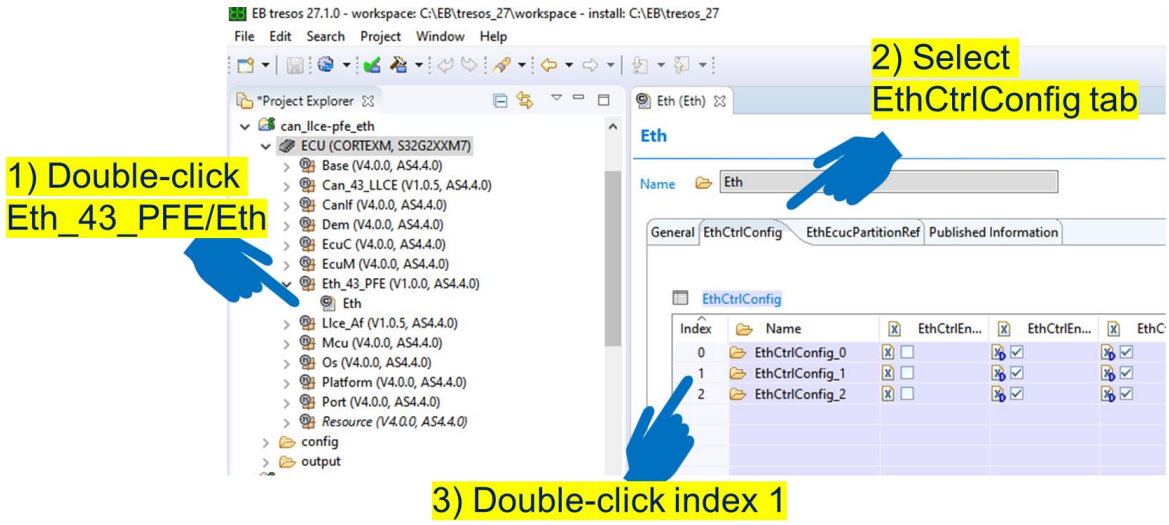


Figure 25. For S32G2, fix Eth config

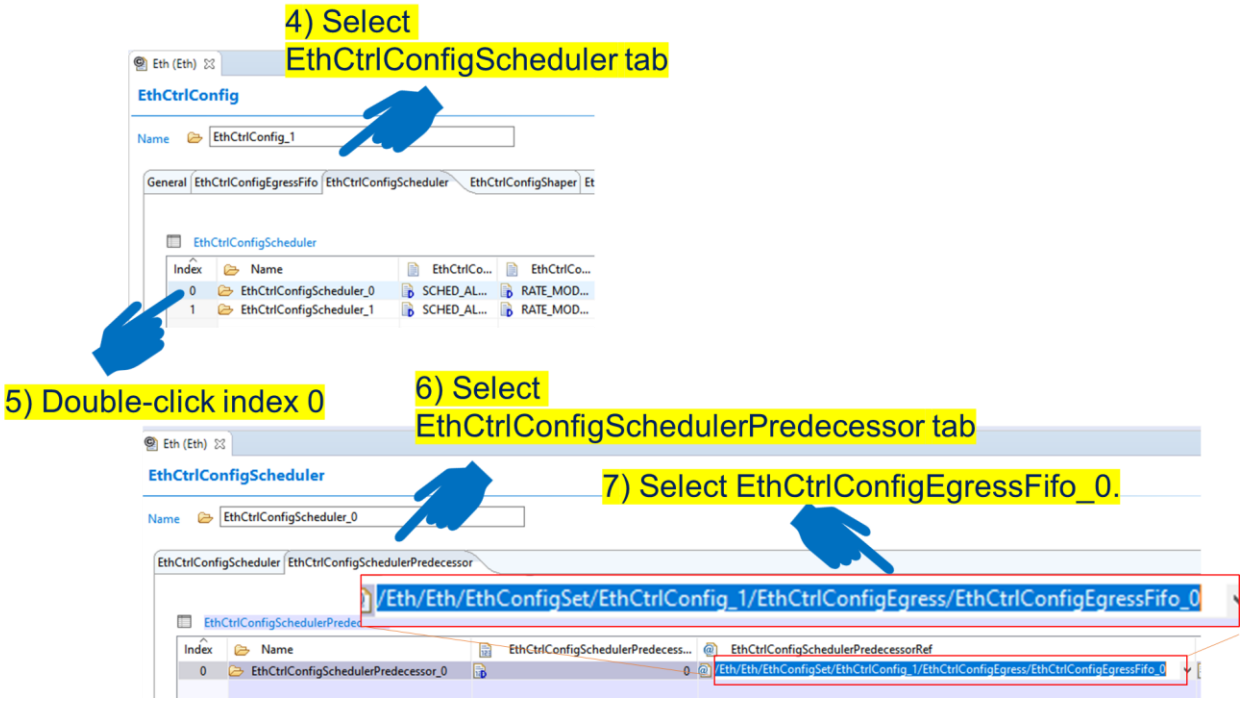


Figure 26. For S32G2, fix Eth config

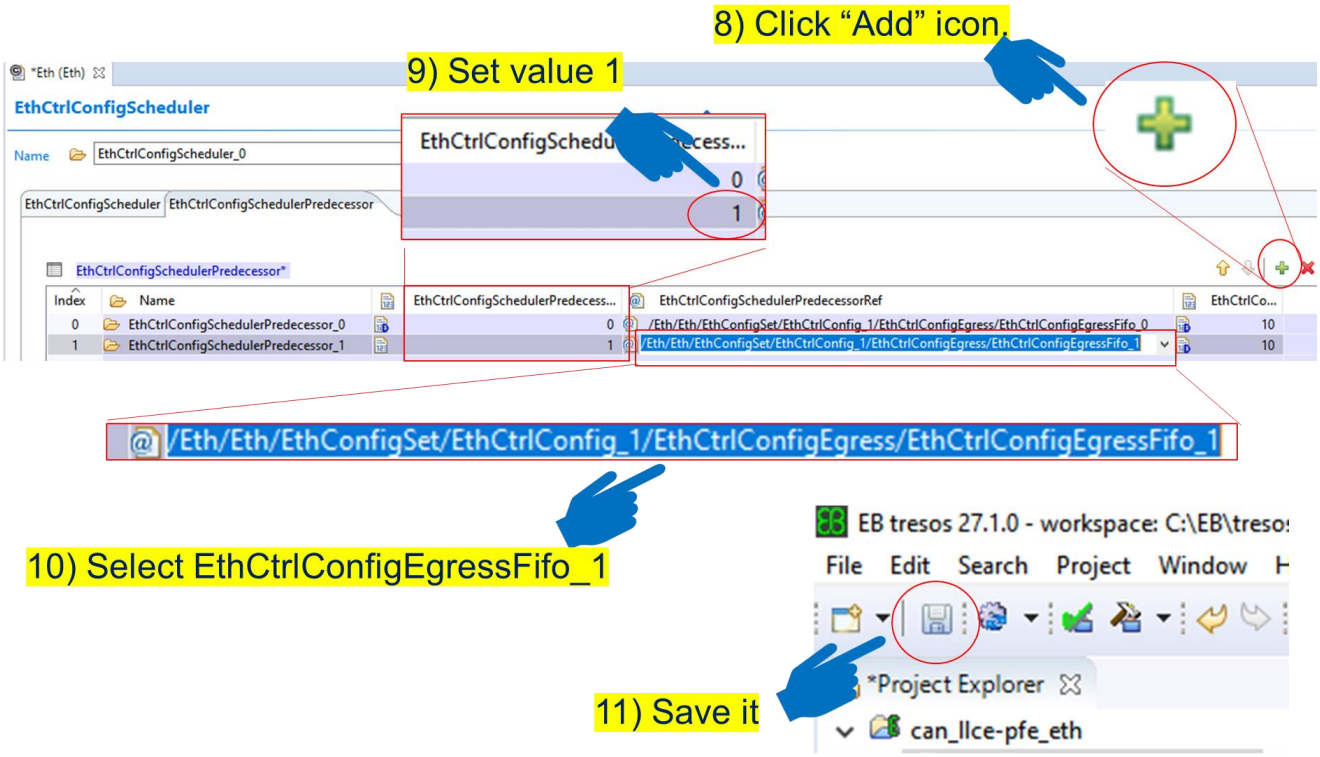


Figure 27. For S32G2, fix Eth config

Then, generate the config code.

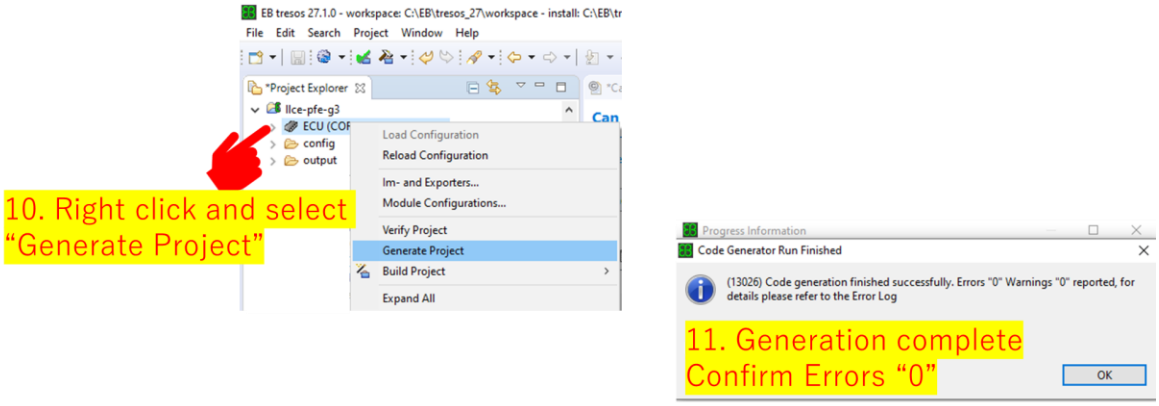


Figure 28. Generate config code

Under llce_sample_app_pfe folder, you can build as following.

- $make clean
- $make

You can see the elf file “int_app.elf” under llce_sample_app_pfe/out.

3.3. Connect the wires and run

For CAN2CAN, connect the CAN wires between CAN0 and 1, CAN14 and 15. After connecting the wires run the bundled CMM.

The CAN routing sample app performs CAN2CAN routing from CAN0 to CAN15. CAN1 sends the frames to be routed. Connect the external CAN wires between CAN0 and CAN1.

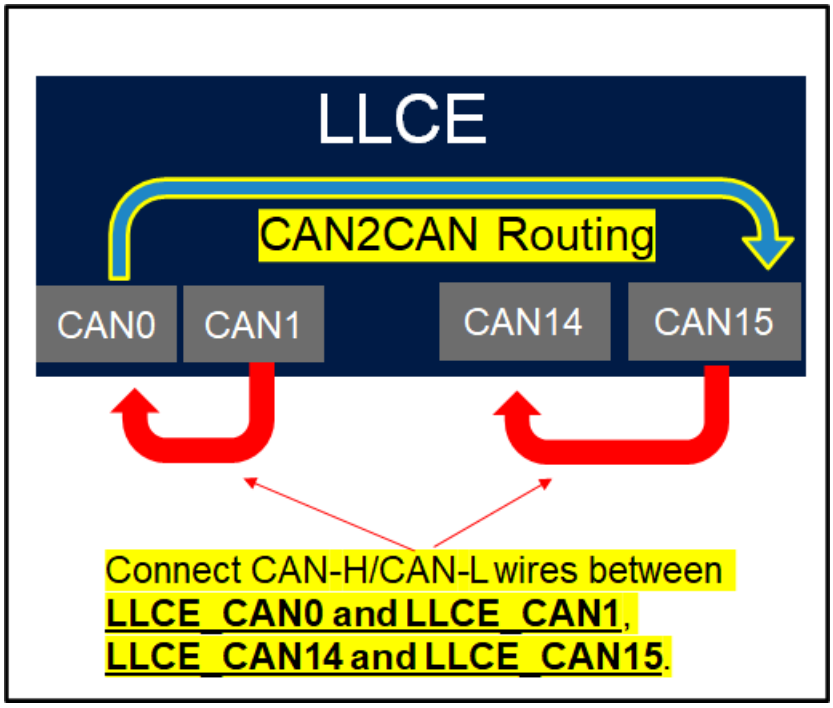


Figure 29. Connecting the CAN wires

You can see the Lauterbach's cmm scripts to run the sample app under folder S32G_LLCE_1_0_5_QLP1\sample_app_llce\llce_sample_app_af\tools\cmm_scripts. "S32G2_app_load.cmm" and "S32G3_app_load.cmm" are there. The former is for S32G2. The latter is for S32G3.

In the CMM, select CAN_ROUTING_DEBUG_MODE instead of CAN_LOOPBACK as below. Then, you can debug the sample app for CAN2CAN on TRACE32.

```
45 ;GOSUB CAN_LOOPBACK
46 ;GOSUB CAN_LOOPBACK_DEBUG_MODE
47
48 ;GOSUB LIN_LOOPBACK
49 ;GOSUB LIN_LOOPBACK_DEBUG_MODE
50
51 ;GOSUB CAN_ROUTING
52 GOSUB CAN_ROUTING_DEBUG_MODE
```

If you capture the two CAN buses with Logic Analyzer, you can see the routings as shown in the following figure.

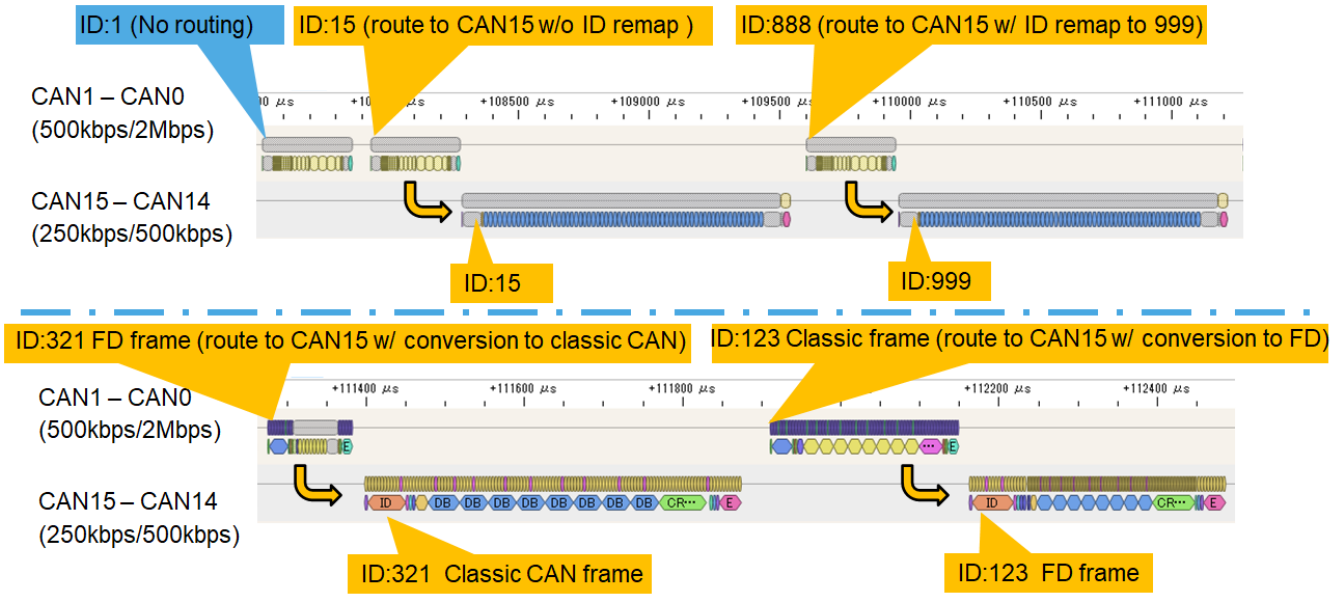


Figure 30. CAN routings

For CAN2ETH, Connect CAN wires between CAN0 and 1. Also Connect Ethernet cable to PFE_MAC1, run the bundled CMM.

CAN0 sends 64 CANFD frames. If you connect CAN wires between CAN0 and CAN1, CAN1 receives those frames and encapsulates them into IEEE1722 packets and into UDP packets. Then, LLCE sends the packets to PFE without host CPU's intervention. Then PFE sends them from PFE_MAC1.

If you connect an Ethernet cable between your PC and PFE_MAC1 (For RDB2/RDB3, it corresponds to P3A connector as shown below), you can capture those routed packets by your PC (e.g. Wireshark).

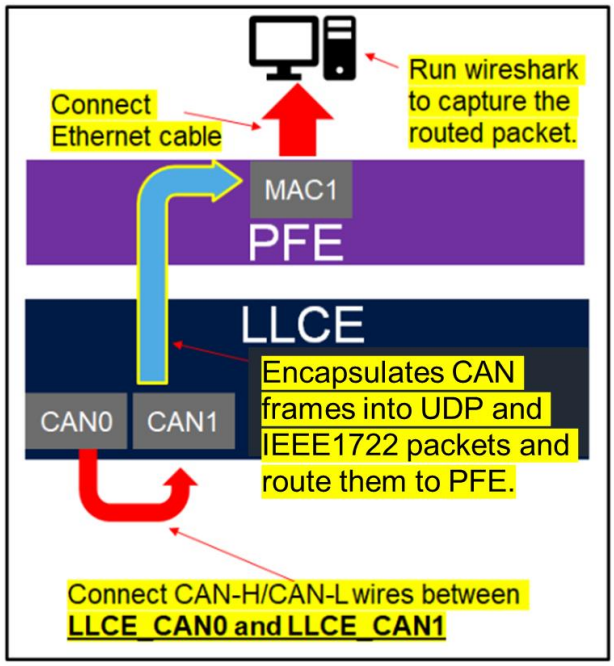
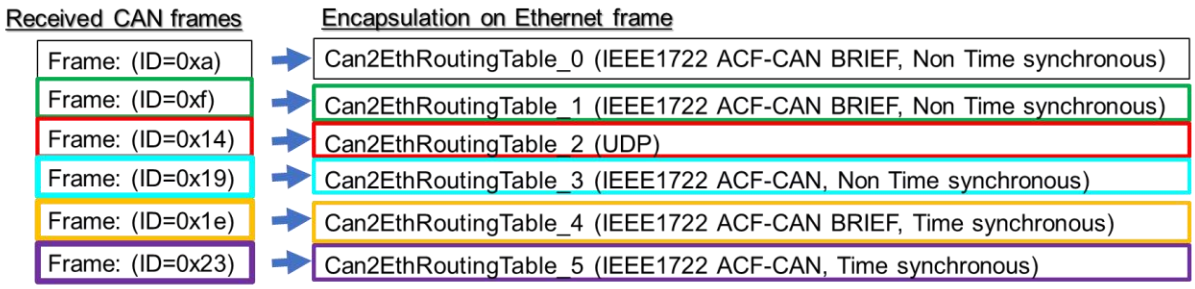


Figure 31. CAN2ETH routing

Run the Lauterbach's cmm script "*s32g.cmm*" under folder S32G_LLCE_1_0_5_QLP1\sample_app_llce\llce_sample_app_pfe.

You can debug sample app on TRACE32.

If you capture the routed packets, you can see encapsulated CAN frames. CAN0 sends frames which has seven kinds of IDs (ID=0x5,0xa,0xf,0x14,0x19,0x1e and 0x23). With this app's Tresos config, each CAN frames except ID=0x5 are processed as shown below.



