NXP Semiconductors
Application Notes

Document Number: AN13423
Rev. 0, 11/2021

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. PFE sends it to the Ethernet.

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

Contents

route it to CAN channels. The following figure shows an overview diagram of CAN2CAN, CAN2ETH and ETH2CAN with respect to LLCE in S32G2.

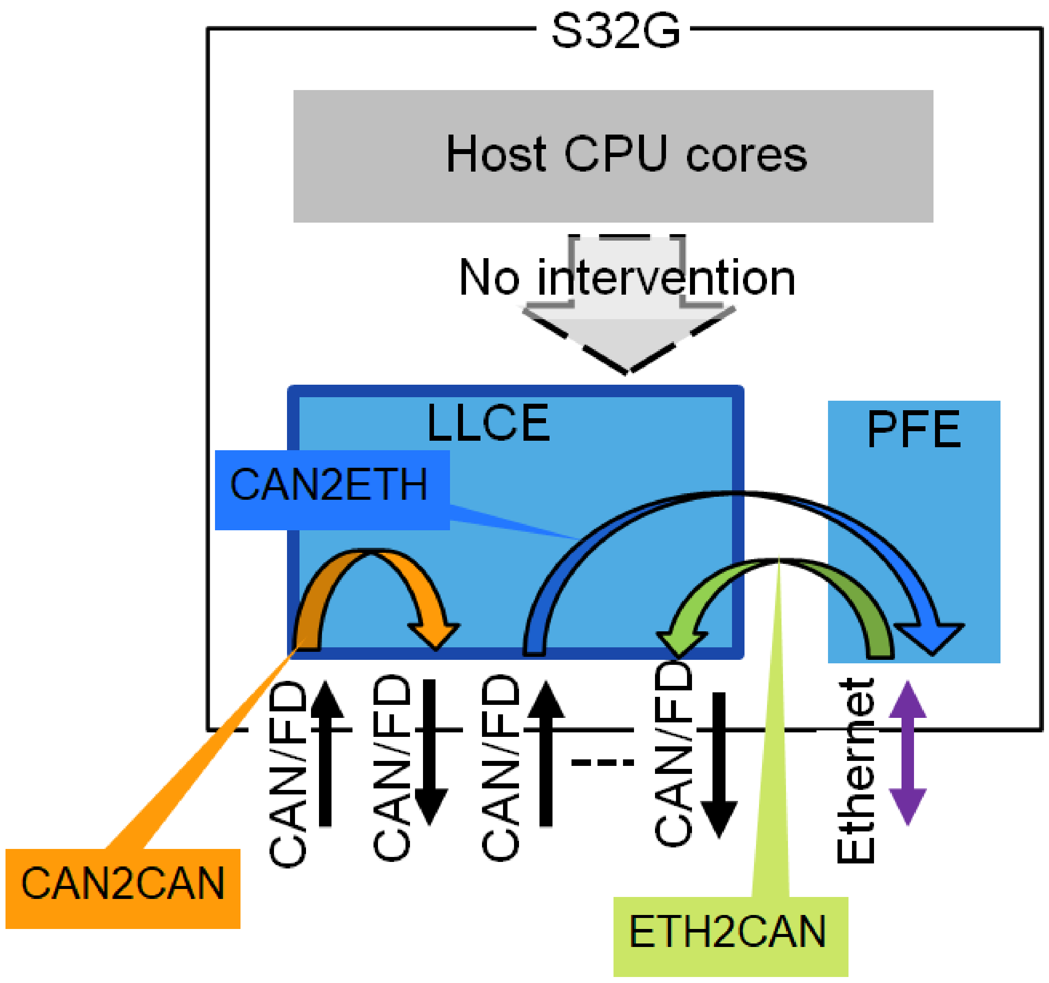


Figure 1. **Overview diagram**

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.

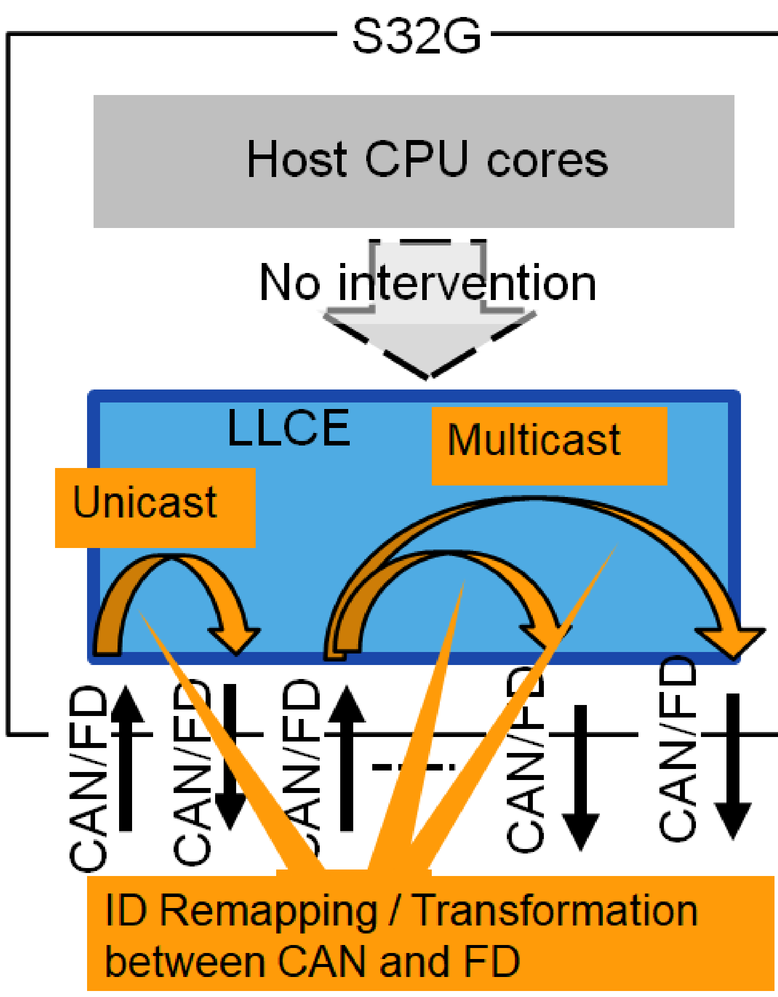


Figure 2. **CAN2CAN feature**

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 Non time synchronous control format. Packed as ACF CAN Brief messages.
- 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.
- The source CAN bus is not recorded in the can_bus_id field.
- Stream ID is constant, Not configurable.