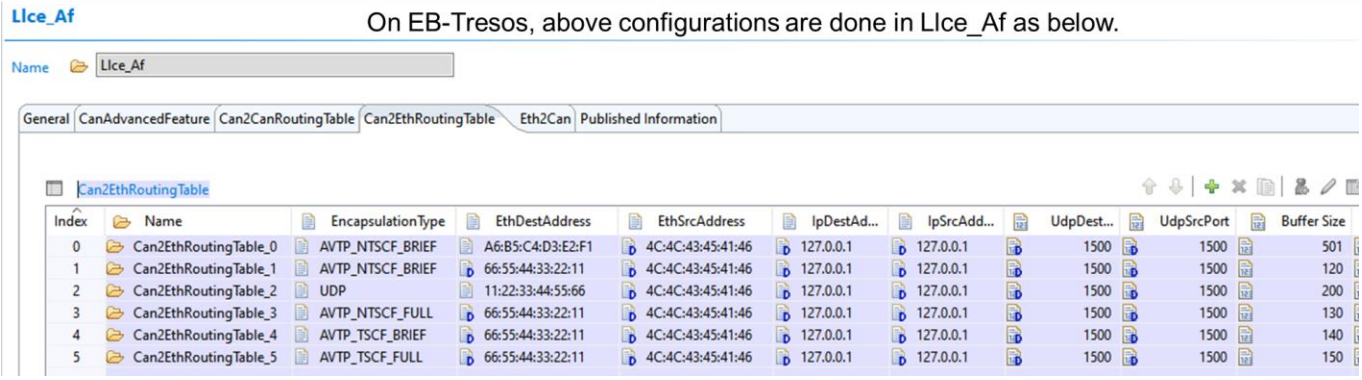


Figure 32. CAN frame encapsulation config

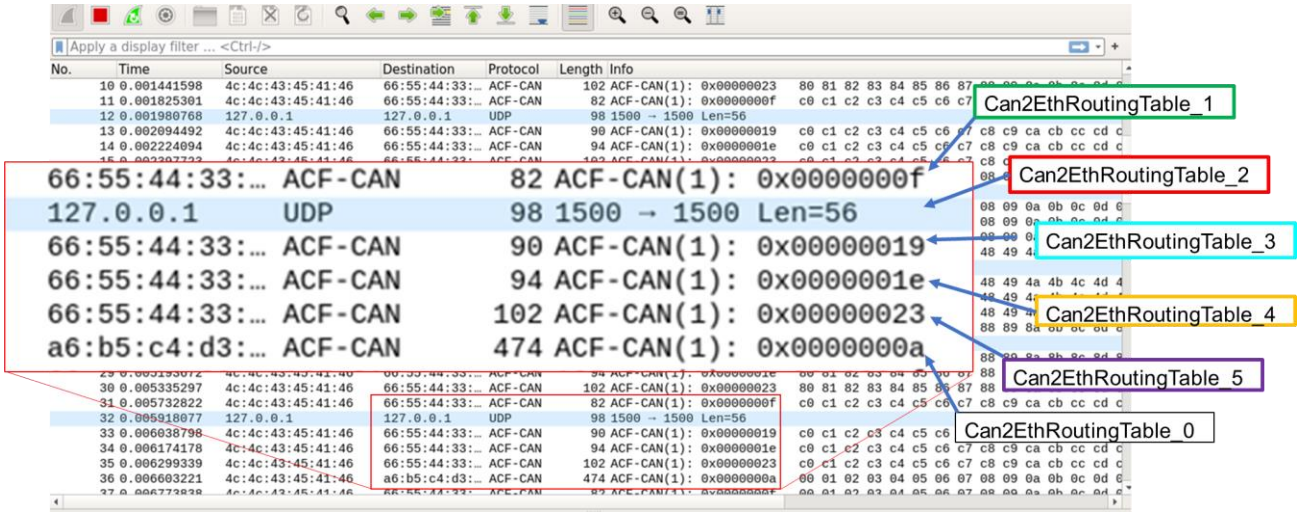


Figure 33. Capturing CAN2ETH frames on Wireshark

For ETH2CAN, you can play it with CAN2ETH setup as is. Running the same elf file as CAN2ETH (use same cmm also), LLCE performs ETH2CAN. If you will simply send back the CAN2ETH UDP packet to S32G, you can play the UDP ETH2CAN example easily.

At first, export PCAP based on the captured CAN2ETH UDP packet (sent via Can2EthRoutingTable_2). As below, on Wireshark, select the CAN2ETH UDP packet and File-Export Specified Packets. This creates PCAP to send back to S32G.

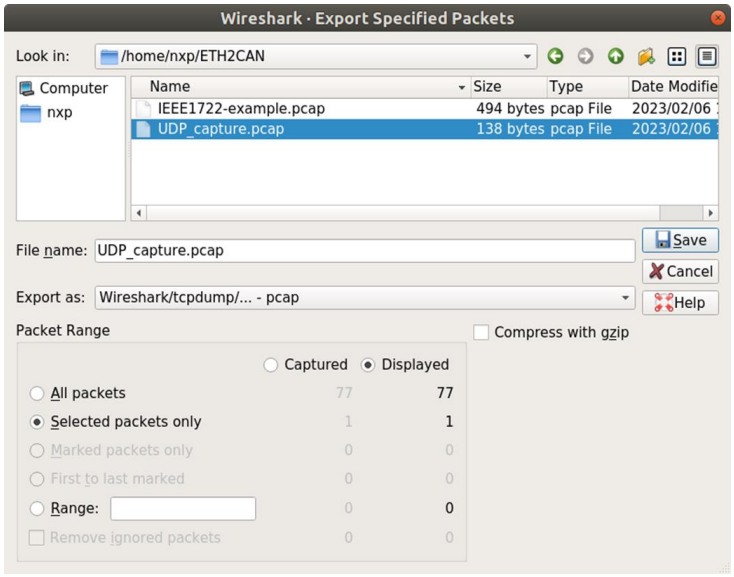


Figure 34. Export PCAP of captured CAN2ETH UDP

Then, send back the UDP packet to S32G using the exported PCAP. For example, you can send the packet using tcpreplay as below.

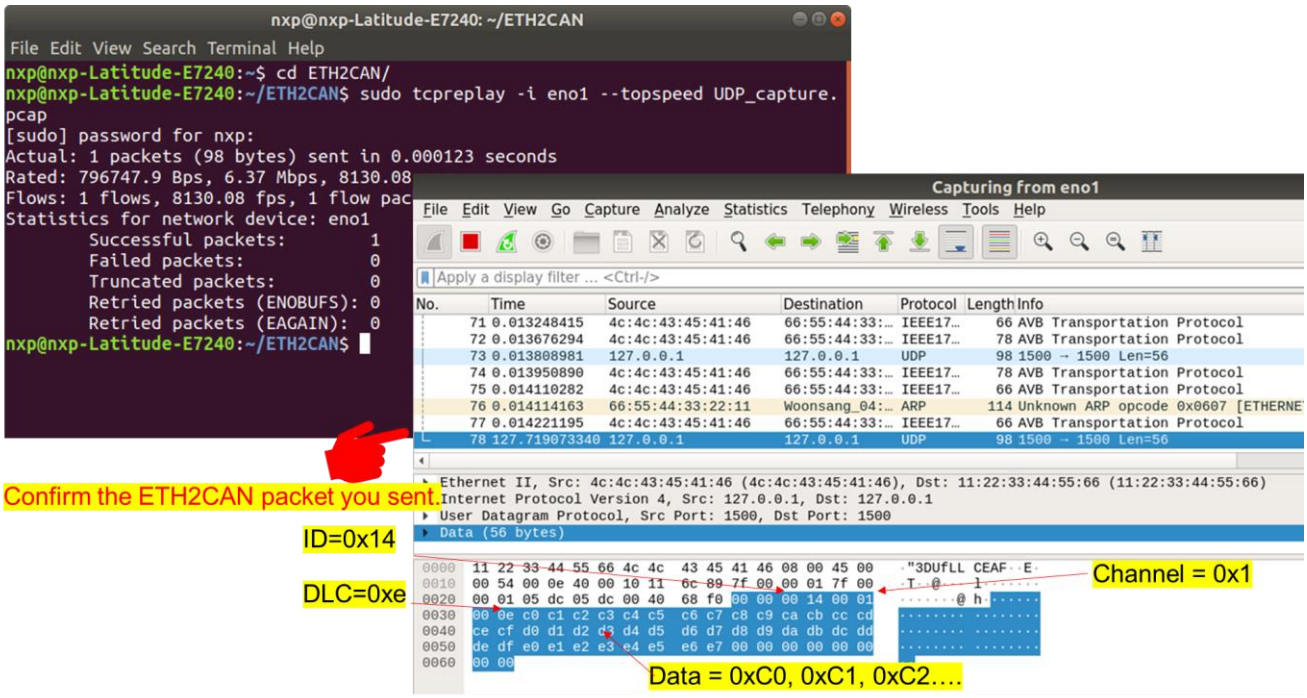


Figure 35. Send back UDP CAN2ETH packet to S32G

Then, you will see the CAN frame on LLCE_CAN1 unpacked from the ETH2CAN UDP packet.

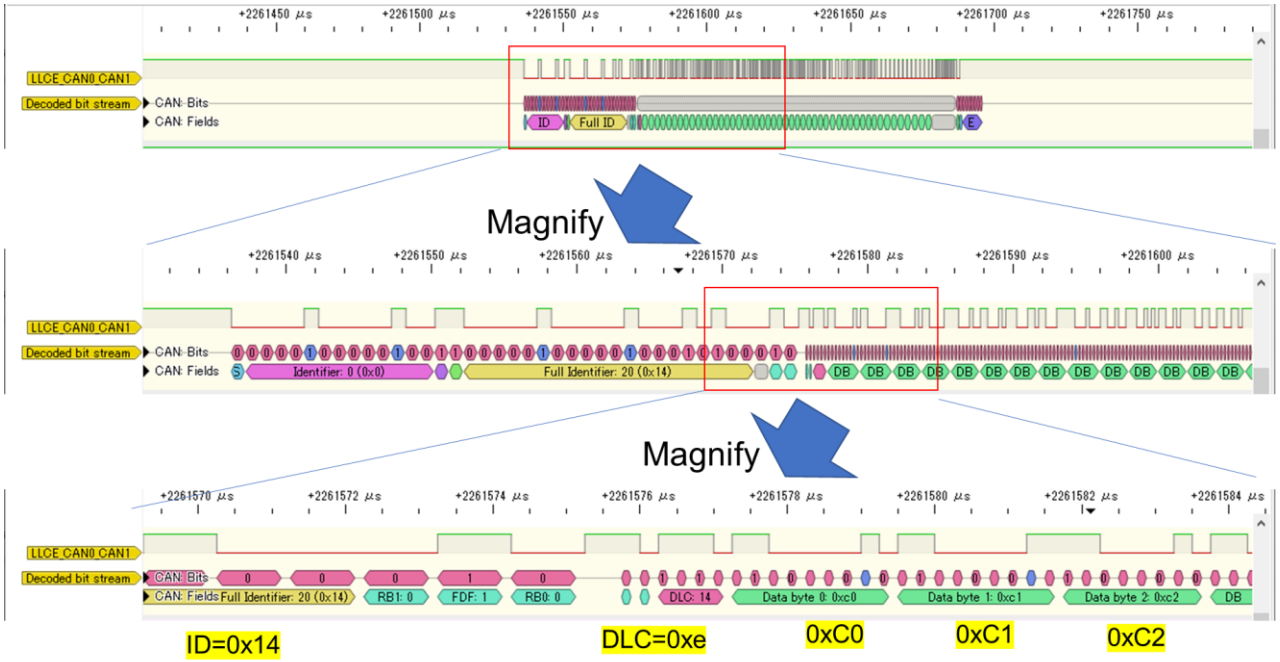


Figure 36. ETH2CAN unpacked CAN frame

As for IEEE1722 ETH2CAN example, if you connect multiple CAN channels, you can see more routed CAN frames.

If you send the packet generated from bundled PCAP “IEEE1722-example.pcap” to PFE_MAC1, LLCE parses it and unpacks the encapsulated CAN frames to each destination according to the ACF CAN msg information embedded in the packet (i.e. all odd CAN channels). If you connect all odd CAN channels to the companion CAN channels (e.g. even channels), you can observe all unpacked CAN frames from the IEEE1722 Ethernet frame.

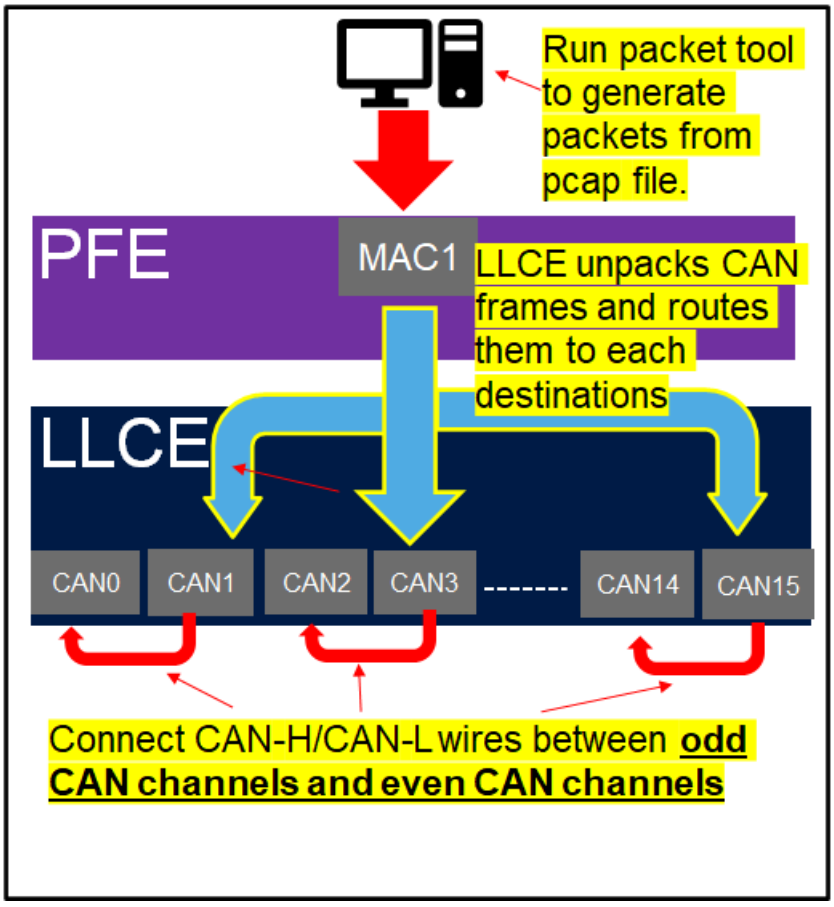


Figure 37. IEEE1722 ETH2CAN routing example

If you capture the odd CAN buses with Logic Analyzer, you can see the routed CAN frames.

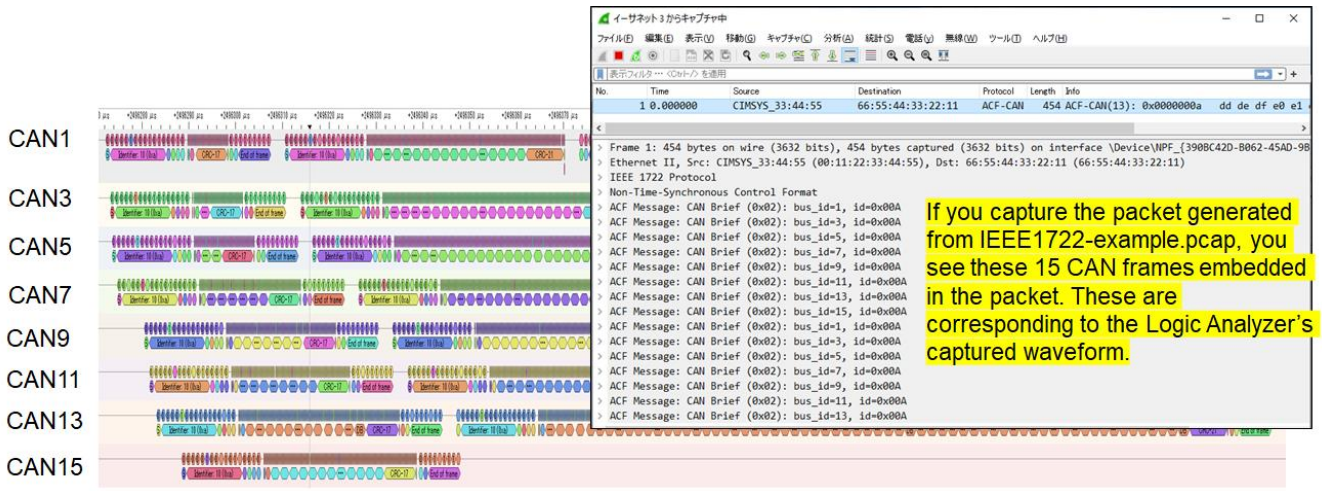


Figure 38. Routed CAN frames

4. Configurating on EB Tresos

This section explains how to configure essential items for customization of CAN2CAN/CAN2ETH/ETH2CAN. Import the EB-Tresos configuration delivered in the sample app as a template, then customize it.

NOTE

This section is based on the sample app config latest release as of February 2023. (i.e. S32G_LLCE_GATEWAY_1.0.5_QLP1_D2302.exe).

4.1. Importing the sample config

Run EB-Tresos Studio and import sample config.

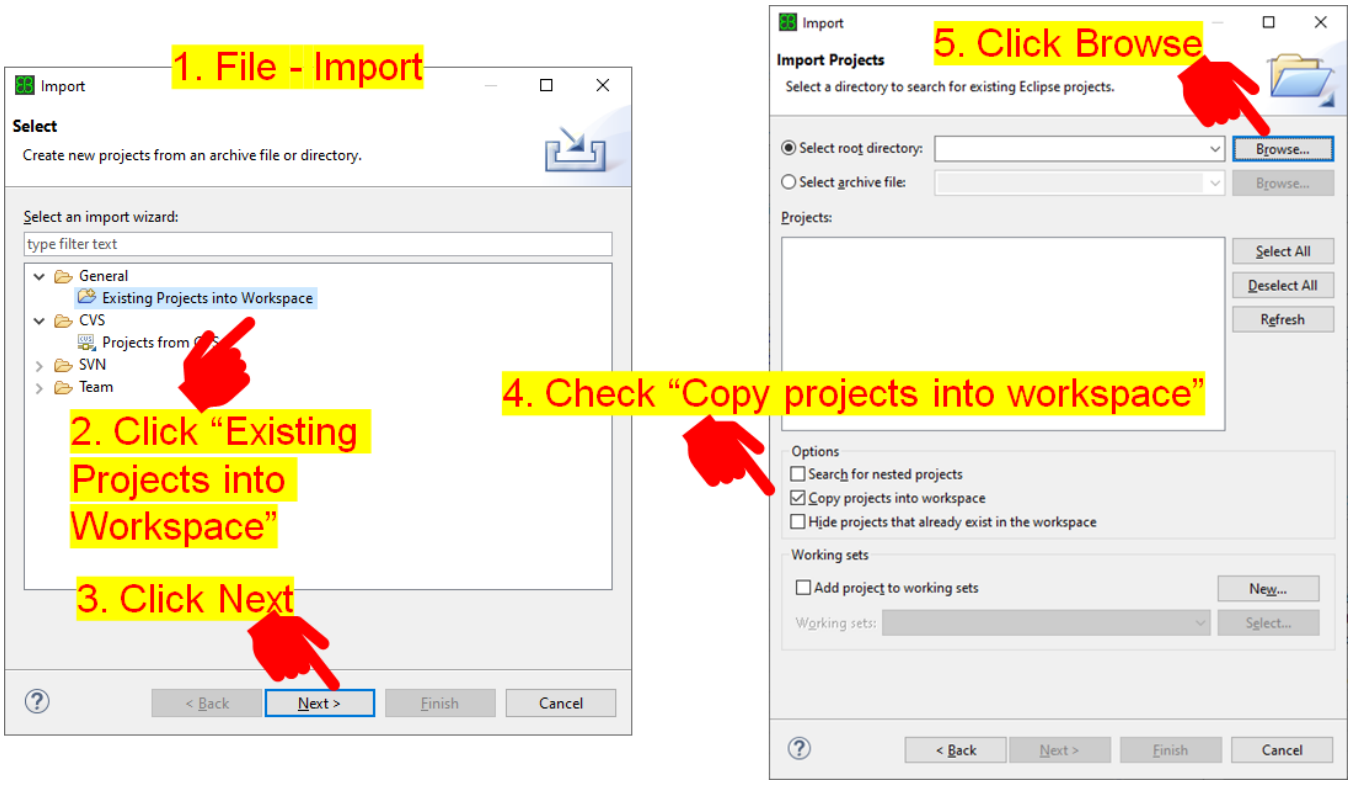


Figure 39. Importing file

In case of CAN2CAN, import the following file. Choose "tresos_s32g3" or "tresos_s32g2" according to your target, and then, select "Tresos_CAN2CAN_Project".

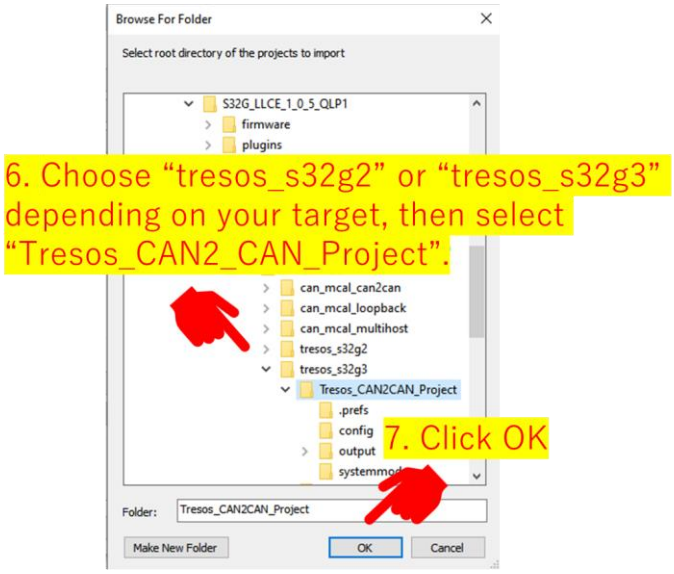


Figure 40. Importing CAN2CAN config file

In case of CAN2ETH and ETH2CAN, choose "tresos_S32G2" or "tresos_S32G3" depending on your target.

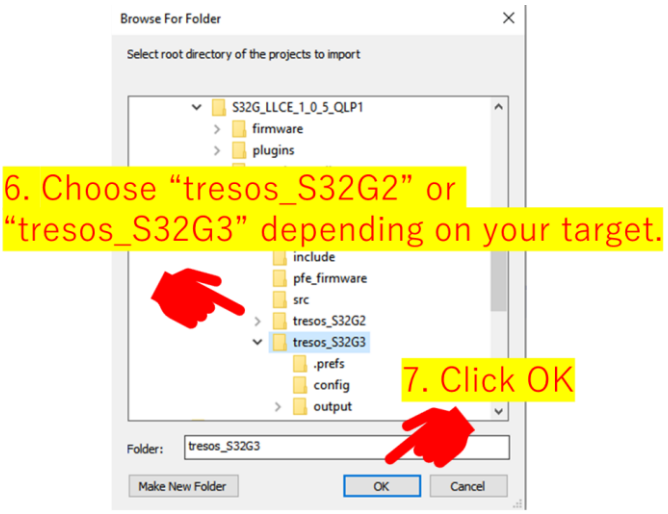


Figure 41. Importing CAN2ETH and ETH2CAN config file

To import sample configuration, browse and select the root directory, click on finish. Refer to the following screenshot.

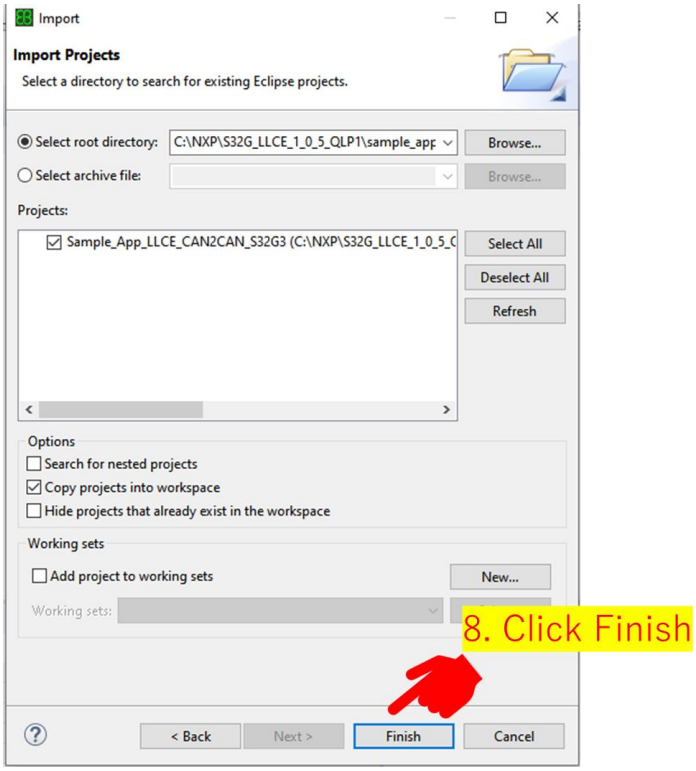


Figure 42. Importing sample configuration

To rename the imported project config right click and select Rename. Enter the new name in the dialog box and click OK.

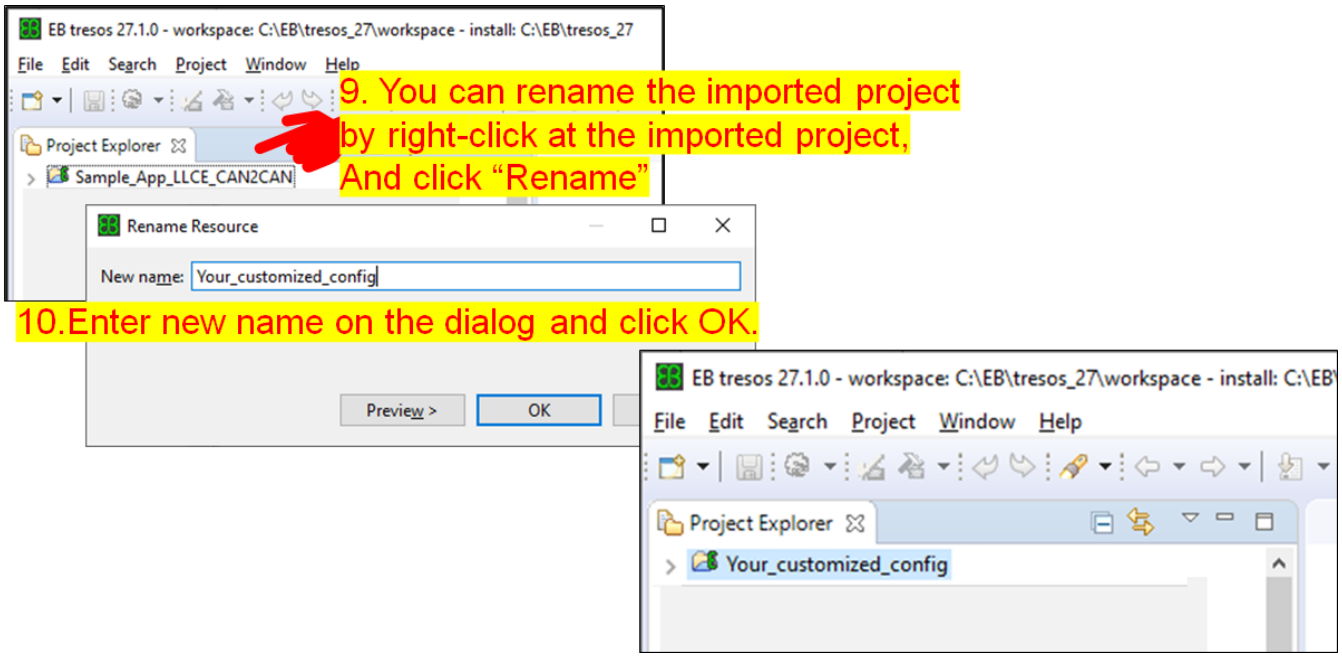


Figure 43. Renaming the imported project

Every time you start configuration on the Tresos studio, you need to load config.

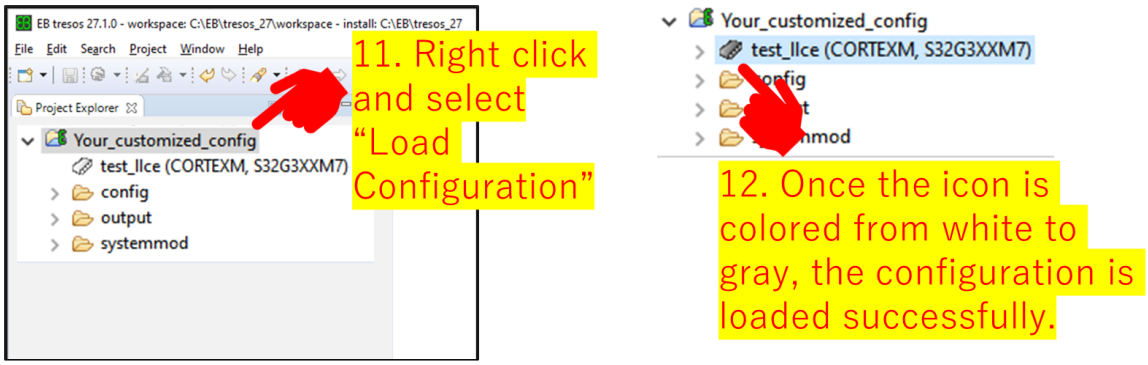


Figure 44. Restarting and selecting the config

4.2. Configure Llce_Af for CAN2CAN

In Configure Can2CanRoutingTable follow the steps to configure Llce_Af.

1. Double click Llce_Af.
2. Select Can2CanRoutingTable Tab.
3. You can add/delete these for your CAN2CAN use case. In order to configure routing details, double click the entry index.

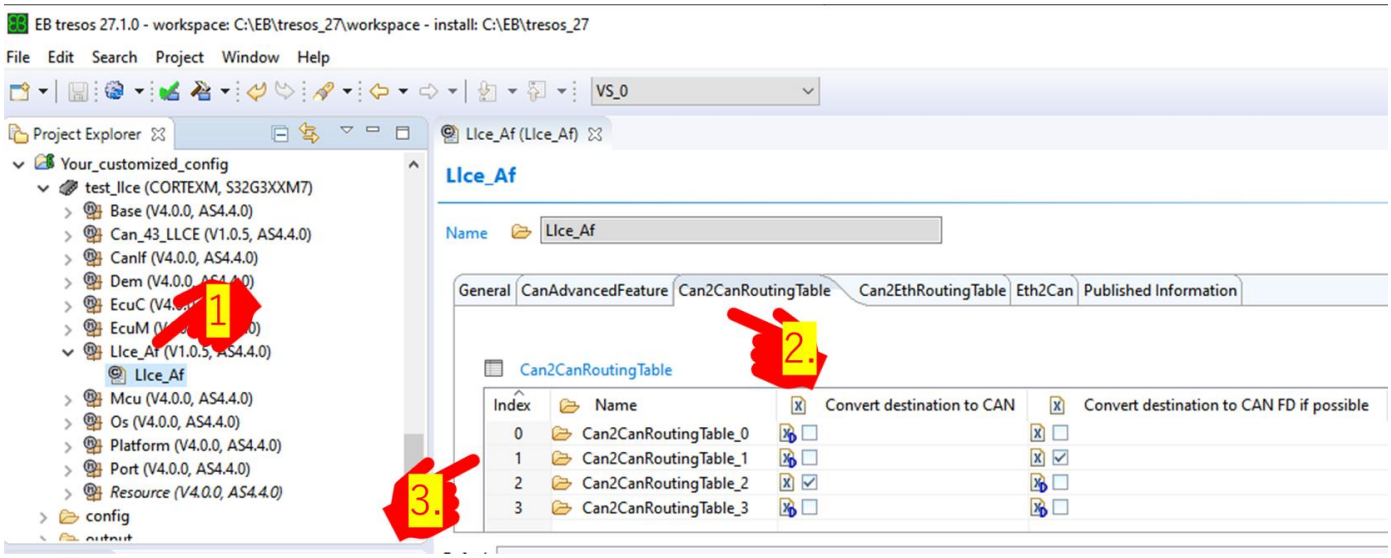


Figure 45. Configuring Llce_Af

In CanDestinationList, configure routing destinations.

1. Select CanDestinationList
2. Select the destination channel from the pull-down list. If the desired channel is missing in the list, you should add it on the Can_43_Llce/CanController (as explained in the section “Configure CanController”)
3. You can add/delete entries for the destination. Now there is only 1 destination in this list hence this routing is unicast. If you add destination, the corresponding routing will be multicast routing.

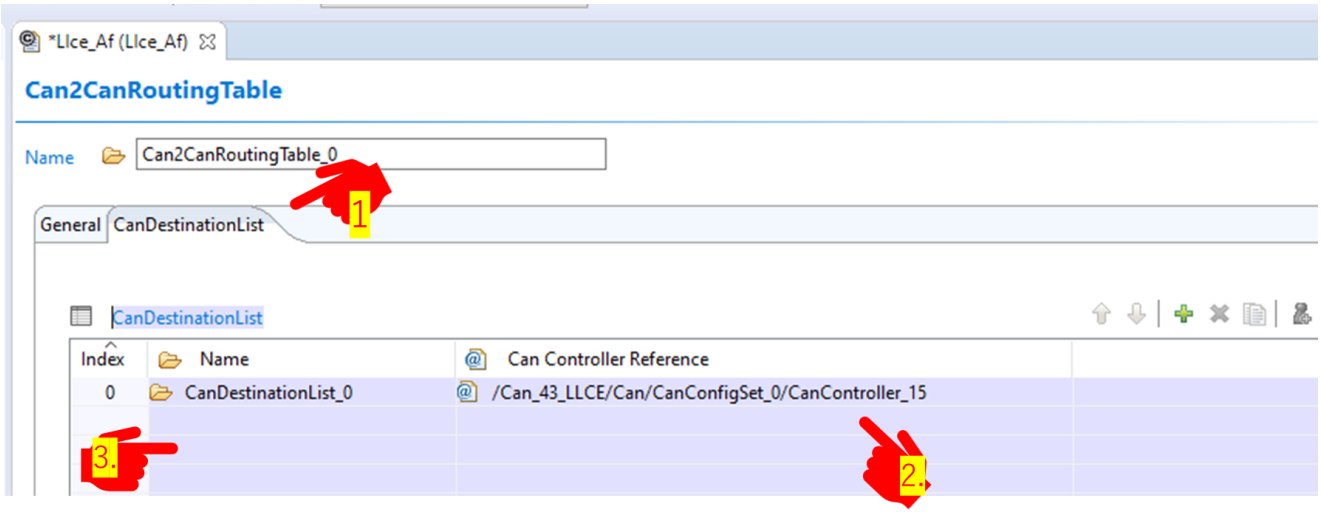


Figure 46. Configuring Destination

Configure routing details in General Tab.

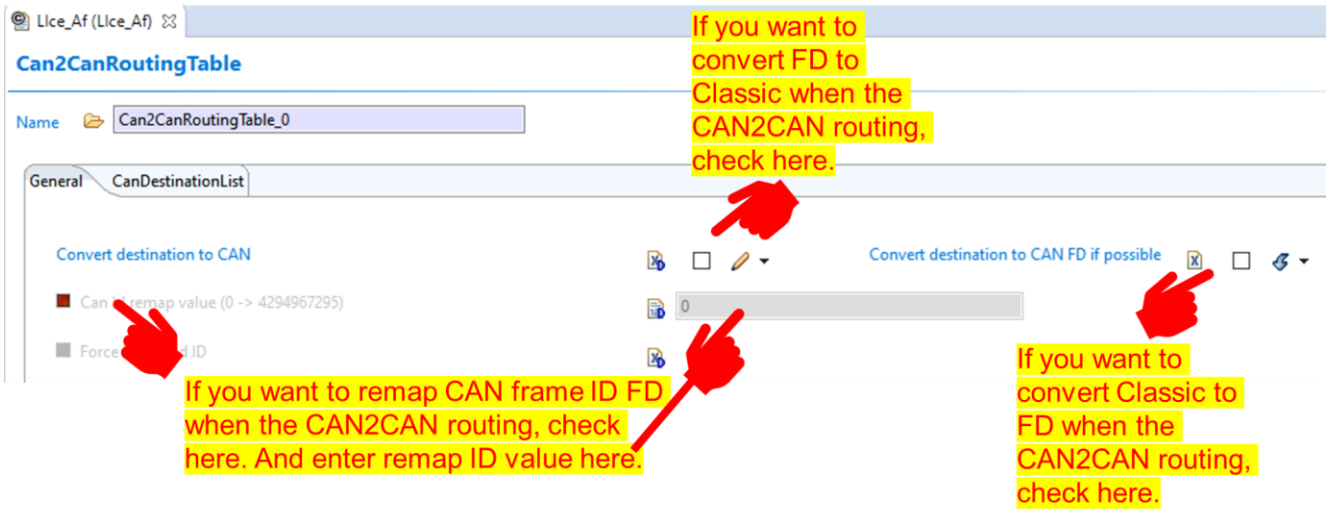


Figure 47. Configuring the General tab

You should ensure that the configured Can2CanRouting is referred from CanAdvancedFeature table. Follow the steps given below.

1. Click home icon.
2. Select CanAdvancedFeature.
3. You can add/delete these entries. Note these entries are referred from Hardware Receive Handle, which will be configured in Can_43_LLCE/CanHardwareObject.
4. Select the routing table from the pull-down list.

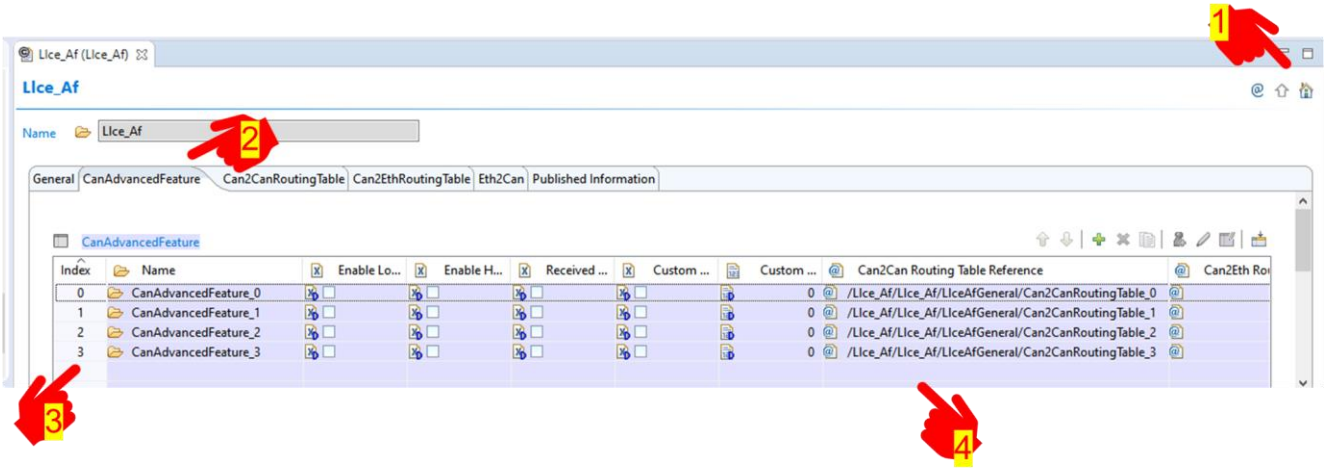


Figure 48. CAN advanced feature table

4.3. Configuring Llce_Af for CAN2ETH

In Configure Can2EthRoutingTable follow the steps to configure Llce_Af.

1. Double click Llce_Af.
2. Select Can2EthRouting Table Tab.
3. You can add/delete these for your CAN2ETH use case. In order to configure routing details, double click the entry index.