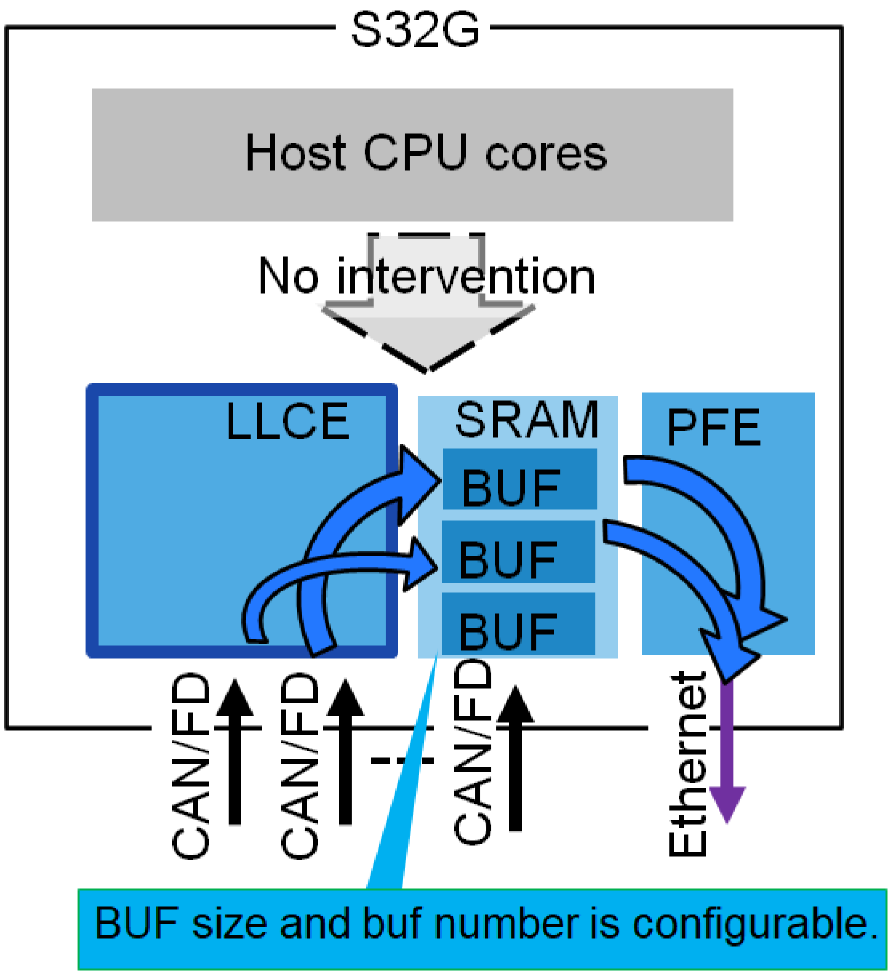


Figure 3. **CAN2ETH feature**

The ETH2CAN feature performs the off-loaded unpacking IEEE1722 AVTP non time synchronous control format frames. The following steps are taken for unpacking the frames:

- Any IEEE1722 frames will be unpacked and routed. Stream Id and sequential number in the time synchronous control format header are “don’t care’.
- 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.

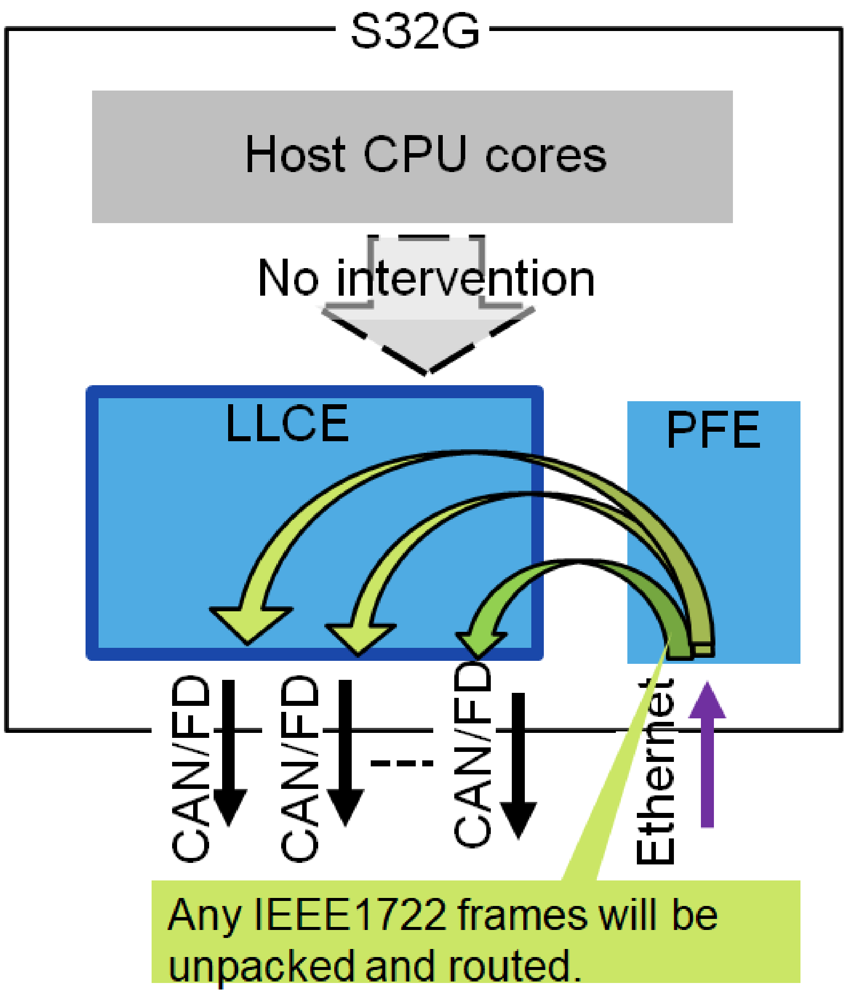


Figure 4. **ETH2CAN feature**

3. Using sample application

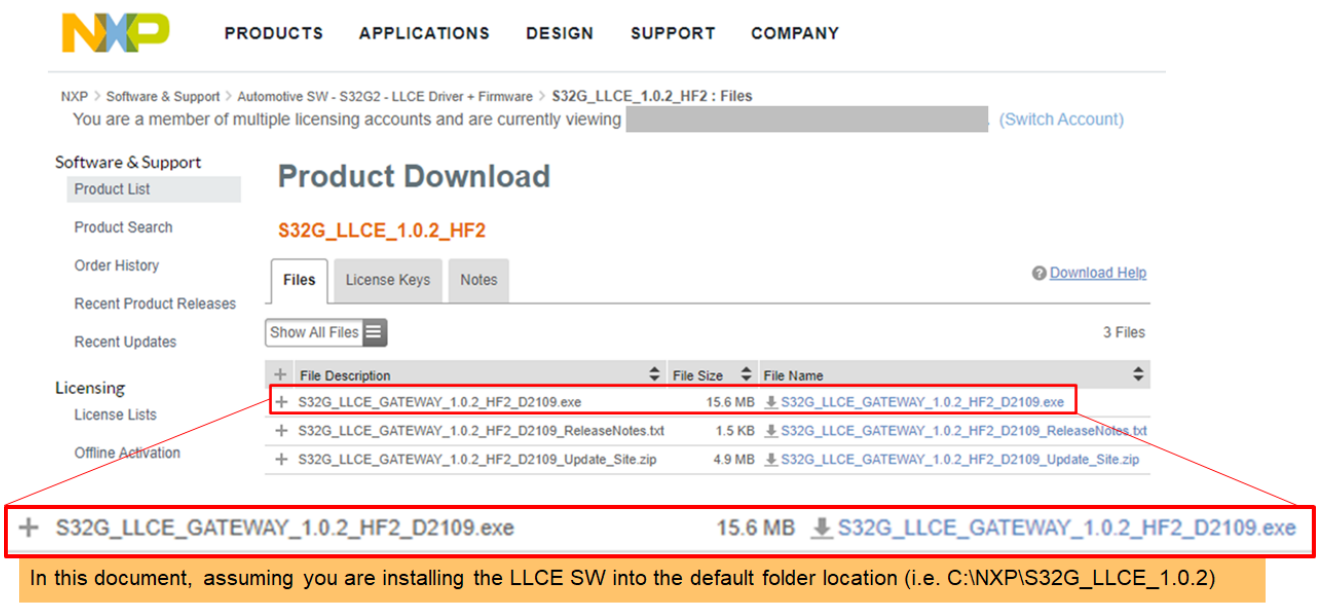
This section and sub sections describes how to use the sample application. The steps that needs to be followed are shown in the section.