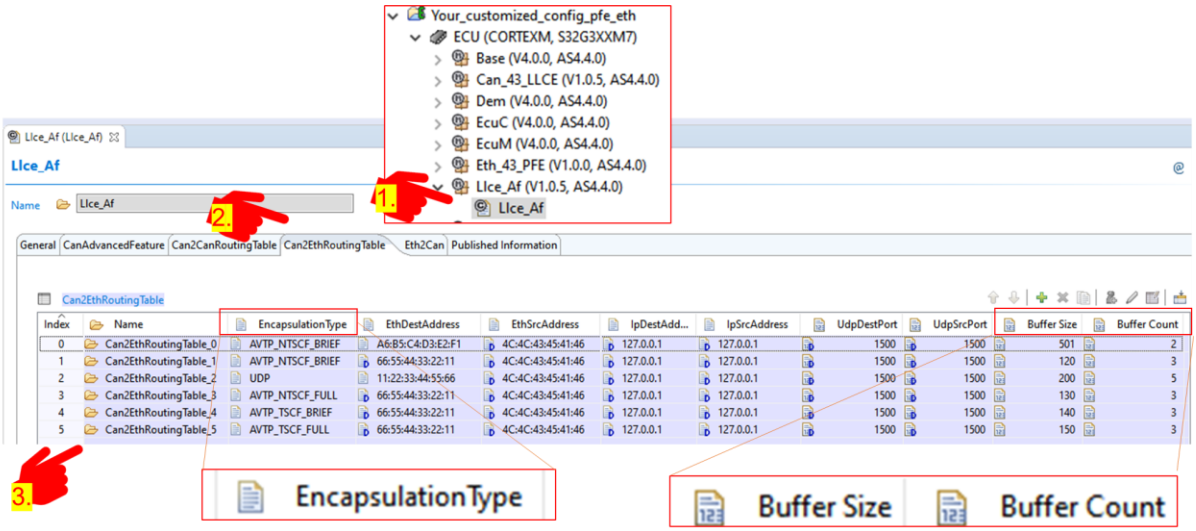


Figure 49. Configuring Llce_Af

As for the EncapsulationType, you can choice as following.

AVTP_NTSCF_BRIEF : IEEE1722 ACF_CAN_BRIEF on the Non- Time-Synchronous Control Format

AVTP_NTSCF_FULL: IEEE1722 ACF_CAN on the Non-Time-Synchronous Control Format

AVTP_TSCF_BRIEF : IEEE1722 ACF_CAN_BRIEF on the Time-Synchronous Control Format

AVTP_TSCF_FULL: IEEE1722 ACF_CAN on the Time-Synchronous Control Format

As for Buffer Size, if you want to pack ***N*** ACF msg in one IEEE1722 CAN2ETH packet, use the following formula to calculate the buffer size.

In case of the ACF_CAN_BRIEF on the Non- Time-Synchronous Control Format, the Buffer size should be equal or larger than

26+(N-1)*(8+can_msg_payload) -1 + 72

and less than

26+N*(8+can_msg_payload) -1+72

In case of the ACF_CAN on the Non-Time-Synchronous Control Format, the Buffer size should be equal or larger than

26+(N-1)*(16+can_msg_payload) -1 + 80

and less than

26+N*(16+can_msg_payload) -1+80

In case of the ACF_CAN_BRIEF on the Time-Synchronous Control Format, the Buffer size should be equal or larger than

40+(N-1)*(8+can_msg_payload) -1 + 72

and less than

40+N*(8+can_msg_payload) -1+72

In case of the ACF_CAN on the Time-Synchronous Control Format, the Buffer size should be equal or larger than

40+(N-1)*(16+can_msg_payload) -1 + 80

and less than

40+N*(16+can_msg_payload) -1+80

NOTE

"can_msg_payload" is the term of Abbreviated CAN/CAN FD message for IEEE-1722 ACF message. It should be 0 – 16 quadlets.

For example, if you want to pack 10 ACF msg / packet (DLC=1) in ACF_CAN_BRIEF on the Non-Time-Synchronous Control Format, the Buffer size should be equal or larger than 205 (i.e. 26+9*(8+4) -1 + 72) and less than 217 (i.e. 26+10*(8+4) -1+72).

The buffer count depends on a multitude of factors. It is not that easy to calculate exact values without some experimentation.

- There might be a risk data will be over-written when more Can frames arrive before the Eth frame is sent
- Multiple input buses

4.4. Configuring Can controller

In the following example config, BCAN0,1,14 and 15 are configured in default. Follow the steps to add BCAN.

1. Double click Can43_LLCE.
2. Select CanController tab.
3. Select CanController_15 for example.
4. Click Duplicate icon.

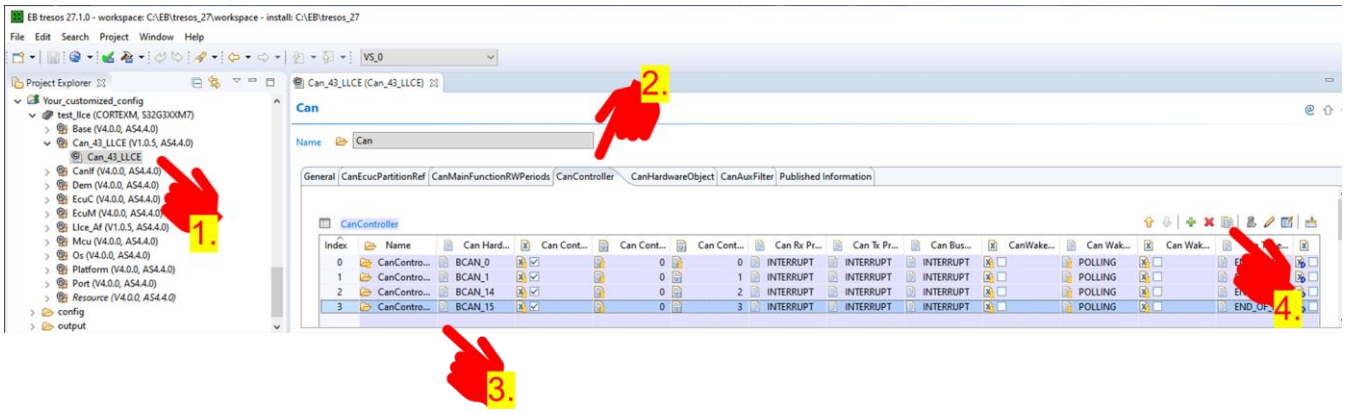


Figure 50. Configuring BCAN (one)

5. Select BCAN at column “Can Hardware Channel”.
6. Set sequential number at column “Can Controller ID” (4 in this case.).
7. Double click the index column of the added element.

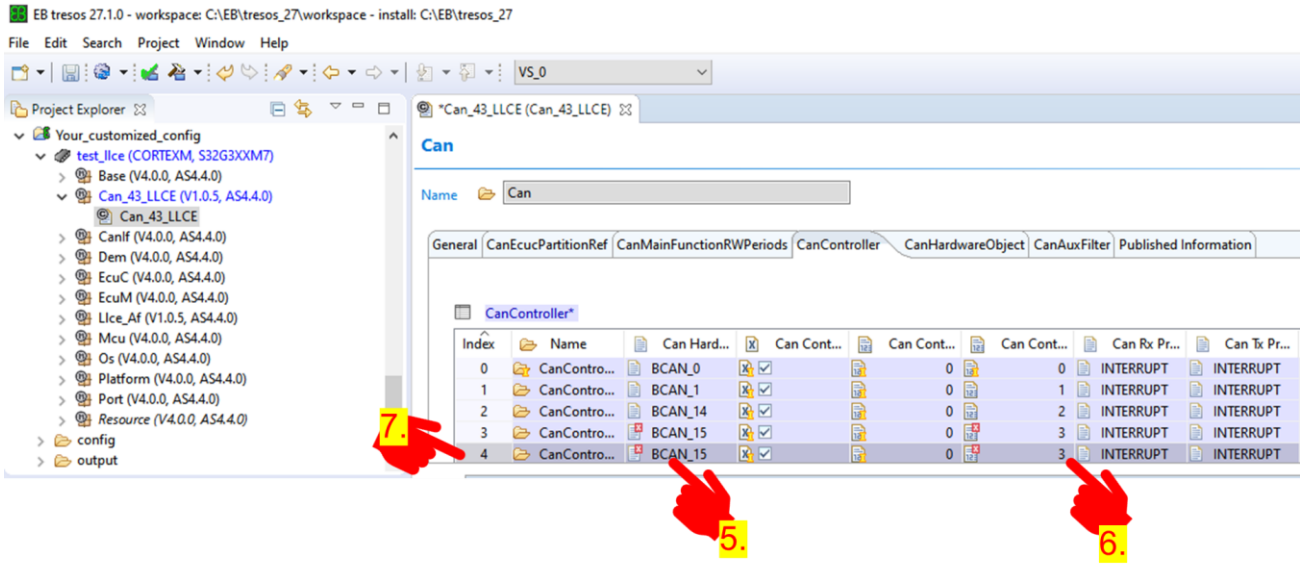


Figure 51. Configuring BCAN (two)

8. Select CanControllerBaudrateConfig Tab.
9. Double click the index column of any of these. (In this explanation, choice index 0).

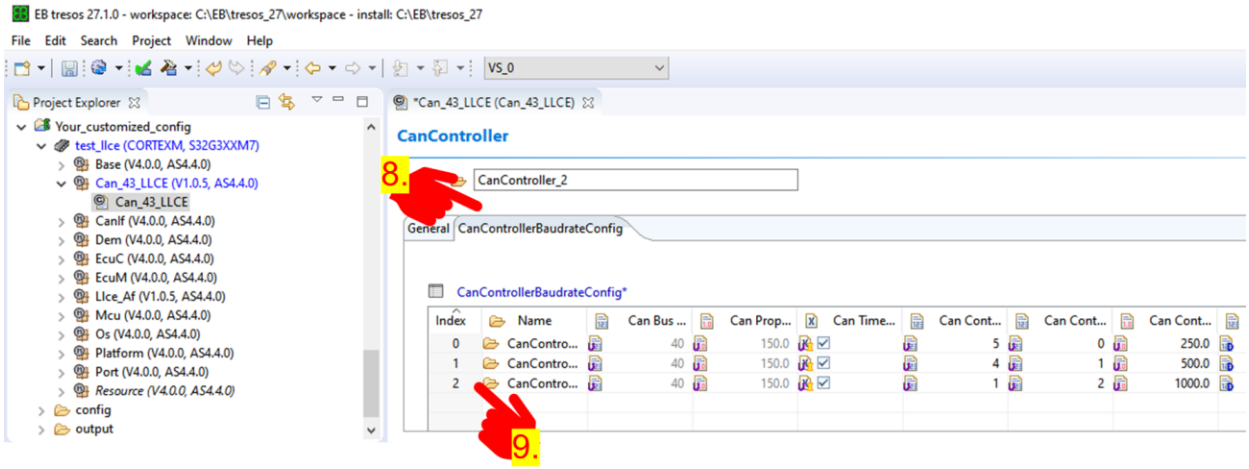


Figure 52. Configuring BCAN (three)

10. Configure baud rate parameters.

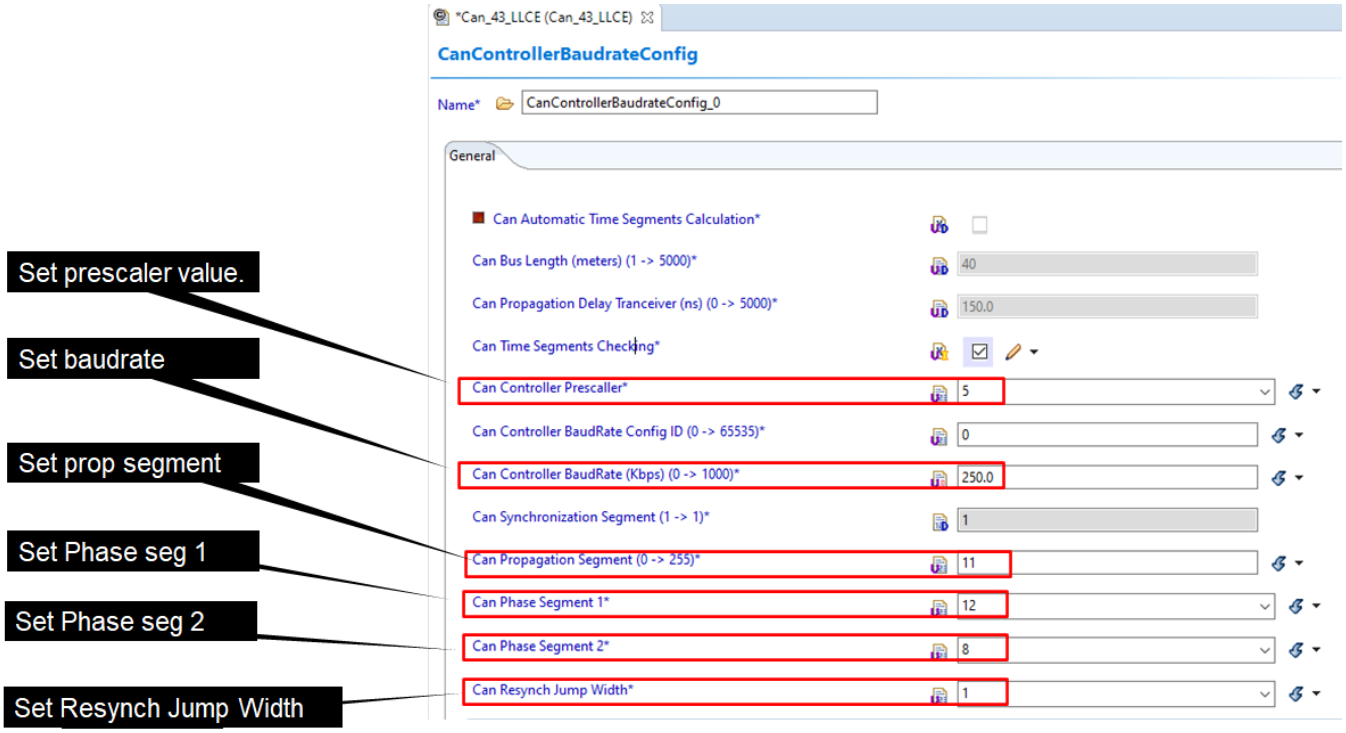


Figure 53. Baud rate setting

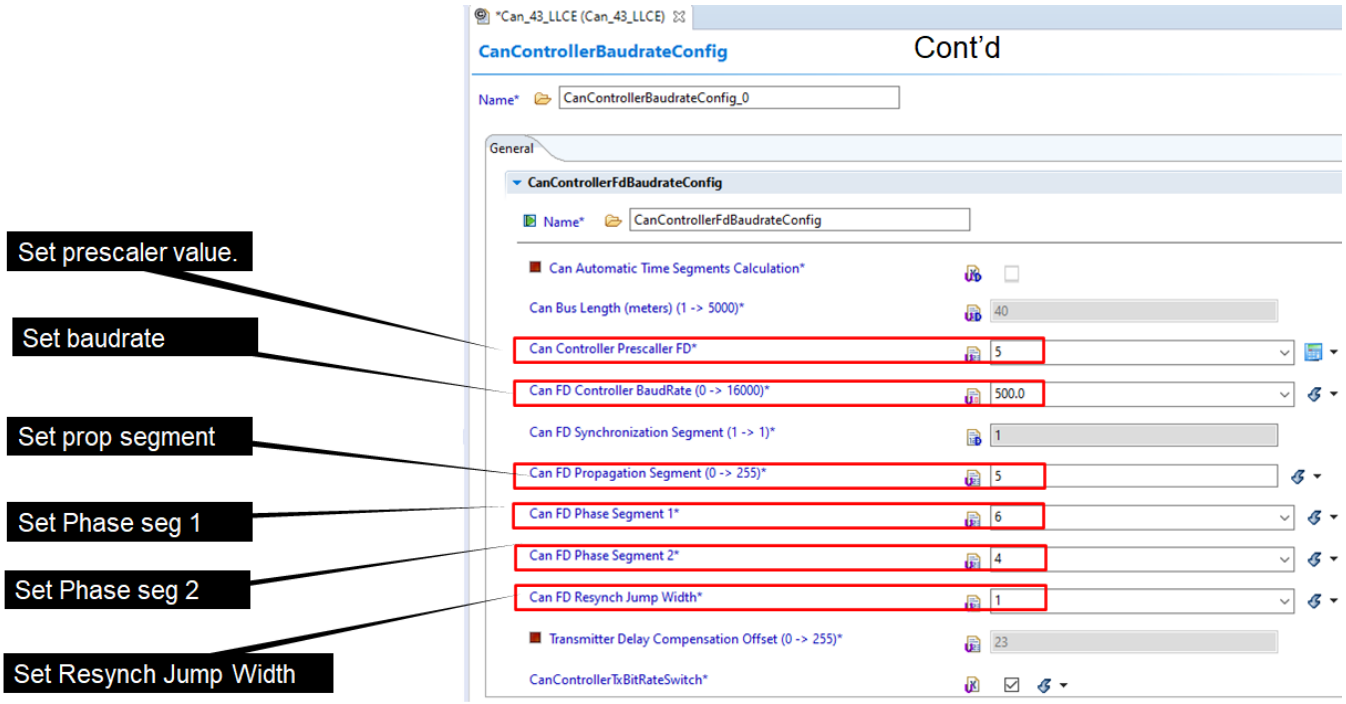
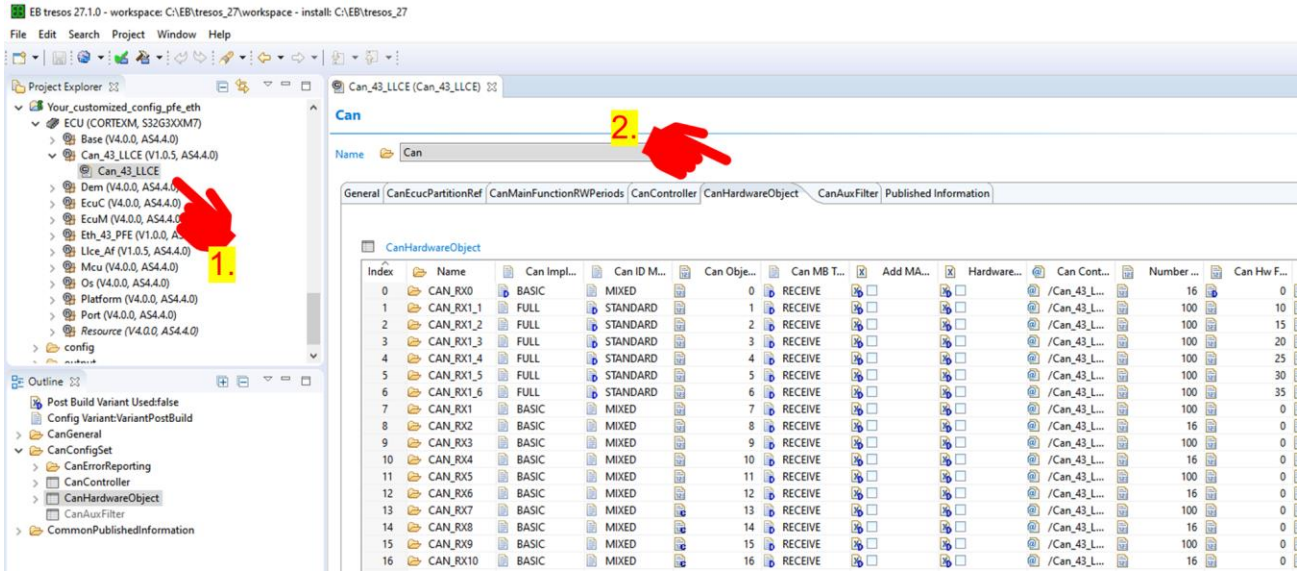


Figure 54. Data phase baud rate setting

4.5. Configure Can hardware object

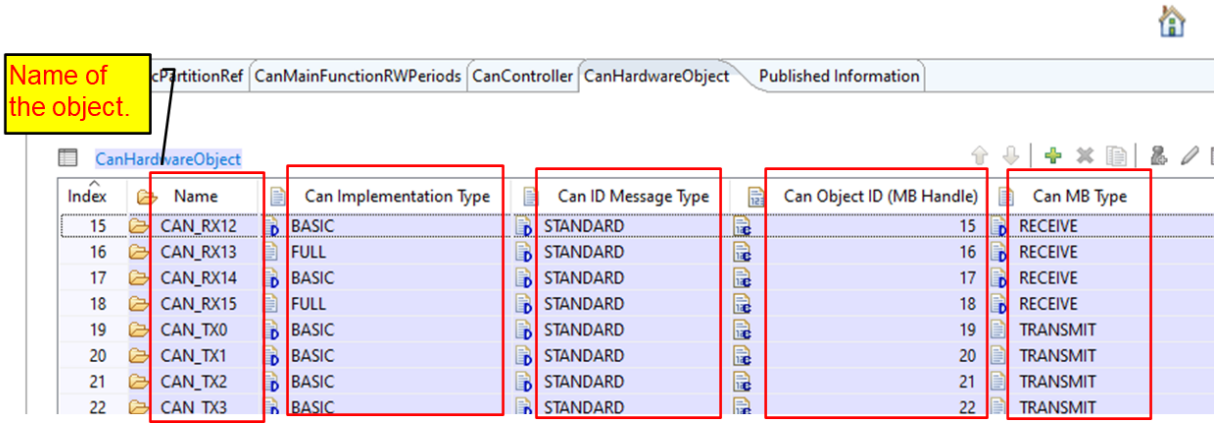
Follow the steps to configure message buffer related settings.

1. Double click Can43_LLCE.
2. Select CanHardwareObject tab.

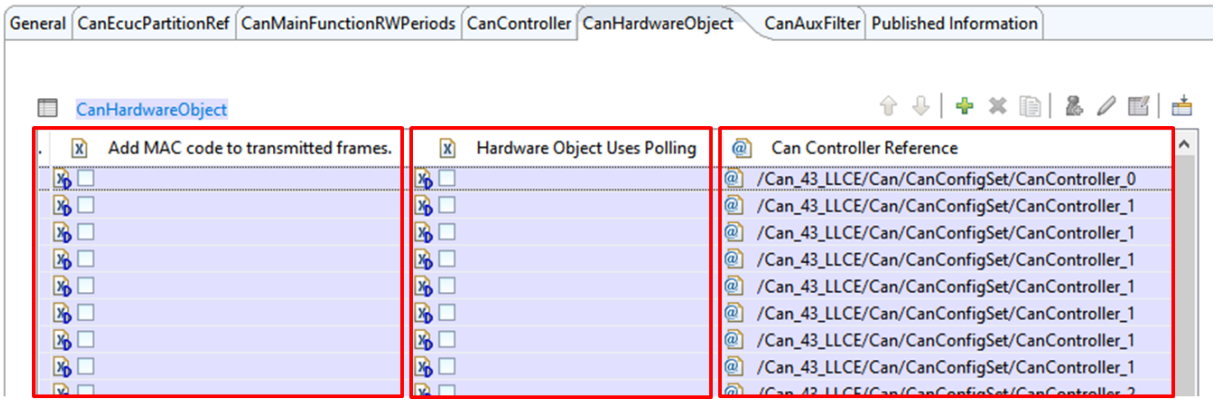


3. Select ID mask BASIC : ID mask enabled. FULL: Exact ID match.
4. Select CAN frame ID type STANDARD / EXTENDED.

5. Object Handle ID. Should start with 0 and continue without any gaps.
6. Select MB Type. RX or TX.

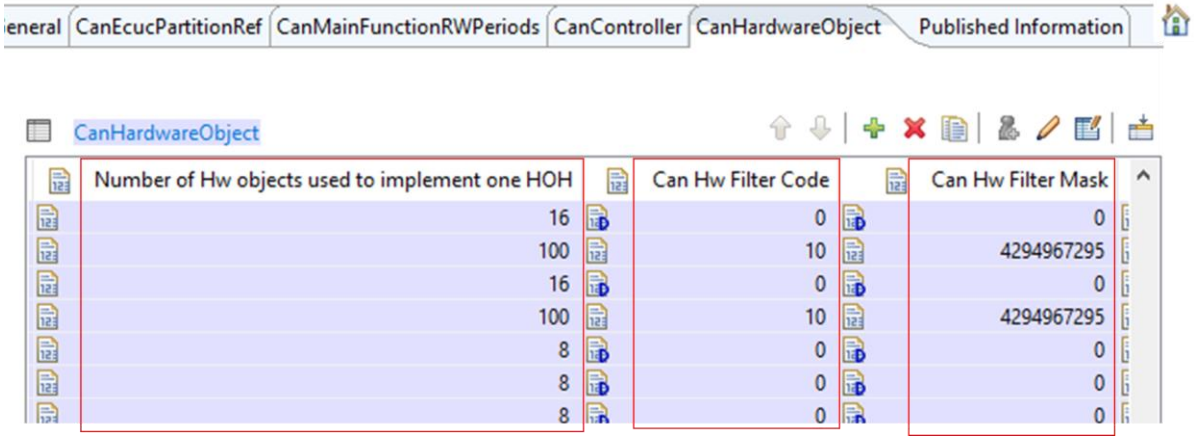


7. MAC feature: Not available for standard enablement FW.
8. Enables polling of the object.
9. Specify which CanController has the object

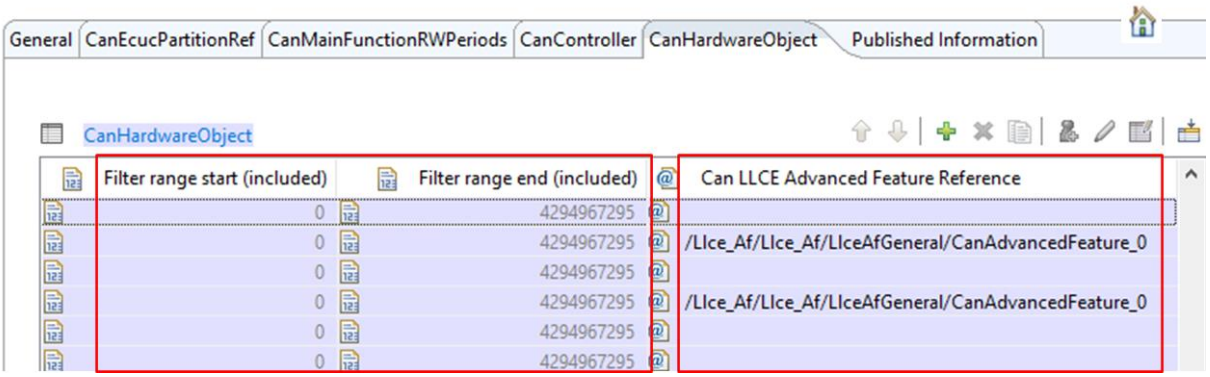


10. Number of hardware objects used to implement the object handle. It means that the number of message buffers which are assigned to the object handle.
11. Specify (together with the filter mask) the frame ID that passes the hardware filter for the RX object.

12. Specify (together with the Filter Code) the range that passes the hardware filter for the RX object.



13. Specify that this filter is of range type. This over-rides the information in the standard CanHwFilter. If enabled, the filter will accept IDs from RangeStart to RangeEnd.
14. Specify which CanAdvancedFeature is used for the RX object. The host should take care of the RX objects which do not have any reference here.



5. Configuring on S32CT

This section explains how to configure essential items on S32CT for customization of CAN2CAN. After installing RTD and the LLCE complex driver, you can open CAN2CAN sample app project on S32DS which has same behavior as this document already described in previous sections. This section guides how to build and play it. It then describes how to config it with S32CT instead of EB Tresos.

NOTE

This section is based on the sample app config of the latest release as of February 2023. (i.e. S32G_LLCE_GATEWAY_1.0.5_QLP1_D2302.exe).

5.1. Installing S32DS 3.5, RTD and LLCE drivers

The following four software packages needs to be downloaded and installed.

- S32 Design Studio v3.5 installer
- S32 Design Studio 3.5.1 development packages for offline use, support for S32G
- S32G Real Time Drivers Version 4.0.0 Update Site
- S32G_LLCE_GATEWAY_1.0.5_QLP1_D2302

Go Flexera, download the S32DS3.5 installer and install it.

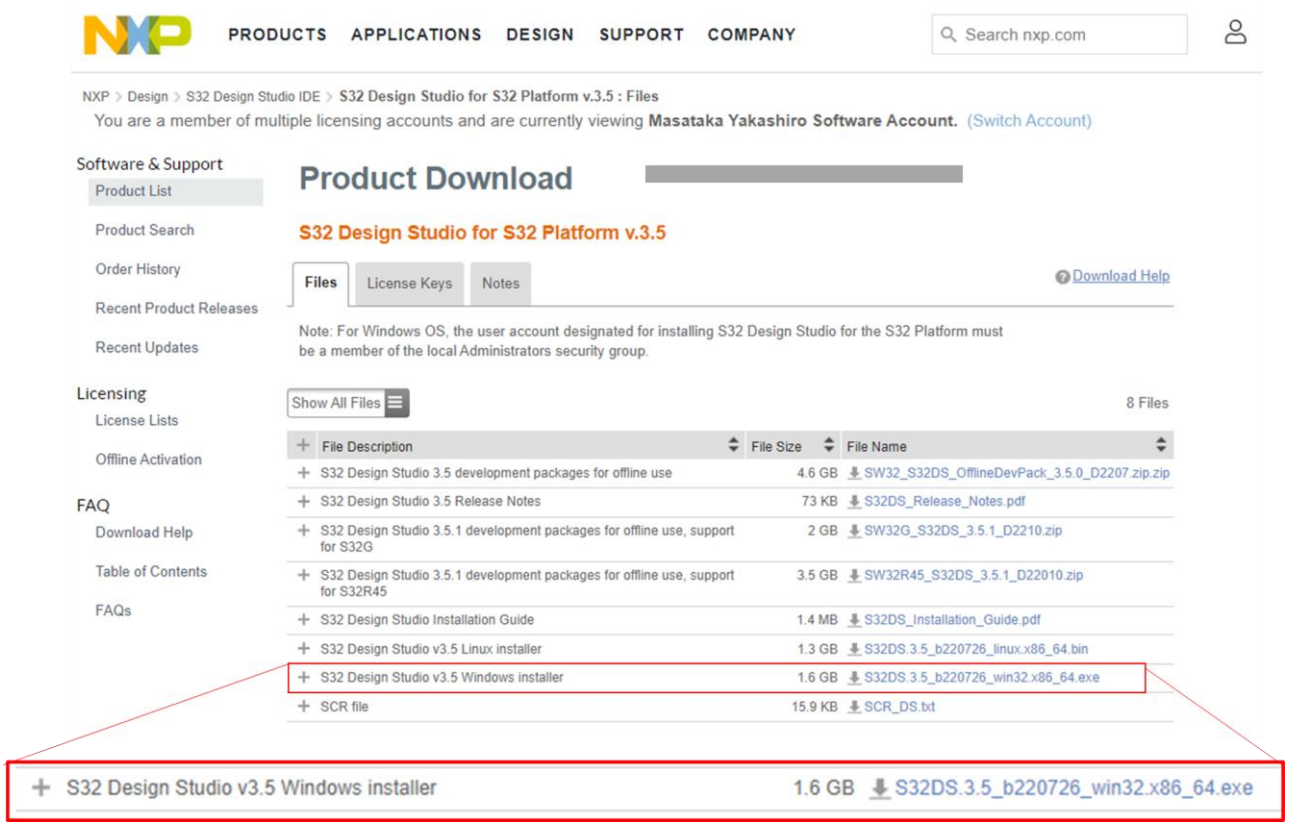


Figure 55. Downloading S32DS3.4

Download the S32 Design Studio 3.5.1 development package for S32G family.

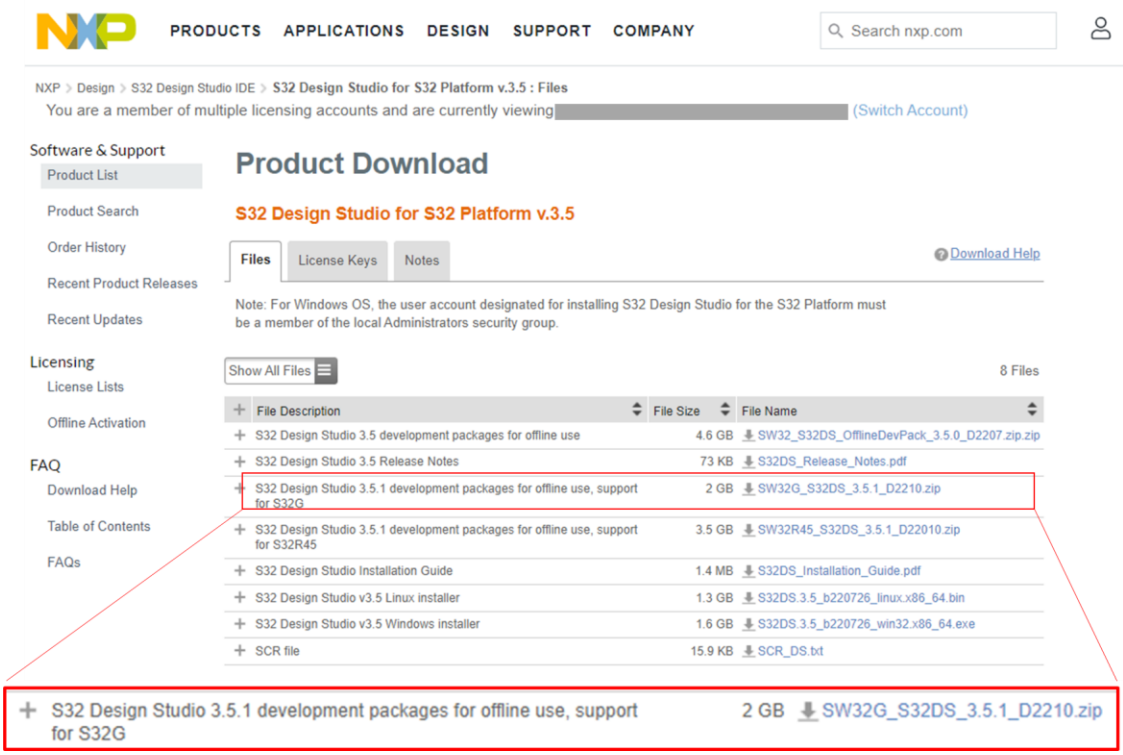


Figure 56. Update for support of S32G2 family

Download the S32G Real Time Drivers Version 4.0.0 Update Site .

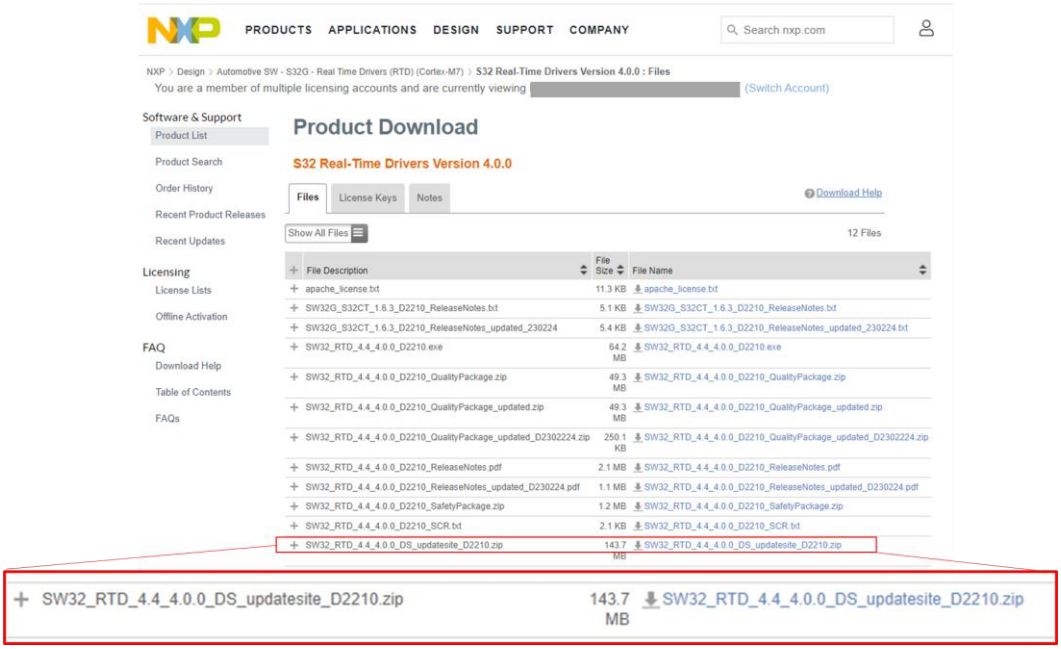


Figure 57. Downloading S32G Real Time Drivers Version 4.0.0 Update Site

As for S32G_LLCE_GATEWAY_1.0.5_QLP1_D2302, assuming you already installed in your PC. The update site file is located under the installed folder.

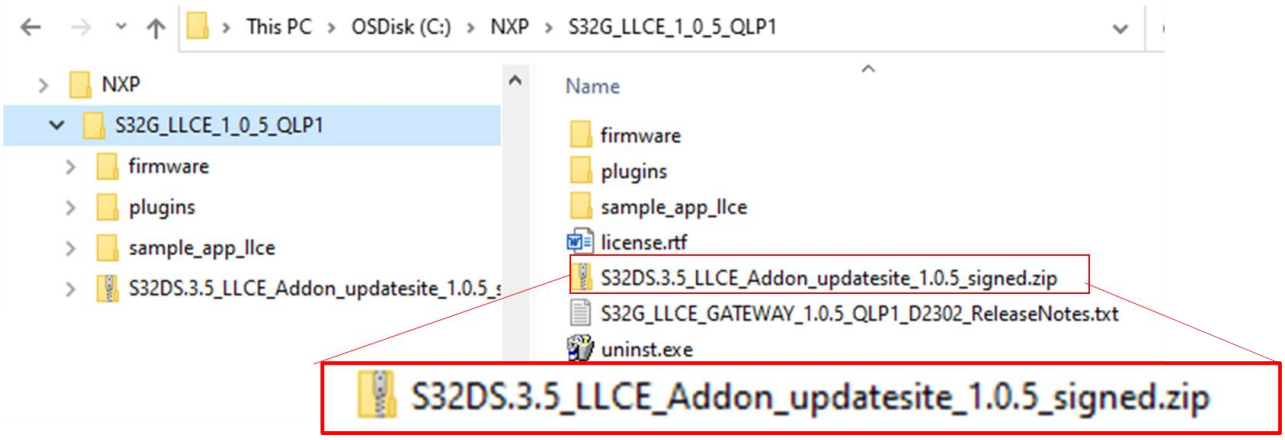
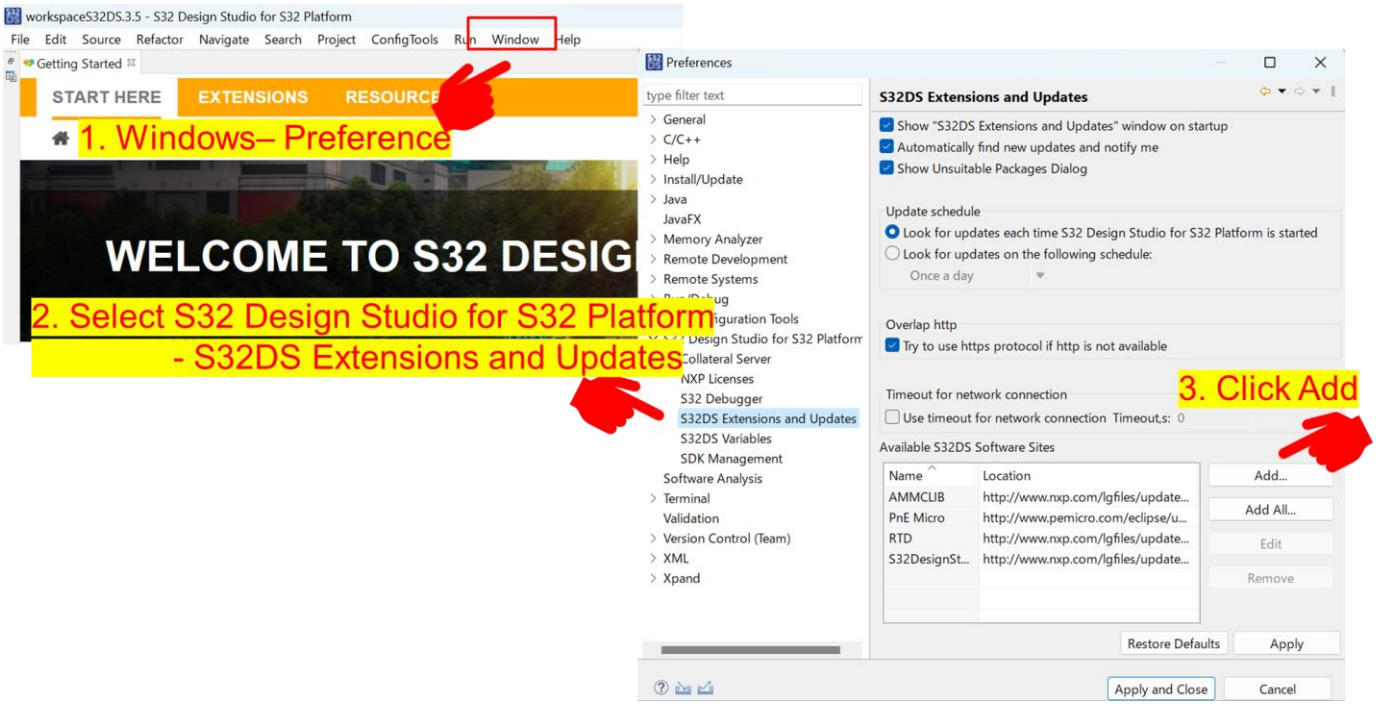


Figure 58. LLCE update site

After installing S32DS3.5, add the downloaded three zip files (S32DS3.5.1 Devlopment package for S32G, RTD4.0.0 updatesite and LLCE1.0.5_updatesite.) in the S32DS.