NOTE

This section is based on the latest release as of September 2021. (i.e. S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.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.


NXP > Software & Support > Automotive SW - S32G2 - LLCE Driver + Firmware > S32G_LLCE_1.0.2_HF2 : Files

You are a member of multiple licensing accounts and are currently viewing [redacted] (Switch Account)

Product Download

S32G_LLCE_1.0.2_HF2

File Description	File Size	File Name
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.exe	15.6 MB	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.exe
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_ReleaseNotes.txt	1.5 KB	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_ReleaseNotes.txt
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_Update_Site.zip	4.9 MB	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_Update_Site.zip

In this document, assuming you are installing the LLCE SW into the default folder location (i.e. C:\NXP\S32G_LLCE_1.0.2)

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.

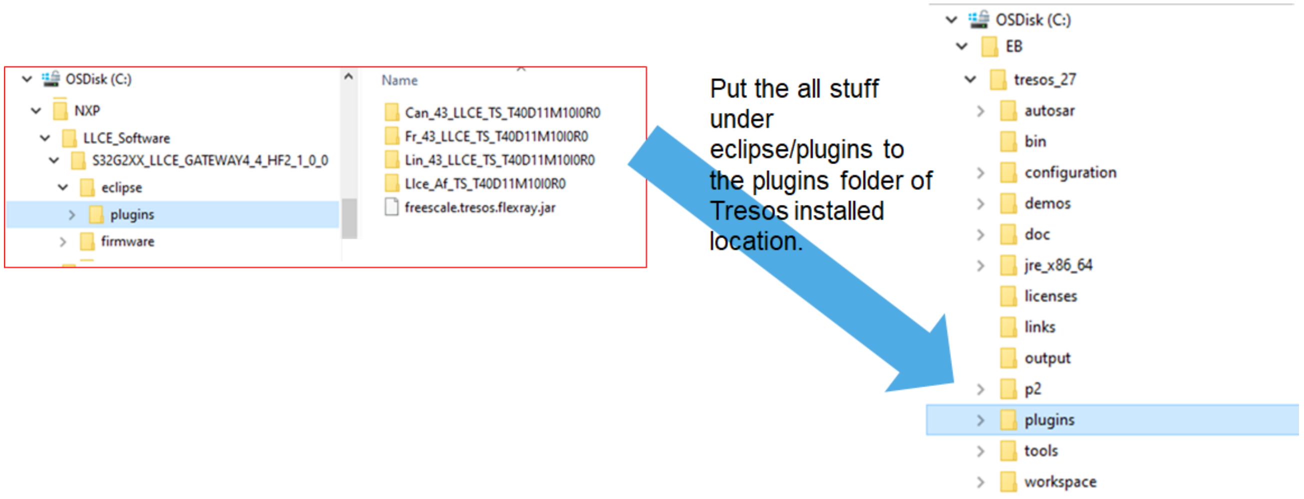
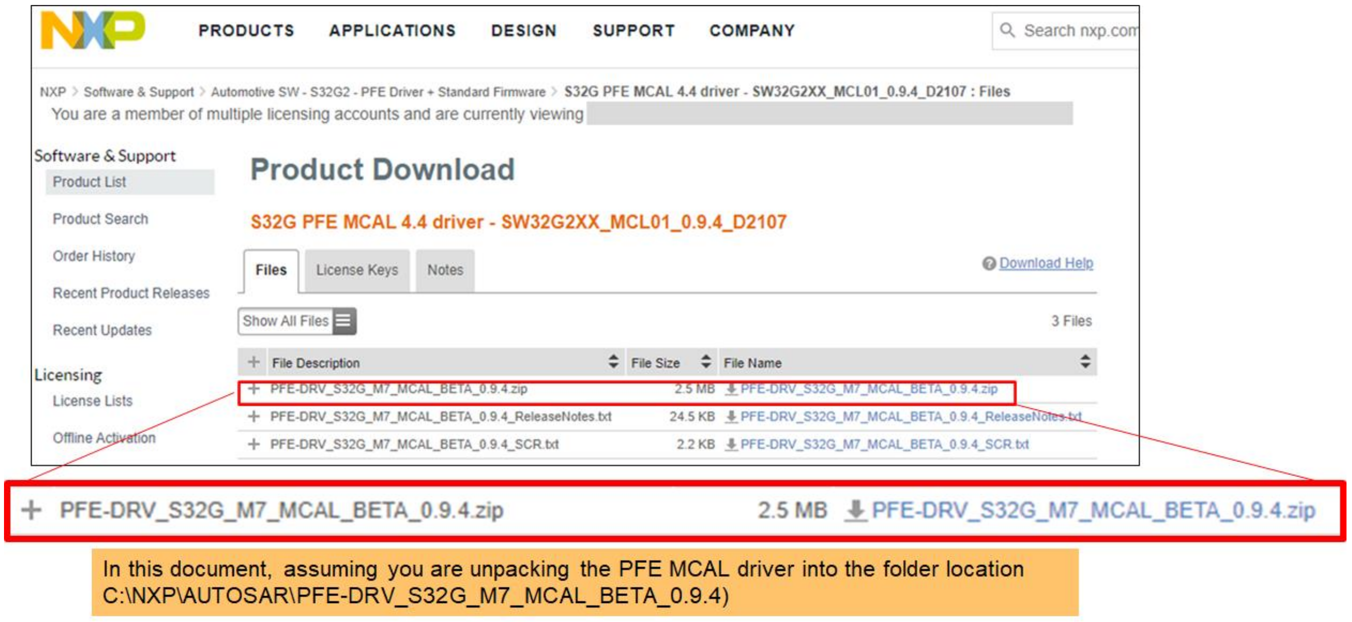

Put the all stuff under eclipse/plugins to the plugins folder of Tresos installed location.

Figure 6. **LLCE SW package in TREOS**

If you do not have the latest PFE MCAL 4.4 driver 0.9.4 and RTD package, download both of them.


PRODUCTS APPLICATIONS DESIGN SUPPORT COMPANY

Search nxp.com

NXP > Software & Support > Automotive SW - S32G2 - PFE Driver + Standard Firmware > S32G PFE MCAL 4.4 driver - SW32G2XX_MCL01_0.9.4_D2107 : Files

You are a member of multiple licensing accounts and are currently viewing [redacted]

Software & Support
- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing
- License Lists
- Offline Activation