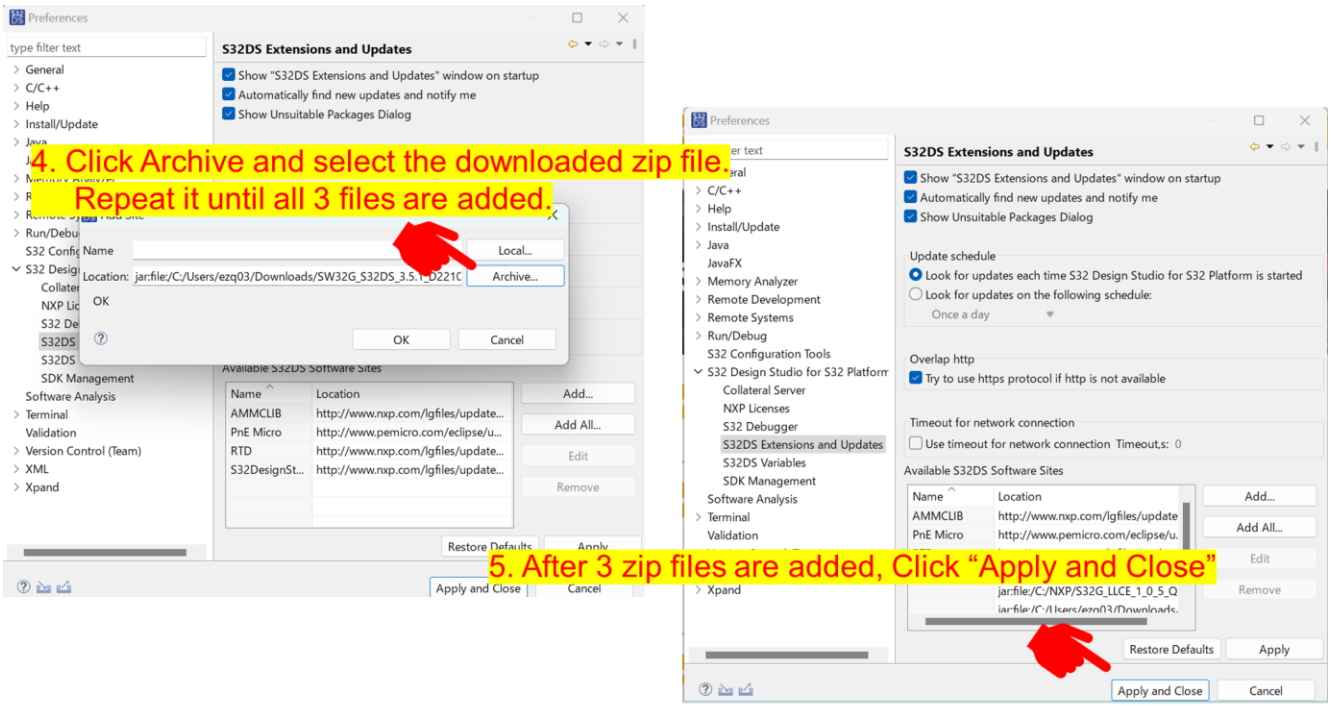


Figure 59. Adding the downloaded zip files

Installing the S32 Design Studio 3.5.1 with support for S32G family. Please follow the below steps.

1. Click on Help and select S32DS Extensions and Updates.
2. Select following extensions:
   - GCC 9.2 build 1649
   - GCC 10.2 build1728
   - Platform pkg.
   - Platform Tools pkg.
   - S32G Dev. Pkg.
3. Click “Install/Update 5 item(s)”.

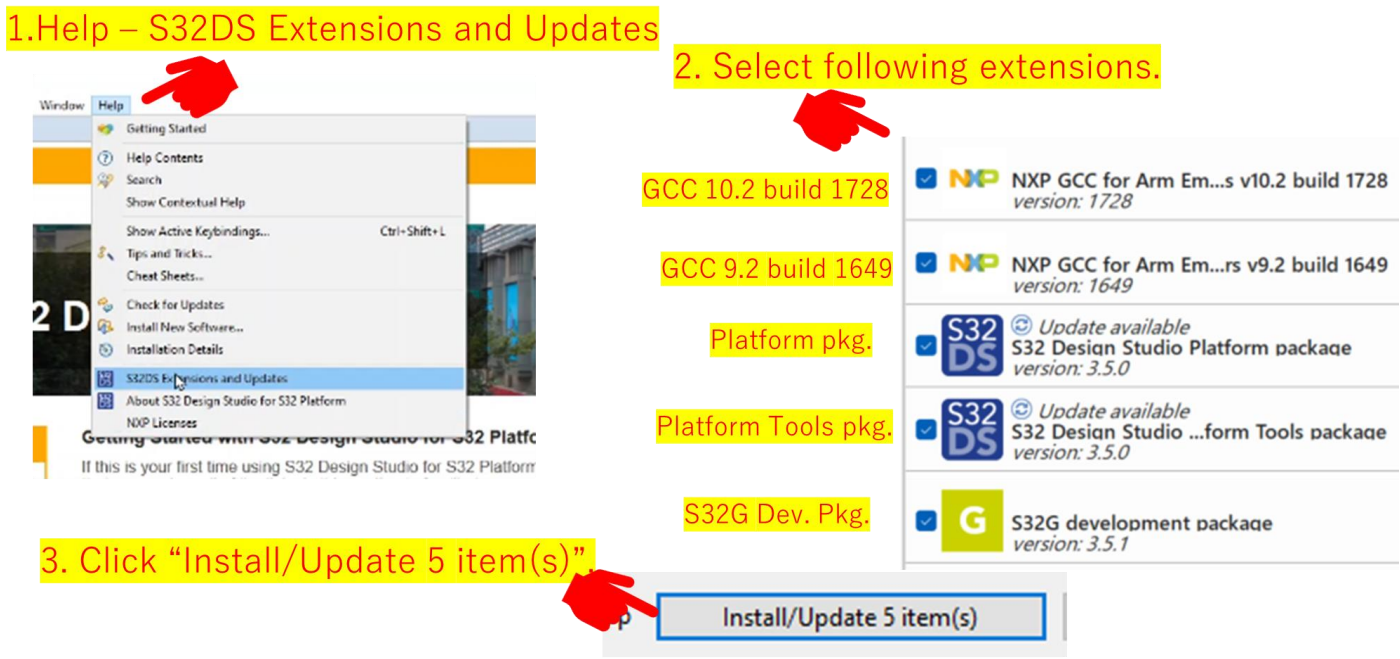


Figure 60. Steps to update support for S32G family

4. Click on Next and in the next window and select “I accept…”. Click finish to complete the installation. A pop up window appears to restart S32DS, click Yes.

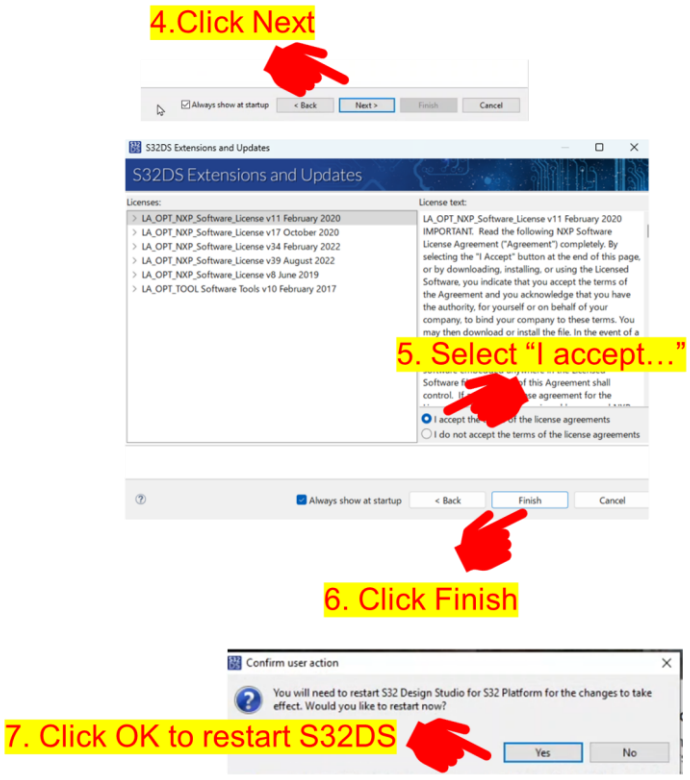


Figure 61. Steps to finish S32DS for S32G2 family

5.2. Installing LLCE driver and RTD on S32DS

To install LLCE driver follow these steps:

1. Select LLCE1.0.5, S32 RTD 4.0.0, S32G2 RTD4.0.0 and S32G3 RTD 4.0.0 on “S32DS Extensions and Updates” window.
2. Click “Install/Update 4 item(s)” and click Next.
3. Select “I accept…” and click on Finish. A pop up window appears to restart S32DS, click Yes.

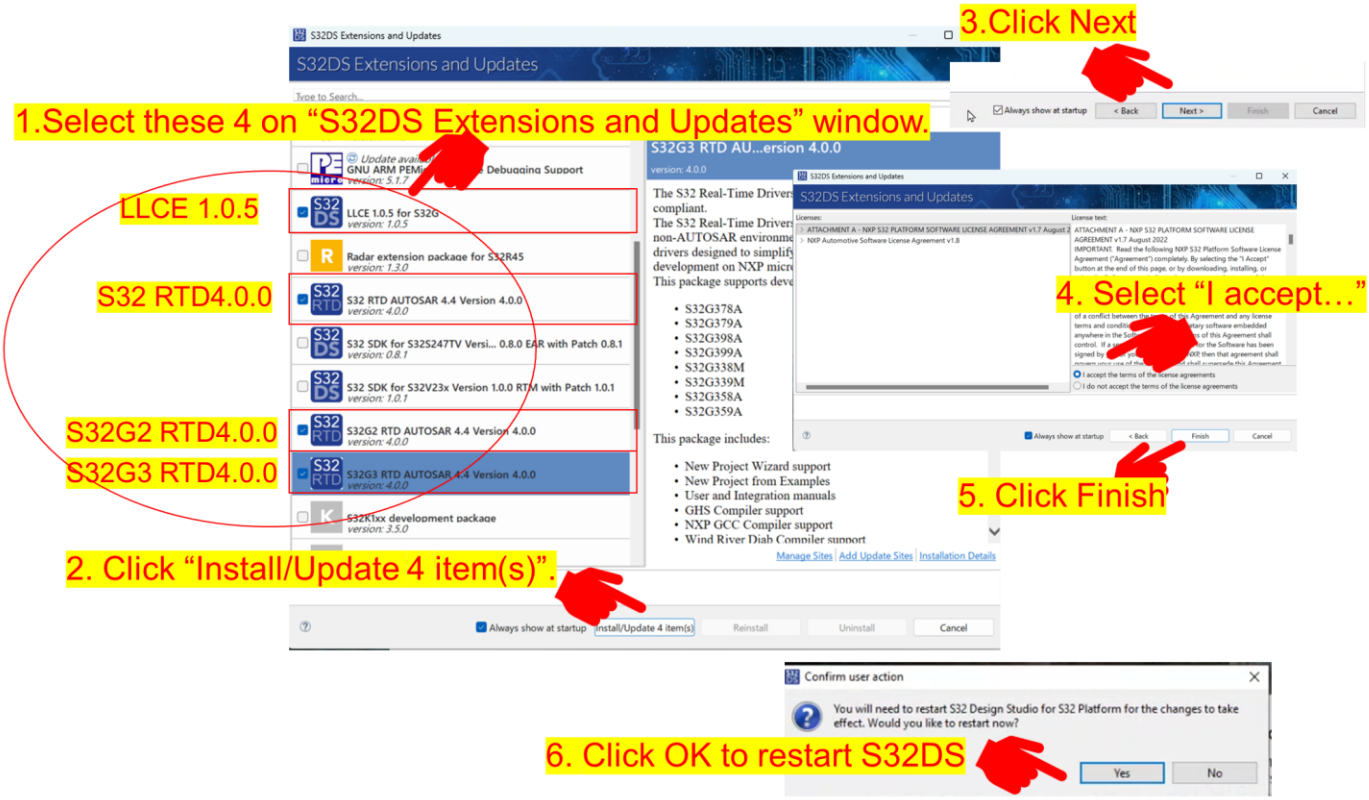


Figure 62. Steps to install LLCE driver and RTD

You will see pop-up window to trust certificates. Then accept it as below.

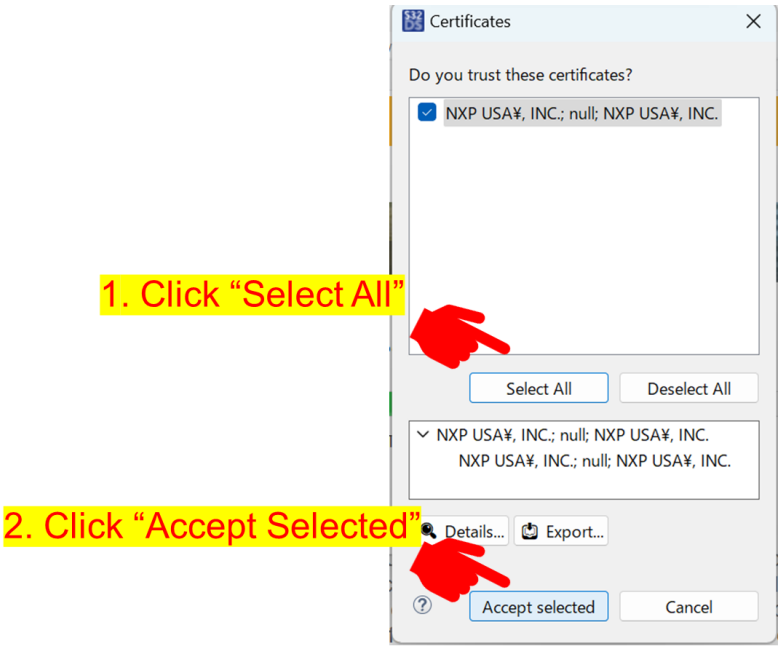


Figure 63. Trust certificate

6. CAN2CAN sample app creation

The following steps show how to create a new project.

1. Click on File, select New → select S32DS Project from Example.

2. Select Can_Llce_DS_Can2Can and click on Finish.

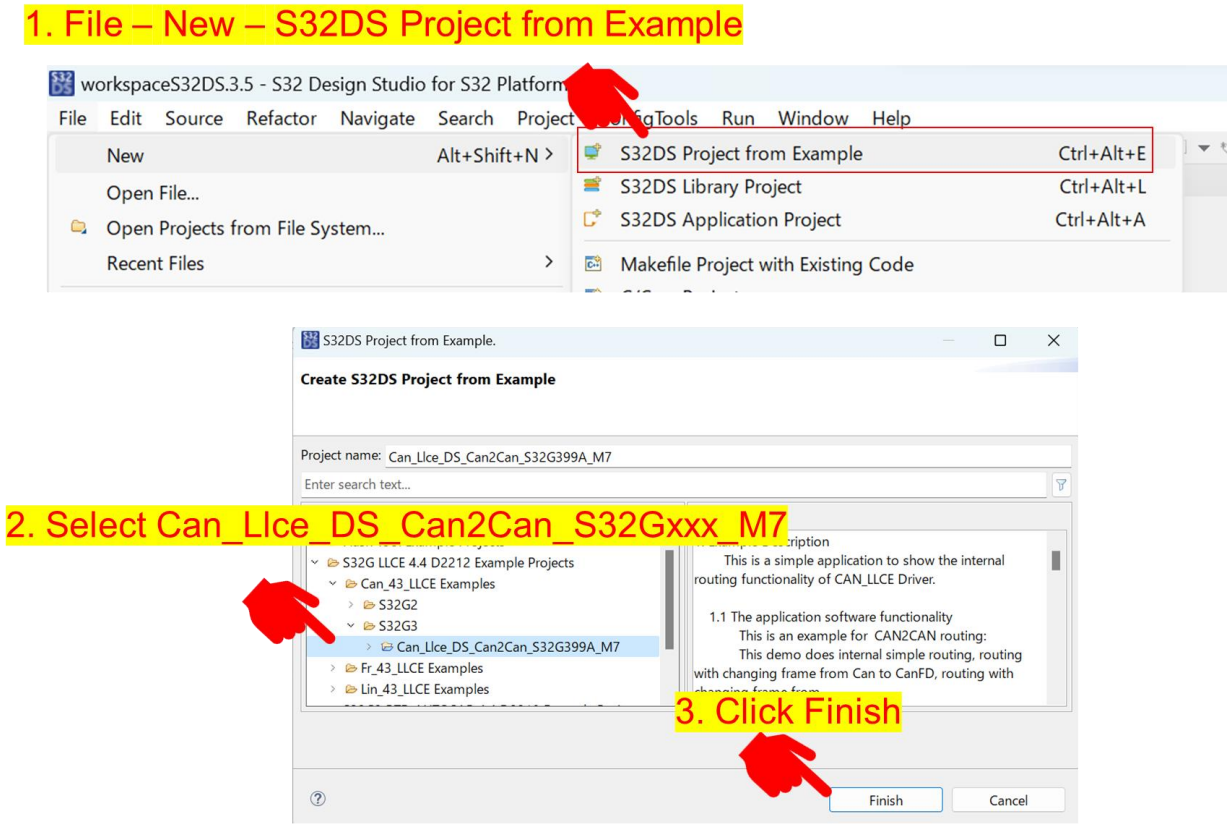


Figure 64. Starting a new project

3. Switch to S32CT Peripheral view and you can see the project of LLCE CAN2CAN sample app which is identical to the one which is already explained in this document. Click on ConfigTools and select Peripherals.

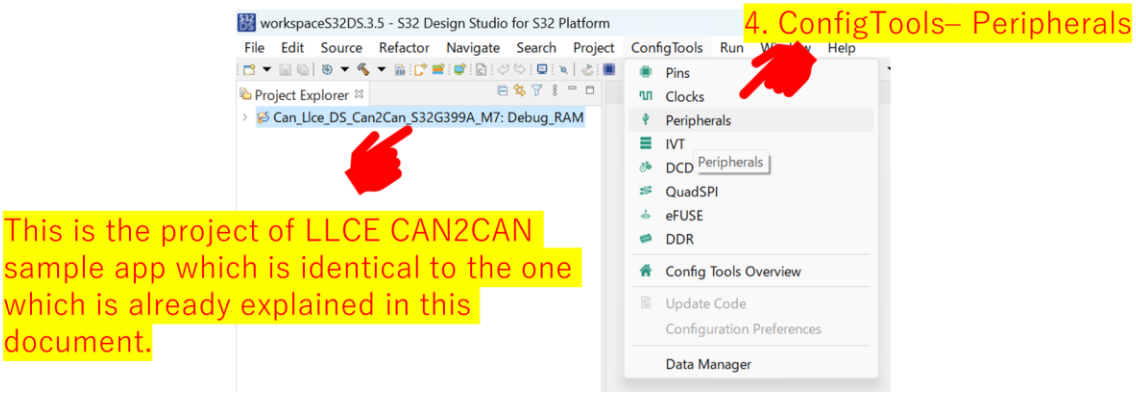


Figure 65. Selecting Peripheral

4. To set up configuration tools Select Can_Llce_DS_Can2Can_S32Gxxx_M7.

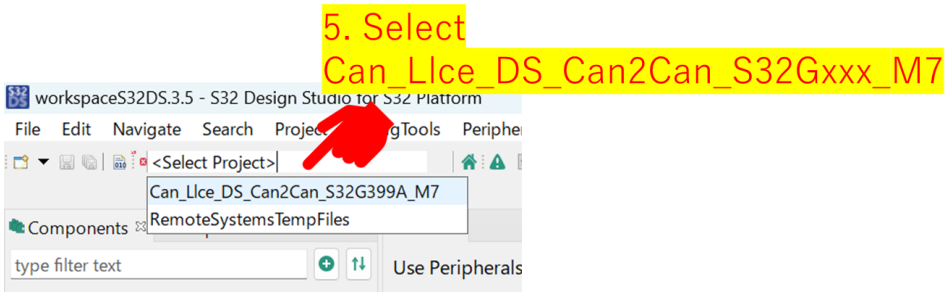


Figure 66. Configuring tools

5. Update the code by clicking on Update Code. At first you see error indicators but once code is updated, they would be all disappeared.