Product Download

S32G PFE MCAL 4.4 driver - SW32G2XX_MCL01_0.9.4_D2107

Files | License Keys | Notes — Download Help

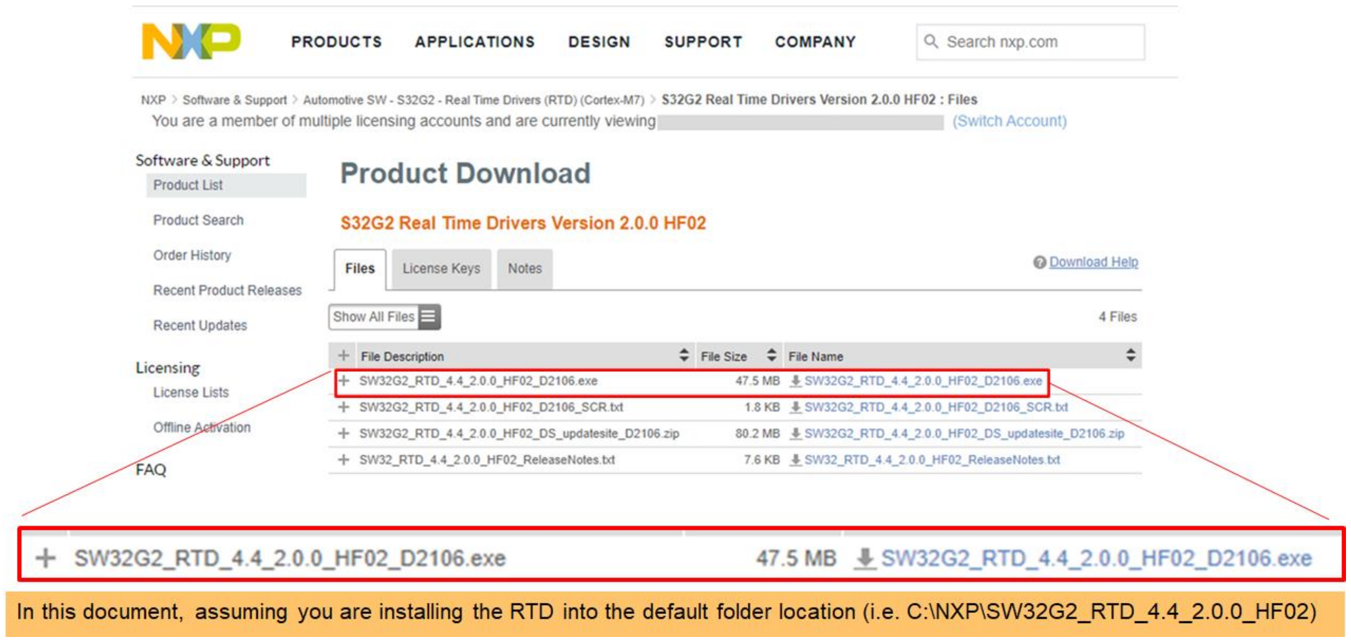
Show All Files — 3 Files

File Description	File Size	File Name
PFE-DRV_S32G_M7_MCAL_BETA_0.9.4.zip	2.5 MB	PFE-DRV_S32G_M7_MCAL_BETA_0.9.4.zip
PFE-DRV_S32G_M7_MCAL_BETA_0.9.4_ReleaseNotes.txt	24.5 KB	PFE-DRV_S32G_M7_MCAL_BETA_0.9.4_ReleaseNotes.txt
PFE-DRV_S32G_M7_MCAL_BETA_0.9.4_SCR.txt	2.2 KB	PFE-DRV_S32G_M7_MCAL_BETA_0.9.4_SCR.txt

+ PFE-DRV_S32G_M7_MCAL_BETA_0.9.4.zip 2.5 MB PFE-DRV_S32G_M7_MCAL_BETA_0.9.4.zip

In this document, assuming you are unpacking the PFE MCAL driver into the folder location C:\NXP\AUTOSAR\PFE-DRV_S32G_M7_MCAL_BETA_0.9.4)

Figure 7. **PFE MCAL 4.4 driver**


PRODUCTS APPLICATIONS DESIGN SUPPORT COMPANY

Search nxp.com

NXP > Software & Support > Automotive SW - S32G2 - Real Time Drivers (RTD) (Cortex-M7) > S32G2 Real Time Drivers Version 2.0.0 HF02 : Files

You are a member of multiple licensing accounts and are currently viewing [redacted] (Switch Account)

Software & Support
- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing
- License Lists
- Offline Activation

FAQ

Product Download

S32G2 Real Time Drivers Version 2.0.0 HF02

Files | License Keys | Notes — Download Help

Show All Files — 4 Files

File Description	File Size	File Name
SW32G2_RTD_4.4_2.0.0_HF02_D2106.exe	47.5 MB	SW32G2_RTD_4.4_2.0.0_HF02_D2106.exe
SW32G2_RTD_4.4_2.0.0_HF02_D2106_SCR.txt	1.8 KB	SW32G2_RTD_4.4_2.0.0_HF02_D2106_SCR.txt
SW32G2_RTD_4.4_2.0.0_HF02_DS_updatesite_D2106.zip	80.2 MB	SW32G2_RTD_4.4_2.0.0_HF02_DS_updatesite_D2106.zip
SW32_RTD_4.4_2.0.0_HF02_ReleaseNotes.txt	7.6 KB	SW32_RTD_4.4_2.0.0_HF02_ReleaseNotes.txt

+ SW32G2_RTD_4.4_2.0.0_HF02_D2106.exe 47.5 MB SW32G2_RTD_4.4_2.0.0_HF02_D2106.exe

In this document, assuming you are installing the RTD into the default folder location (i.e. C:\NXP\SW32G2_RTD_4.4_2.0.0_HF02)

Figure 8. **RTD package**

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

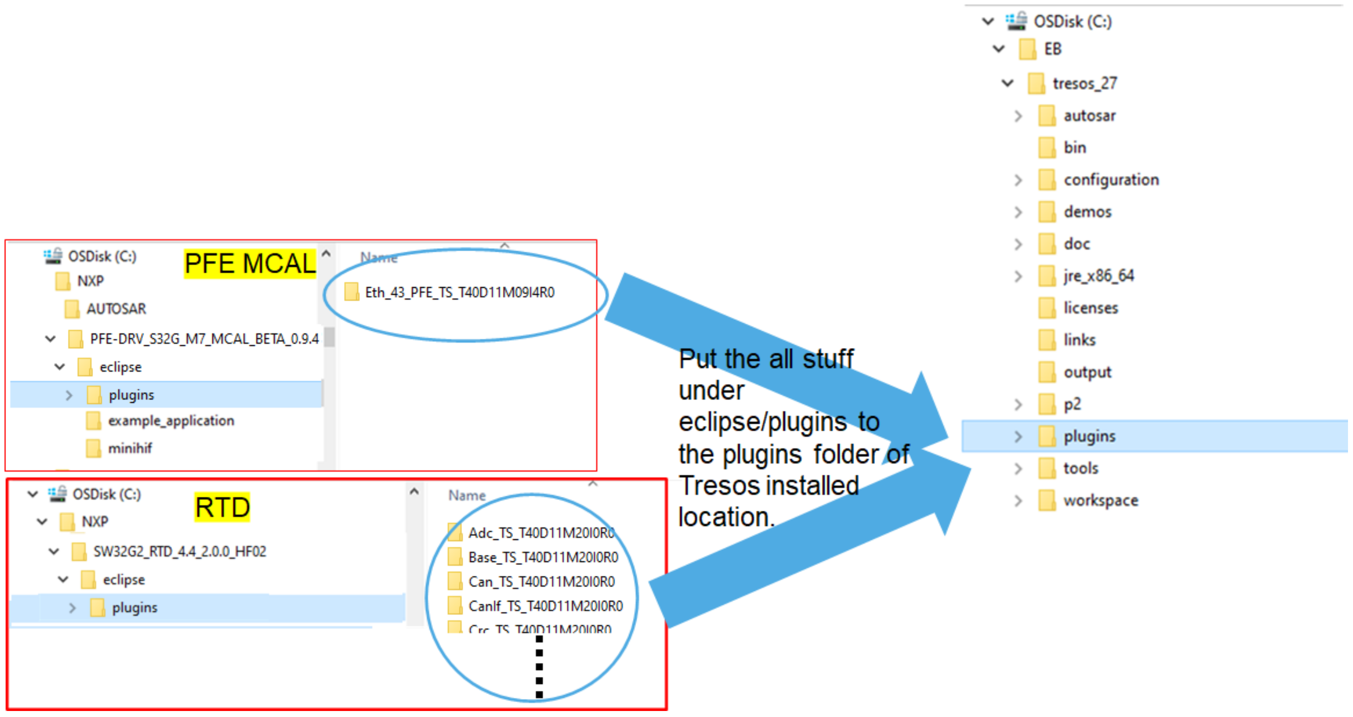
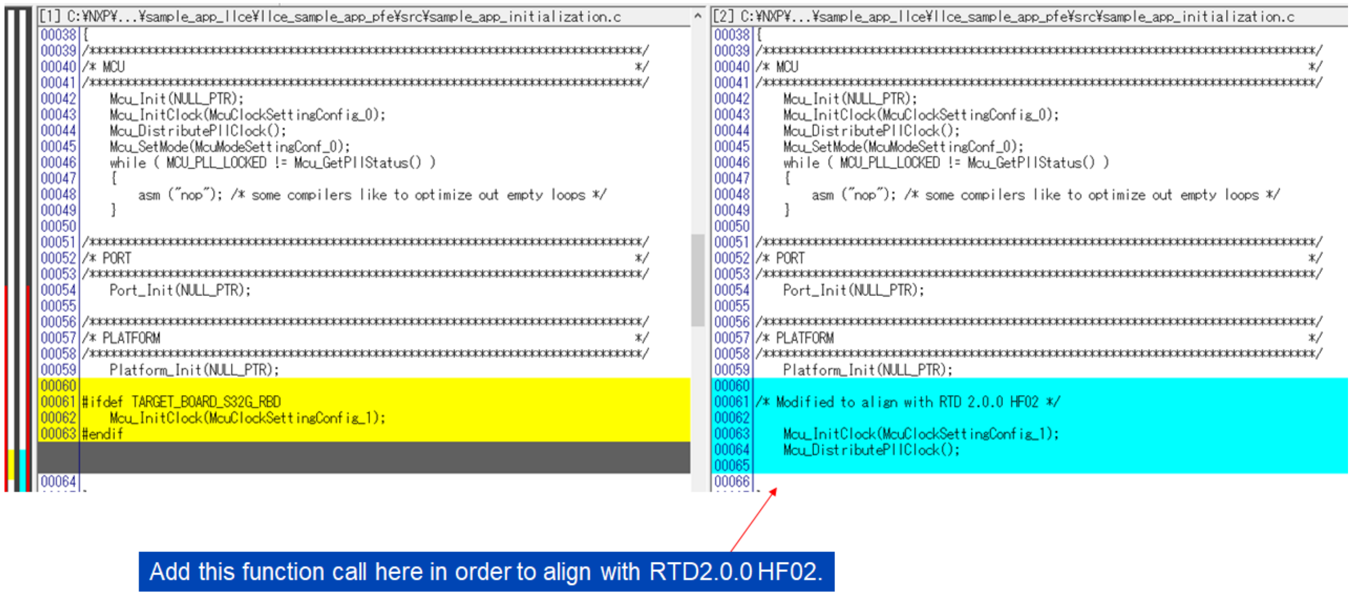


Figure 9. **PFE MCAL and RTD in tresos**

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


```
[1] C:¥NXP¥...¥sample_app_llce¥llce_sample_app_pfe¥src¥sample_app_initialization.c
{
/****************************************************************************/
/* MCU                                                                      */
/****************************************************************************/
    Mcu_Init(NULL_PTR);
    Mcu_InitClock(McuClockSettingConfig_0);
    Mcu_DistributePllClock();
    Mcu_SetMode(McuModeSettingConf_0);
    while ( MCU_PLL_LOCKED != Mcu_GetPllStatus() )
    {
        asm ("nop"); /* some compilers like to optimize out empty loops */
    }

/****************************************************************************/
/* PORT                                                                     */
/****************************************************************************/
    Port_Init(NULL_PTR);

/****************************************************************************/
/* PLATFORM                                                                 */
/****************************************************************************/
    Platform_Init(NULL_PTR);

#ifdef TARGET_BOARD_S32G_RBD
    Mcu_InitClock(McuClockSettingConfig_1);
#endif
```

```
[2] C:¥NXP¥...¥sample_app_llce¥llce_sample_app_pfe¥src¥sample_app_initialization.c
{
/****************************************************************************/
/* MCU                                                                      */
/****************************************************************************/
    Mcu_Init(NULL_PTR);
    Mcu_InitClock(McuClockSettingConfig_0);
    Mcu_DistributePllClock();
    Mcu_SetMode(McuModeSettingConf_0);
    while ( MCU_PLL_LOCKED != Mcu_GetPllStatus() )
    {
        asm ("nop"); /* some compilers like to optimize out empty loops */
    }

/****************************************************************************/
/* PORT                                                                     */
/****************************************************************************/
    Port_Init(NULL_PTR);

/****************************************************************************/
/* PLATFORM                                                                 */
/****************************************************************************/
    Platform_Init(NULL_PTR);

/* Modified to align with RTD 2.0.0 HF02 */

    Mcu_InitClock(McuClockSettingConfig_1);
    Mcu_DistributePllClock();

```

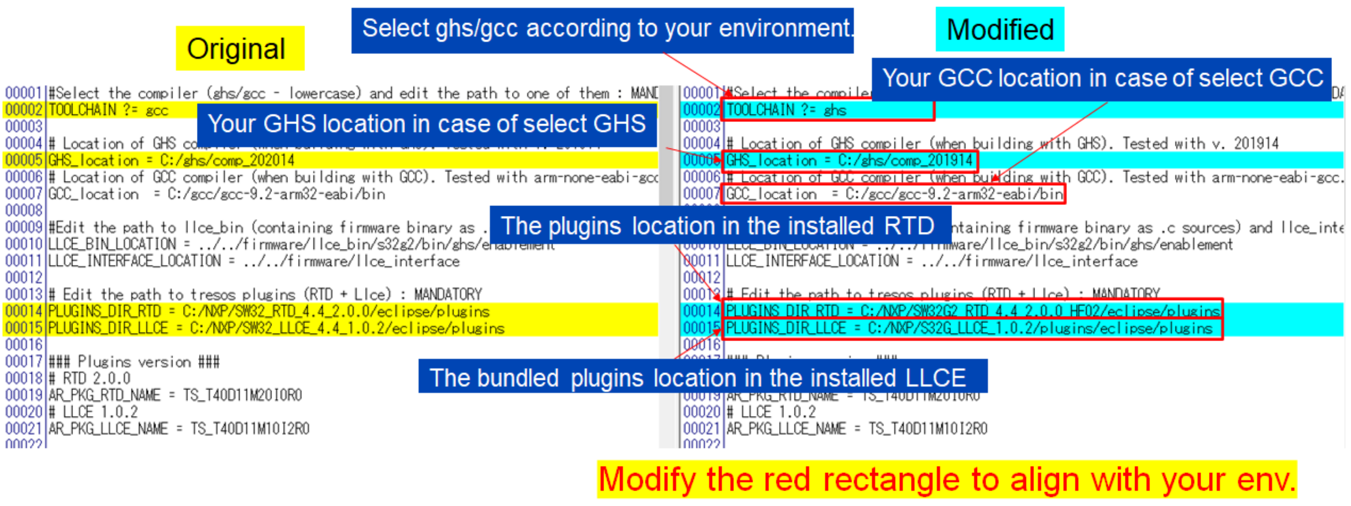
Add this function call here in order to align with RTD2.0.0 HF02.

Figure 10. **Modifying sample_app_initialization.c file**

3.2. Modifying the files and make

Modify the config.mk for your environment. For CAN2CAN, modify the file: C:\NXP\S32G_LLCE_1.0.2\sample_app_llce\llce_sample_app_af\config.mk.

Please refer to the following screenshot.