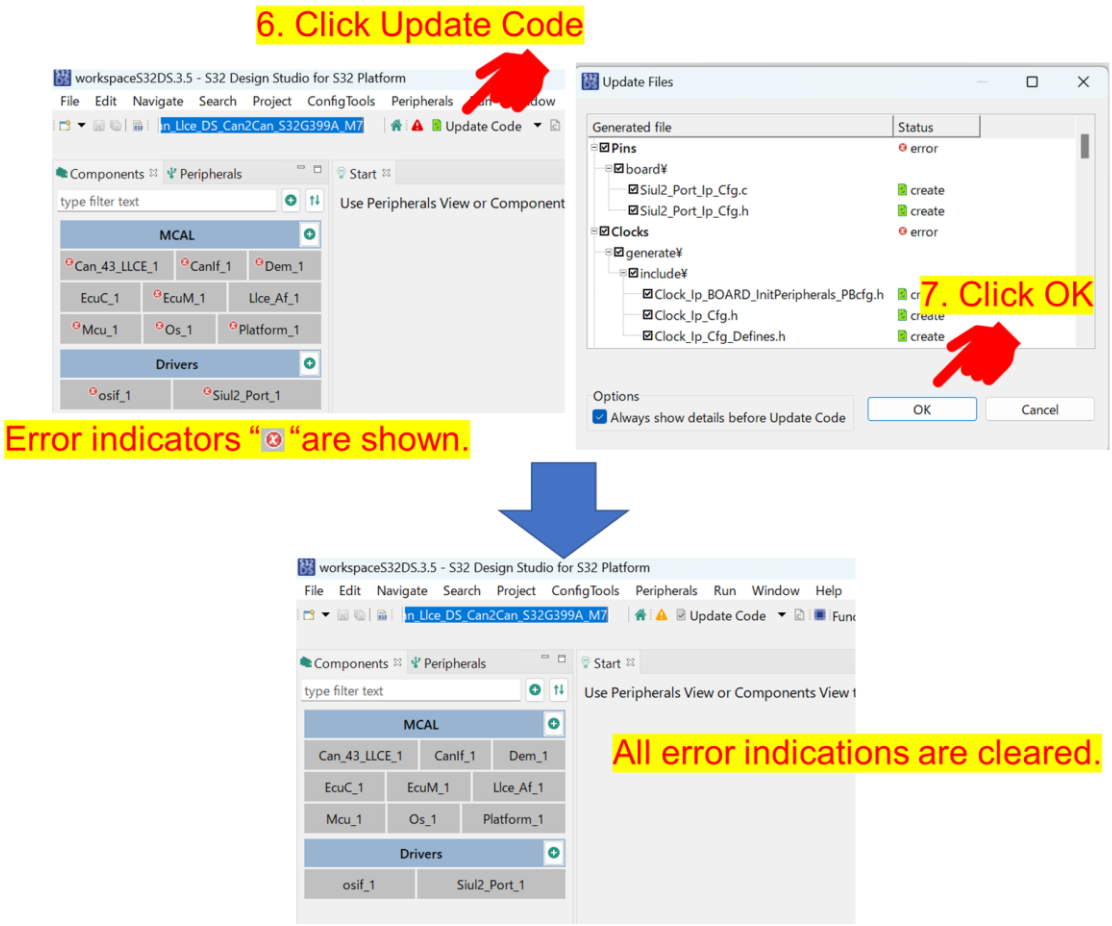


Figure 67. Updating code

Click on the C code view icon in the window and copy the LLCE FW binary files from LLCE installed folder to the LLCE_BIN_DIR folder.

LLCE FW binary files location :

S32G3 C:\NXP\S32G_LLCE_1_0_5_QLP1\firmware\llce_bin\s32g3\bin\ghs\enablement

S32G2 C:\NXP\S32G_LLCE_1_0_5_QLP1\firmware\llce_bin\s32g2\bin\ghs\enablement

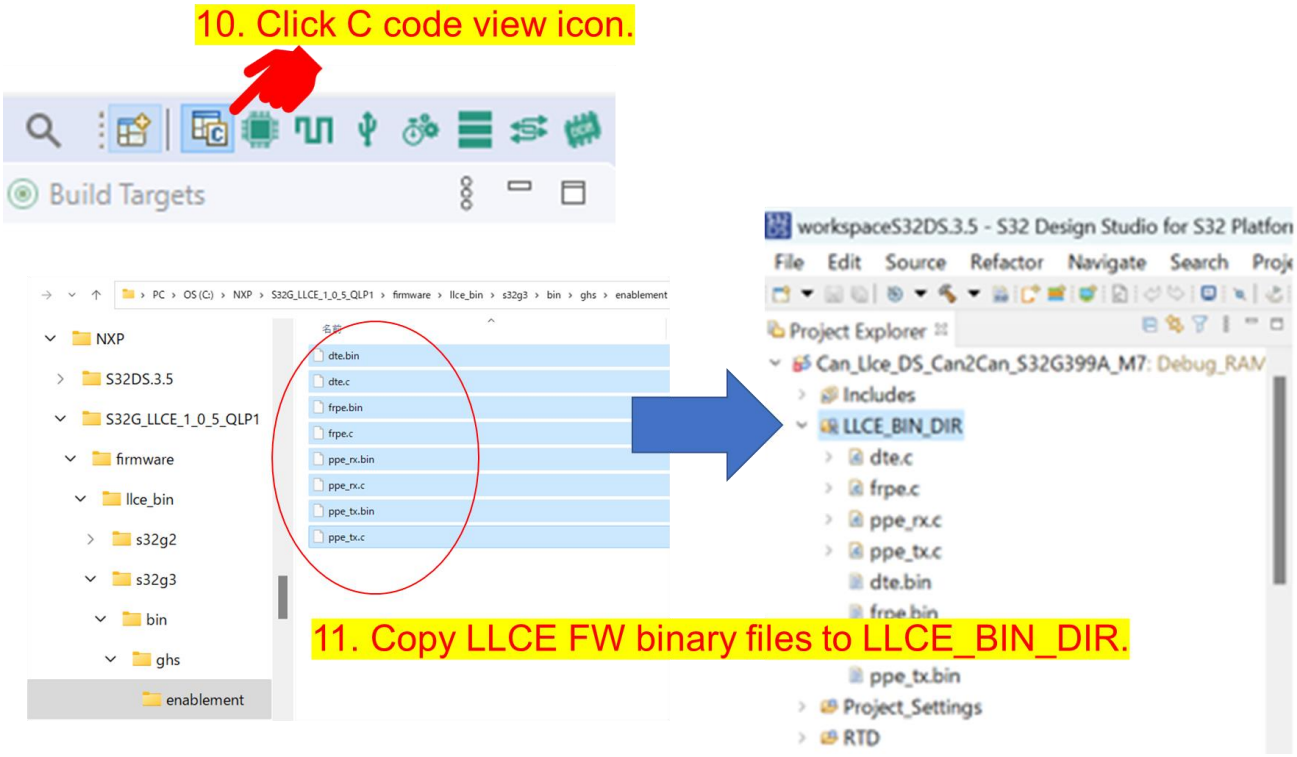


Figure 68. Copy LLCE FW binary files

Now you are ready to build the CAN2CAN sample application once all the steps are successfully completed.

To build the sample application you have to. Right click the project and select Clean Project in the window click on select Build Project.

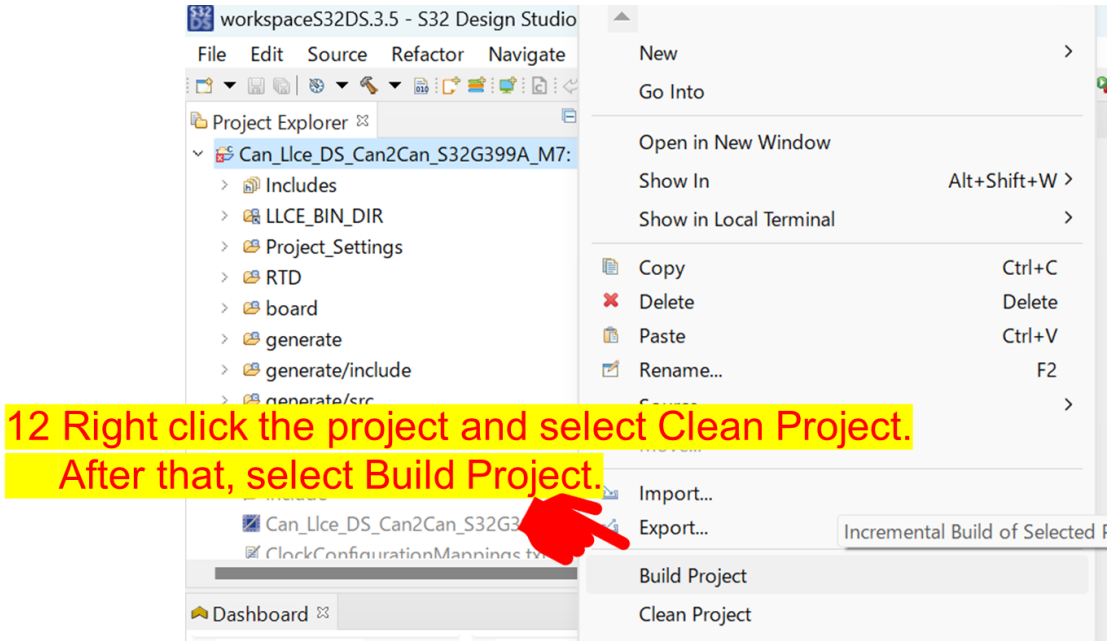


Figure 69. Build project

The Elf file can be found in your workspace inside the folder "Can_Llce_DS_Can2Can_S32Gxxx_M7/Debug_RAM".

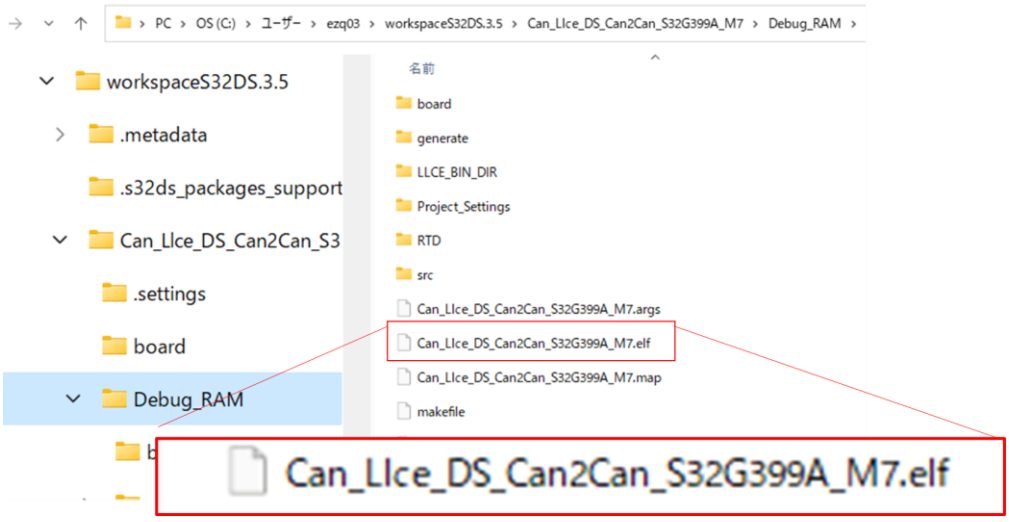


Figure 70. Elf file location

6.1. Configuring LLCE_Af for CAN2CAN

Follow the steps to configure LLCE_Af for CAN2CAN.

1. In Llce_Af, Configure Can2CanRoutingTable click on LLCE_Af_1.
2. Scroll to Can2CanRoutingTable part.
3. You can add/delete these for your CAN2CAN use case. In order to configure routing details, click the index.

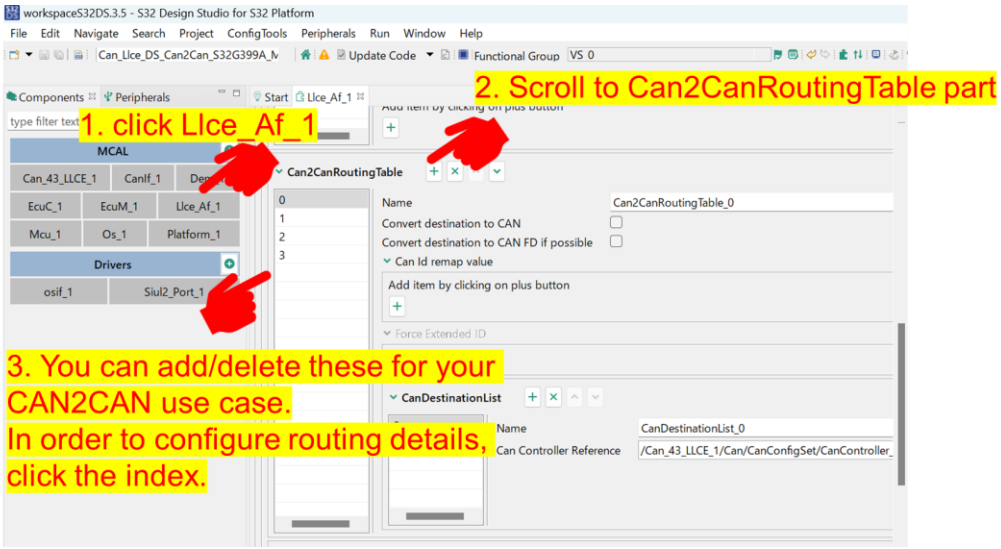


Figure 71. Configuring LLCE_Af for CAN2CAN

4. To configure routing details in the Can2CanRoutingTable part you can either convert FD to Classic or convert Classic to FD in the CAN2CAN routing there are two checkboxes.

5. If you want to remap CAN frame ID when the CAN2CAN routing click on the plus button under Add item by clicking the plus button and enter remap ID value.

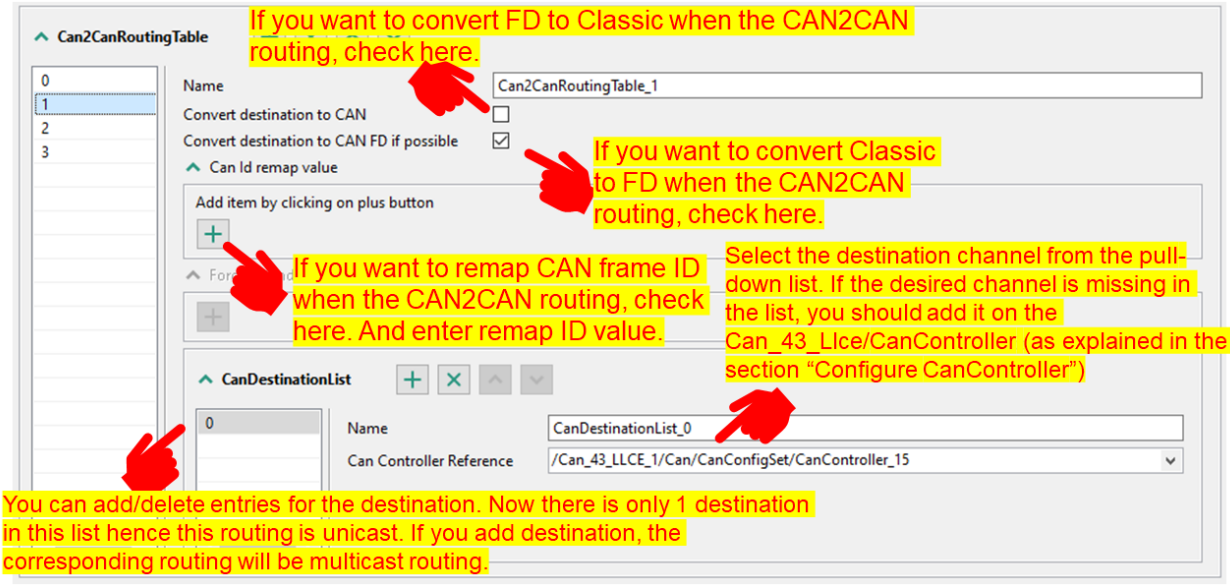


Figure 72. CAN2CAN routing table

6. You can select the destination channel from the pull down list, if it is missing you can add it as explained in Configuring Can controller.
7. You can enter and delete the entries for destination in the CanDestinationList.

NOTE

The entries will be referred from Hardware Receive Handle, which will be configured in Can_43_LLCE/CanHardwareObject.

6.2. Configuring CanController

In the following configuration example, BCAN0, 1, 14 and 15 are configured. To add BCAN follow the steps mentioned below.

1. Click Can43_LLCE_1.
2. Select CanConfigSet tab.
3. Select CanController tab.
4. Right click at 3 (i.e. CanController_15) for example.
5. Click Copy and then click on "+” button to add BCAN.