Original
Select ghs/gcc according to your environment.
Modified
Your GCC location in case of select GCC
Your GHS location in case of select GHS
The plugins location in the installed RTD
The bundled plugins location in the installed LLCE
Modify the red rectangle to align with your env.

```
#Select the compiler (ghs/gcc - lowercase) and edit the path to one of them : MAND
TOOLCHAIN ?= gcc

# Location of GHS co
GHS_location = C:/ghs/comp_202014
# Location of GCC compiler (when building with GCC). Tested with arm-none-eabi-gcc
GCC_location  = C:/gcc/gcc-9.2-arm32-eabi/bin

#Edit the path to llce_bin (containing firmware binary as .
LLCE_BIN_LOCATION = ../../firmware/llce_bin/s32g2/bin/ghs/e
LLCE_INTERFACE_LOCATION = ../../firmware/llce_interface

# Edit the path to tresos plugins (RTD + Llce) : MANDATORY
PLUGINS_DIR_RTD = C:/NXP/SW32_RTD_4.4_2.0.0/eclipse/plugins
PLUGINS_DIR_LLCE = C:/NXP/SW32_LLCE_4.4_1.0.2/eclipse/plugins

### Plugins version ###
# RTD 2.0.0
AR_PKG_RTD_NAME = TS_T40D11M20I0R0
# LLCE 1.0.2
AR_PKG_LLCE_NAME = TS_T40D11M10I2R0
```

```
TOOLCHAIN ?= ghs

# Location of GHS compiler (when building with GHS). Tested with v. 201914
GHS_location = C:/ghs/comp_201914
# Location of GCC compiler (when building with GCC). Tested with arm-none-eabi-gcc.
GCC_location  = C:/gcc/gcc-9.2-arm32-eabi/bin

ntaining firmware binary as .c sources) and llce_inte
ware/llce_bin/s32g2/bin/ghs/enablement
LLCE_INTERFACE_LOCATION = ../../firmware/llce_interface

# Edit the path to tresos plugins (RTD + Llce) : MANDATORY
PLUGINS_DIR_RTD = C:/NXP/SW32G2_RTD_4.4_2.0.0_HF02/eclipse/plugins
PLUGINS_DIR_LLCE = C:/NXP/S32G_LLCE_1.0.2/plugins/eclipse/plugins

# LLCE 1.0.2
AR_PKG_LLCE_NAME = TS_T40D11M10I2R0
```

Figure 11. **Modifying config.mk**

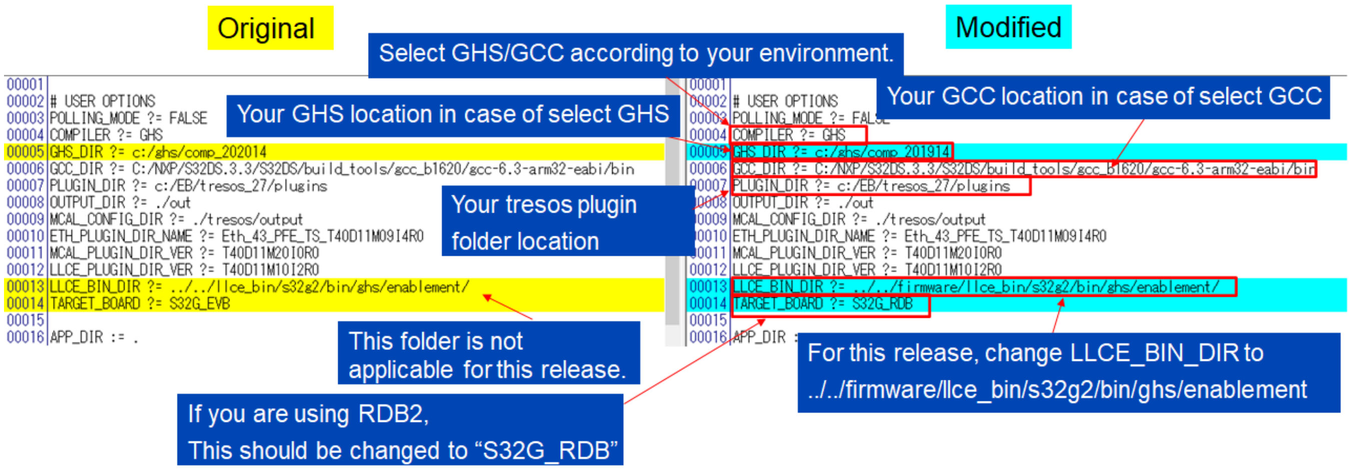
Under llce_sample_app_af folder, you can build as following.

- $make clean
- $make can_routing

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

Modify the config.mak for your environment. For CAN2ETH/ETH2CAN, modify C:\NXP\S32G_LLCE_1.0.2\sample_app_llce\llce_sample_app_pfe\config.mak

Please refer to the following screenshot.


Original
Select GHS/GCC according to your environment.
Modified
Your GCC location in case of select GCC
Your GHS location in case of select GHS
Your tresos plugin folder location
This folder is not applicable for this release.
If you are using RDB2, This should be changed to "S32G_RDB"
For this release, change LLCE_BIN_DIR to ../../firmware/llce_bin/s32g2/bin/ghs/enablement

```

# USER OPTIONS
POLLING_MODE ?= FALSE
COMPILER ?= GHS
GHS_DIR ?= c:/ghs/comp_202014
GCC_DIR ?= C:/NXP/S32DS.3.3/S32DS/build_tools/gcc_b1620/gcc-6.3-arm32-eabi/bin
PLUGIN_DIR ?= c:/EB/tresos_27/plugins
OUTPUT_DIR ?= ./out
MCAL_CONFIG_DIR ?= ./tresos/output
ETH_PLUGIN_DIR_NAME ?= Eth_43_PFE_TS_T40D11M09I4R0
MCAL_PLUGIN_DIR_VER ?= T40D11M20I0R0
LLCE_PLUGIN_DIR_VER ?= T40D11M10I2R0
LLCE_BIN_DIR ?= ../../llce_bin/s32g2/bin/ghs/enablement/
TARGET_BOARD ?= S32G_EVB

APP_DIR := .
```

```

# USER OPTIONS
POLLING_MODE ?= FALSE
COMPILER ?= GHS
GHS_DIR ?= c:/ghs/comp_201914
GCC_DIR ?= C:/NXP/S32DS.3.3/S32DS/build_tools/gcc_b1620/gcc-6.3-arm32-eabi/bin
PLUGIN_DIR ?= c:/EB/tresos_27/plugins
OUTPUT_DIR ?= ./out
MCAL_CONFIG_DIR ?= ./tresos/output
ETH_PLUGIN_DIR_NAME ?= Eth_43_PFE_TS_T40D11M09I4R0
MCAL_PLUGIN_DIR_VER ?= T40D11M20I0R0
LLCE_PLUGIN_DIR_VER ?= T40D11M10I2R0
LLCE_BIN_DIR ?= ../../firmware/llce_bin/s32g2/bin/ghs/enablement/
TARGET_BOARD ?= S32G_RDB

APP_DIR :
```

Figure 12. **Modifying config.mak**

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.