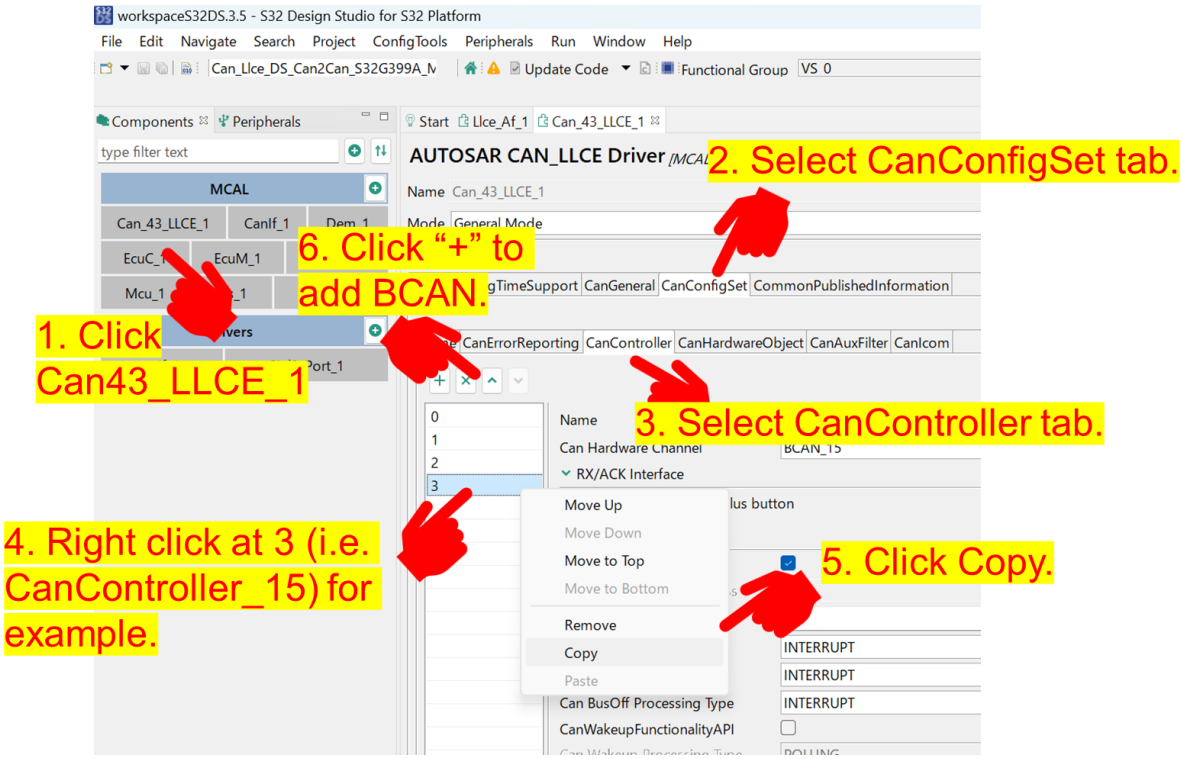


Figure 73. Configuring CanController

6. Right click at newly added BCAN (i.e. 4 in this case) and then click Paste to copy the BCAN15’s configuration.

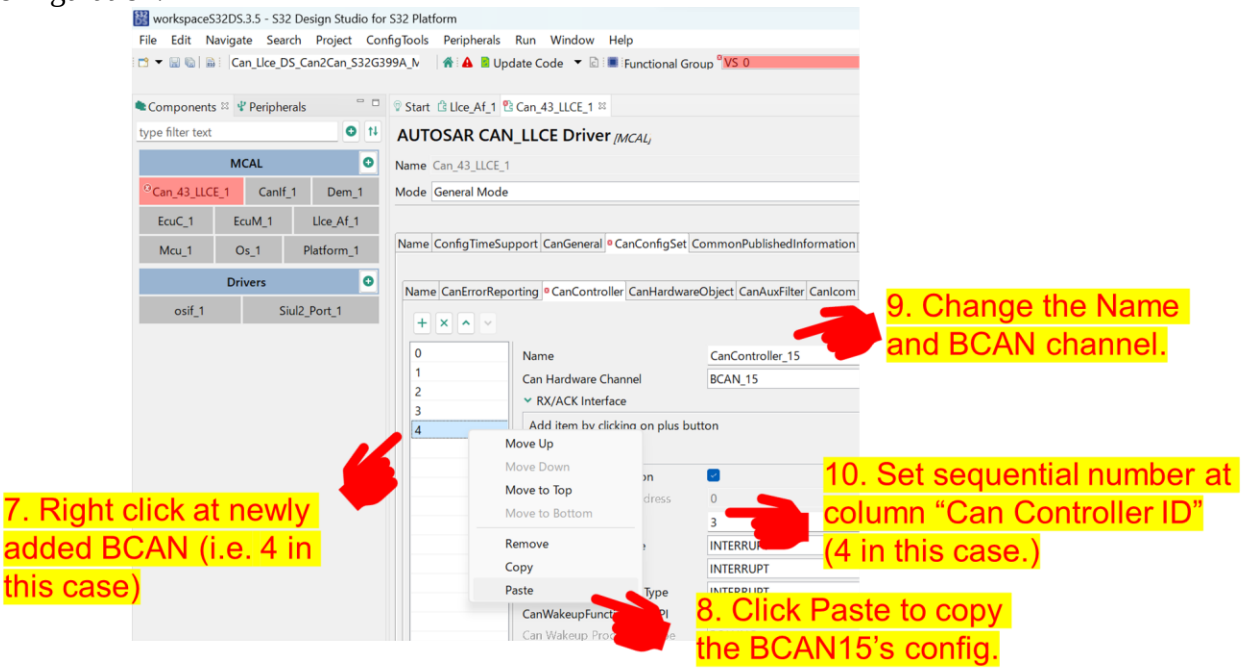


Figure 74. Configuring CanControler 2

7. Change the Name and BCAN channel and set sequential number at column “Can Controller ID”

(4 in this case).

8. Scroll to CanControllerBaudrateConfig part and click the index of any of these (in this explanation, click index 0).
9. Configure baud rate parameters for arb phase.

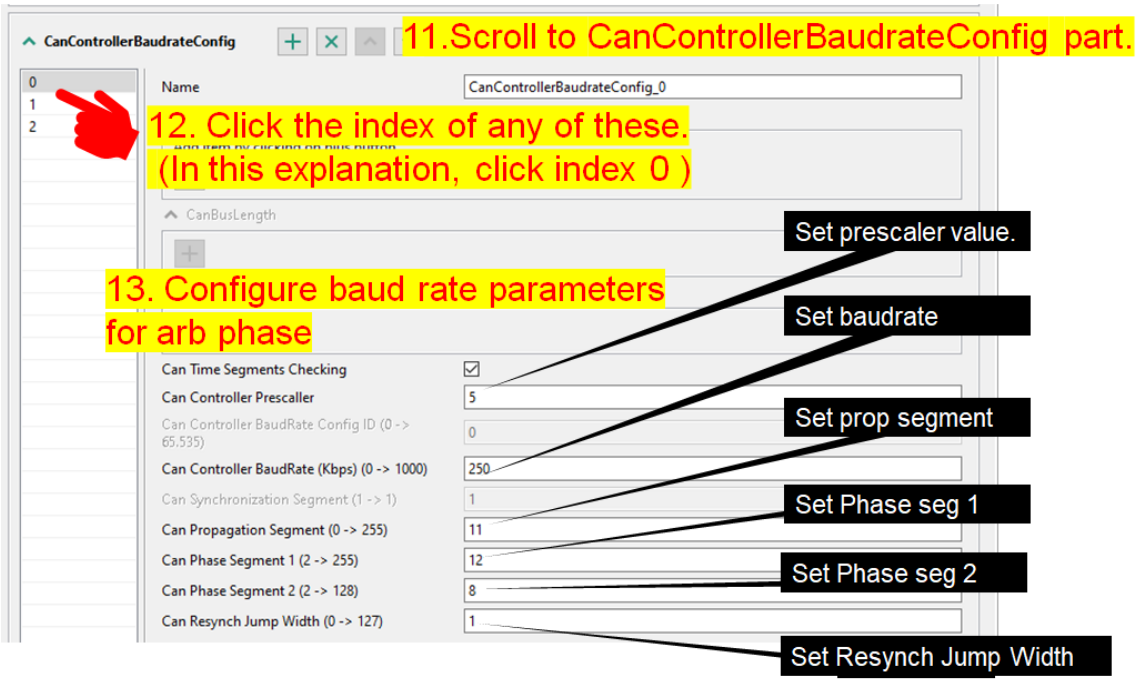


Figure 75. Configuring Baud rate setting

10. Configure baud rate parameters for data phase.

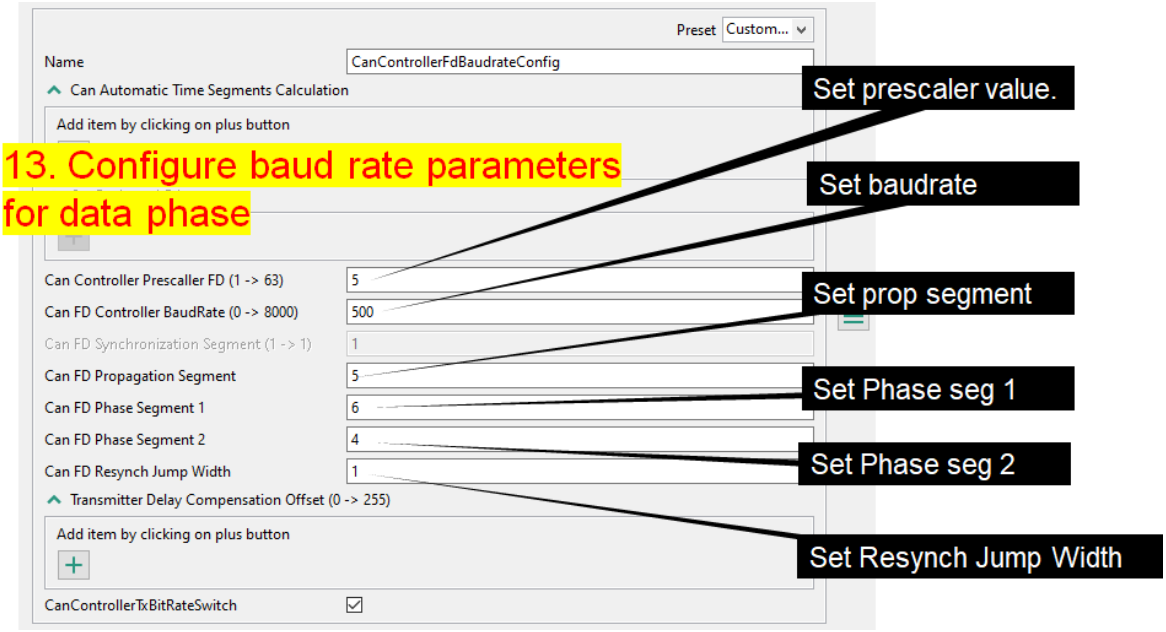


Figure 76. Configuring Baud rate parameters

6.3. Configuring CAN hardware object

To configure CAN hardware object follow the steps mentioned below.

1. Click on Can43_LLCE_1, select CanConfigSet tab and then select CanHardwareObject tab.

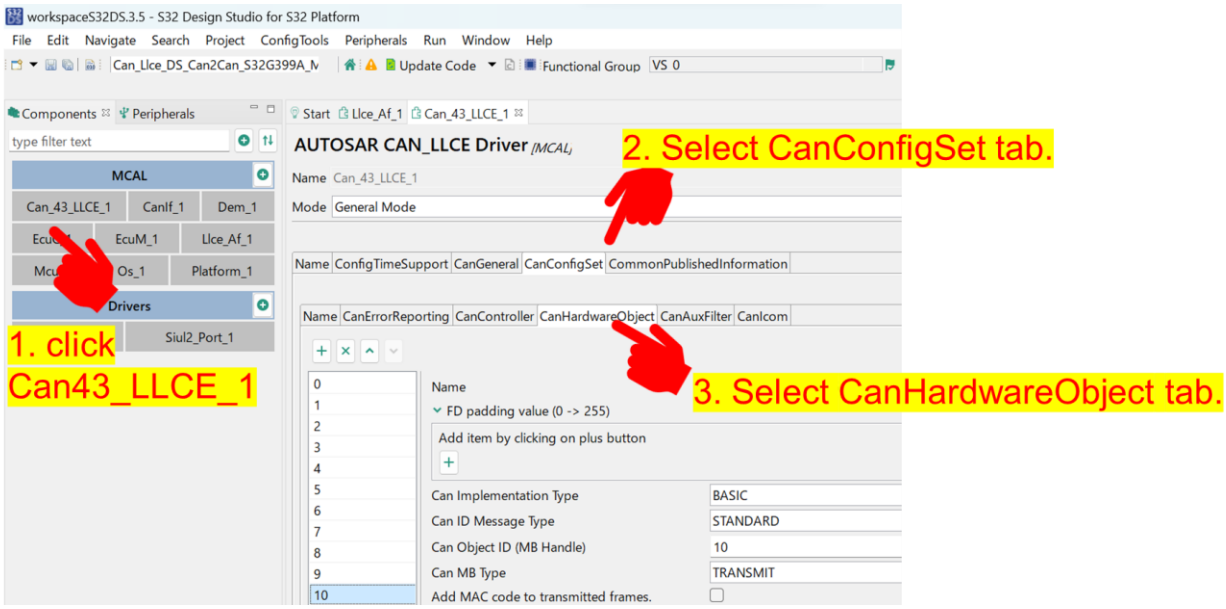


Figure 77. Configuring Can hardware object

2. You have the option to choose or add or remove the Hardware Object Handle in the Left side of the window.
3. In the right side of the window you can name the object, select the ID mask. You can also select the CAN frame type, select the Object Handle type and also select the MB type (RX or TX). You can also specify which CanController has the object.

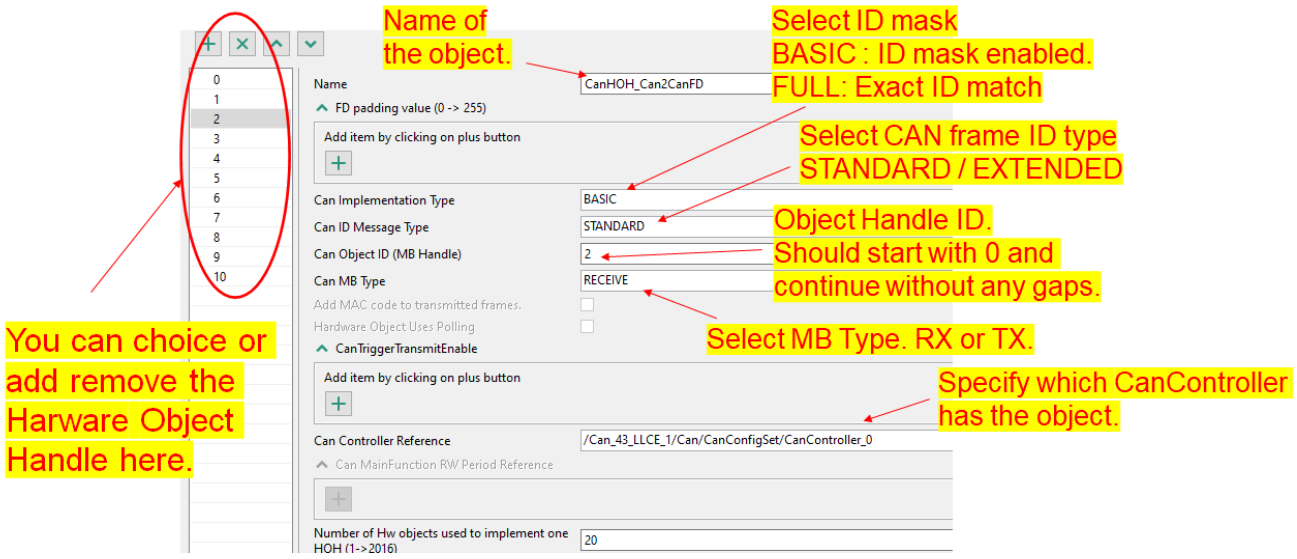


Figure 78. Configuring Can hardware object 2

4. In this window you can configure message buffer related settings.

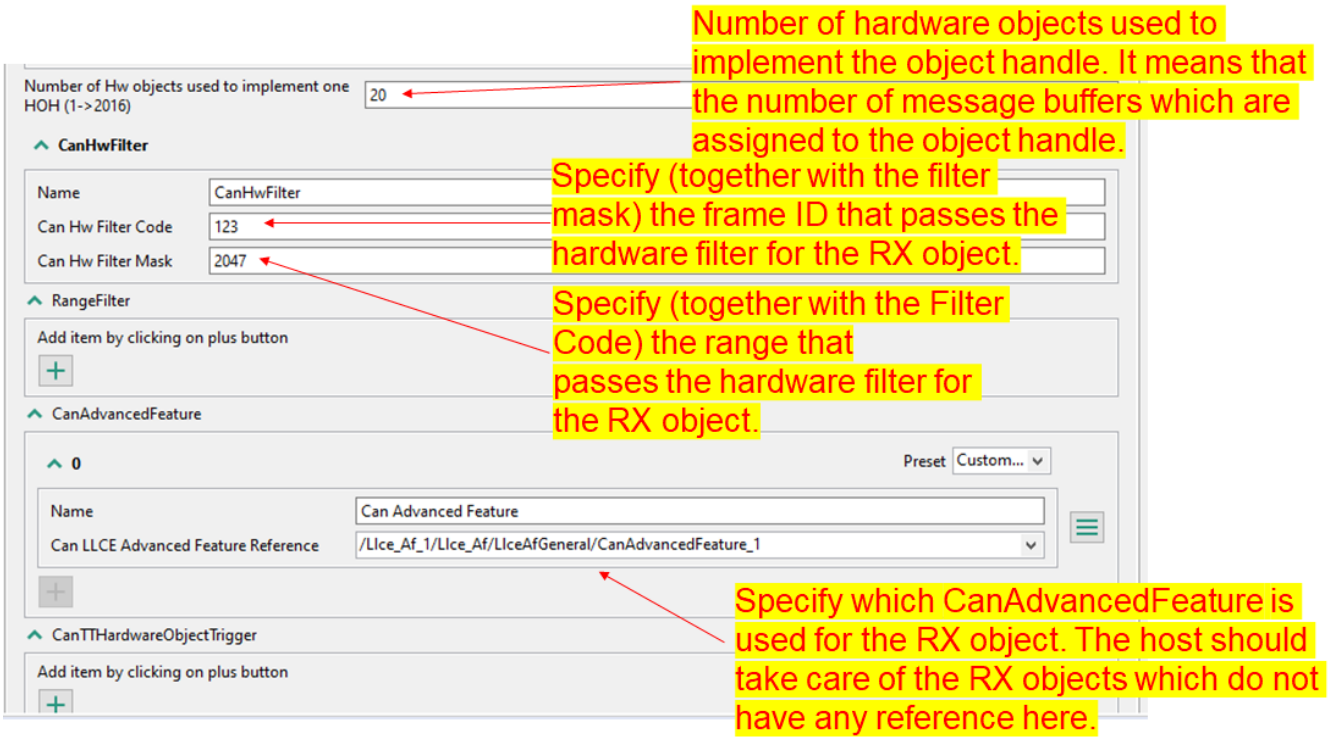


Figure 79. Message buffer configuration

7. Revision history

Revision No.	Release Date	Changes
0	11/2021	Initial release
1	03/2022	• Updated the bullets in CAN2CAN, CAN2ETH and ETH2CAN features. • Updated the note in Using sample application. • Updated Modifying the files and make. • Updated Configuring LLCE_Af for CAN2CAN.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an “as is” and “with all faults” basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer’s exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Suitability for use in automotive and/or industrial applications — This NXP product has been qualified for use in automotive and/or industrial applications. It has been developed in accordance with ISO 26262 respectively IEC 61508, and has been ASIL- respectively SIL-classified accordingly. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as “Critical Applications”), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys’ fees) that NXP may incur related to customer’s incorporation of any product in a Critical Application.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Airfast — is a trademark of NXP B.V.

Altivec — is a trademark of NXP B.V.

CodeWarrior — is a trademark of NXP B.V.

ColdFire — is a trademark of NXP B.V.

ColdFire+ — is a trademark of NXP B.V.

CoolFlux — is a trademark of NXP B.V.

CoolFlux DSP — is a trademark of NXP B.V.

DESFire — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

EdgeScale — is a trademark of NXP B.V.

EdgeVerse — is a trademark of NXP B.V.

eIQ — is a trademark of NXP B.V.

Embrace — is a trademark of NXP B.V.

Freescale — is a trademark of NXP B.V.

GreenChip — is a trademark of NXP B.V.

HITAG — is a trademark of NXP B.V.

ICODE and I-CODE — are trademarks of NXP B.V.

Immersiv3D — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

JCOP — is a trademark of NXP B.V.

Kinetis — is a trademark of NXP B.V.

Layerscape — is a trademark of NXP B.V.

MagniV — is a trademark of NXP B.V.

Mantis — is a trademark of NXP B.V.

MCCI — is a trademark of NXP B.V.

MIFARE — is a trademark of NXP B.V.

MIFARE Classic — is a trademark of NXP B.V.

MIFARE FleX — is a trademark of NXP B.V.

MIFARE4Mobile — is a trademark of NXP B.V.

MIFARE Plus — is a trademark of NXP B.V.

MIFARE Ultralight — is a trademark of NXP B.V.

MiGLO — is a trademark of NXP B.V.

MOBILEGT — is a trademark of NXP B.V.

NTAG — is a trademark of NXP B.V.

NXP SECURE CONNECTIONS FOR A SMARTER WORLD — is a trademark of NXP B.V.

PEG — is a trademark of NXP B.V.

Plus X — is a trademark of NXP B.V.

POR — is a trademark of NXP B.V.

PowerQUICC — is a trademark of NXP B.V.

Processor Expert — is a trademark of NXP B.V.

QorIQ — is a trademark of NXP B.V.

QorIQ Qonverge — is a trademark of NXP B.V.

SafeAssure — is a trademark of NXP B.V.

SafeAssure — logo is a trademark of NXP B.V.

SmartLX — is a trademark of NXP B.V.

SmartMX — is a trademark of NXP B.V.

StarCore — is a trademark of NXP B.V.

Symphony — is a trademark of NXP B.V.

Synopsys & Designware — are registered trademarks of Synopsys, Inc.

Synopsys — Portions Copyright © 2021 Synopsys, Inc. Used with permission. All rights reserved.

Tower — is a trademark of NXP B.V.

TriMedia — is a trademark of NXP B.V.

UCODE — is a trademark of NXP B.V.

VortiQa — is a trademark of NXP B.V.

Vybrid — is a trademark of NXP B.V.

arm

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 03/2023
Document identifier: AN13423