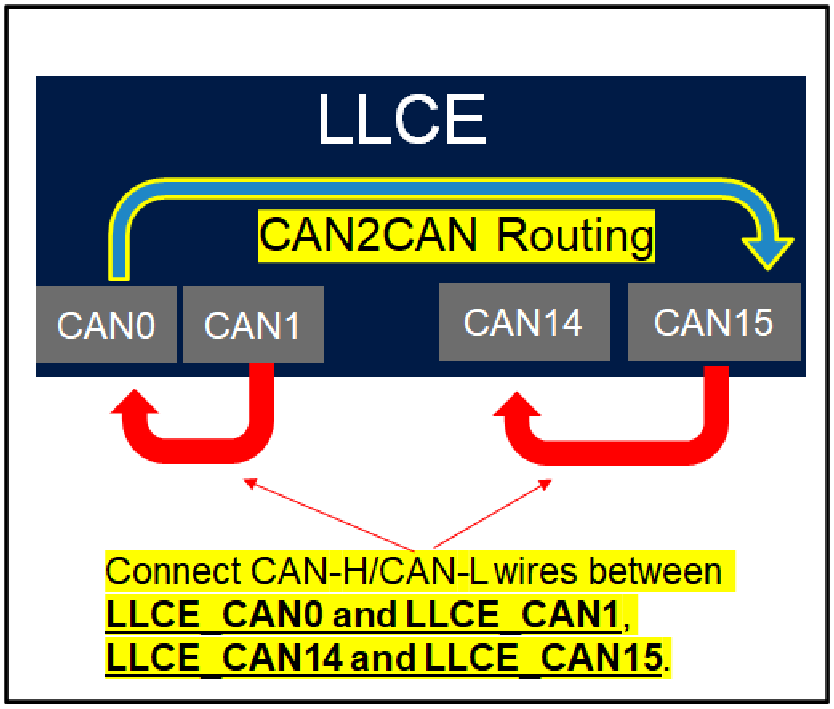
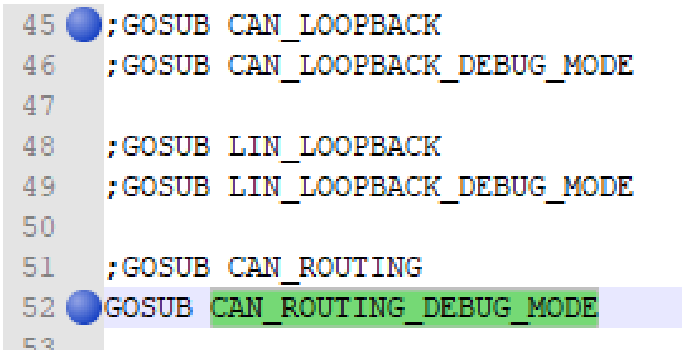


Figure 13.   **Connecting the CAN wires**

You can see the Lauterbach's cmm script "S32G_app_load.cmm" to run the sample app under folder S32G_LLCE_1.0.2\sample_app_llce\llce_sample_app_af\tools\cmm_scripts.

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.



```
;GOSUB CAN_LOOPBACK
;GOSUB CAN_LOOPBACK_DEBUG_MODE

;GOSUB LIN_LOOPBACK
;GOSUB LIN_LOOPBACK_DEBUG_MODE

;GOSUB CAN_ROUTING
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.

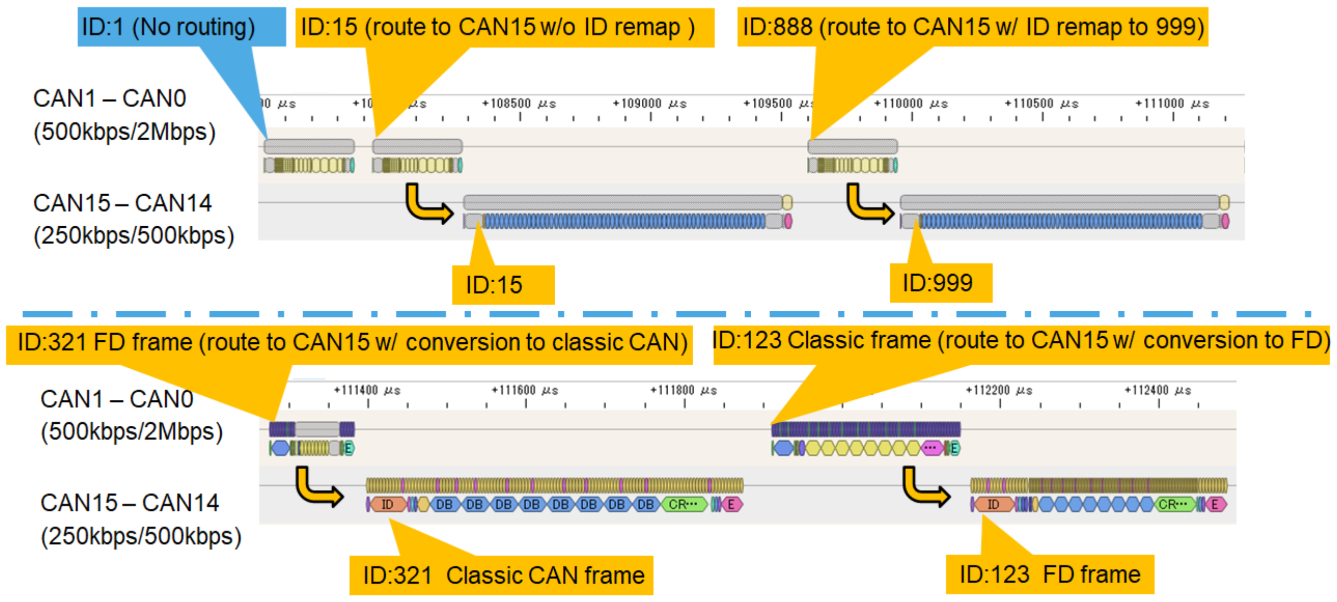

ID:1 (No routing)
ID:15 (route to CAN15 w/o ID remap)
ID:888 (route to CAN15 w/ ID remap to 999)
CAN1 – CAN0 (500kbps/2Mbps)
CAN15 – CAN14 (250kbps/500kbps)
ID:15
ID:999
ID:321 FD frame (route to CAN15 w/ conversion to classic CAN)
ID:123 Classic frame (route to CAN15 w/ conversion to FD)
CAN1 – CAN0 (500kbps/2Mbps)
CAN15 – CAN14 (250kbps/500kbps)
ID:321 Classic CAN frame
ID:123 FD frame

Figure 14. **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. Then, LLCE sends the packet to PFE without host CPU's intervention. Then PFE sends it from PFE_MAC1.

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

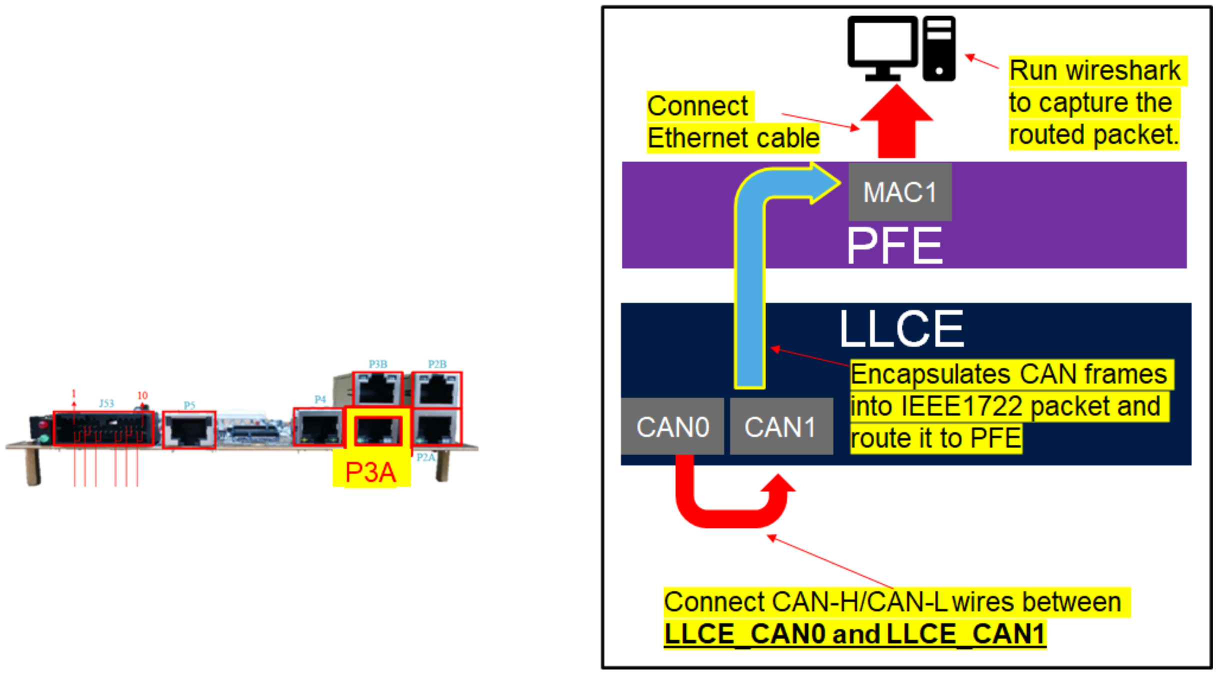
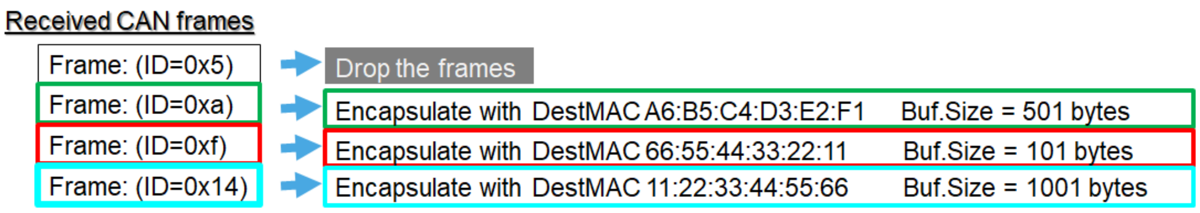

P3A
Run wireshark to capture the routed packet.
Connect Ethernet cable
MAC1
PFE
LLCE
Encapsulates CAN frames into IEEE1722 packet and route it to PFE
CAN0
CAN1
Connect CAN-H/CAN-L wires between LLCE_CAN0 and LLCE_CAN1

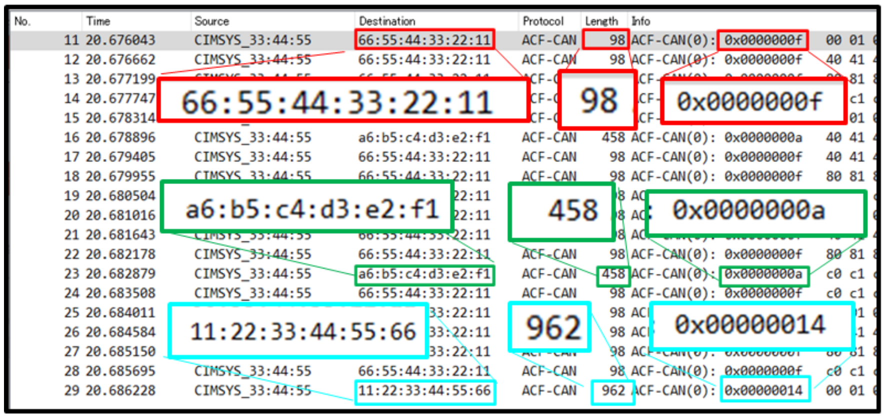
Figure 15. **CAN2ETH routing**

Run the Lauterbach’s cmm script “*run_mcal_pfe.cmm*” under folder S32G_LLCE_1.0.2\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 four kinds of IDs (ID=5,10,15 and 20). With this app’s Tresos config, each CAN frames are processed as shown below.


Received CAN frames
Frame: (ID=0x5)
Drop the frames
Frame: (ID=0xa)
Encapsulate with DestMAC A6:B5:C4:D3:E2:F1 Buf.Size = 501 bytes
Frame: (ID=0xf)
Encapsulate with DestMAC 66:55:44:33:22:11 Buf.Size = 101 bytes
Frame: (ID=0x14)
Encapsulate with DestMAC 11:22:33:44:55:66 Buf.Size = 1001 bytes



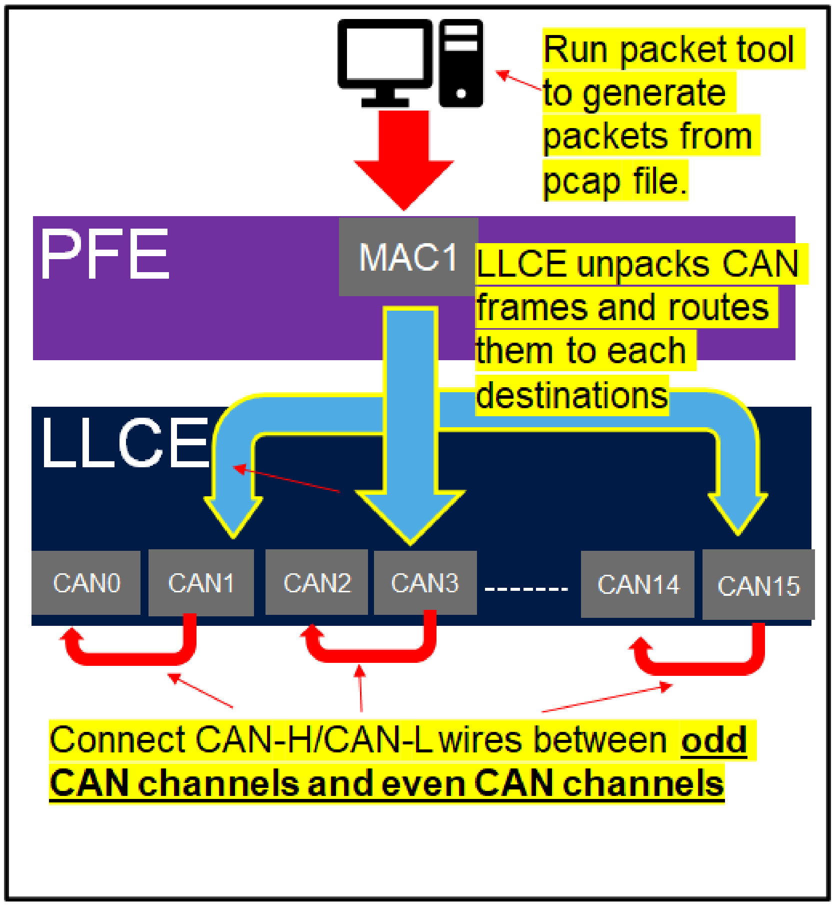
No.	Time	Source	Destination	Protocol	Length	Info
11	20.676043	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f 00 01
12	20.676662	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f 40 41
13	20.677199					81
14	20.677747					c1
15	20.678314					01
16	20.678896	CIMSYS_33:44:55	a6:b5:c4:d3:e2:f1	ACF-CAN	458	ACF-CAN(0): 0x0000000a 40 41
17	20.679405	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f 40 41
18	20.679955	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f 80 81
19	20.680504		22:11			
20	20.681016		22:11			
21	20.681643		22:11			
22	20.682178	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f 80 81
23	20.682879	CIMSYS_33:44:55	a6:b5:c4:d3:e2:f1	ACF-CAN	458	ACF-CAN(0): 0x0000000a c0 c1
24	20.683508	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f c0 c1
25	20.684011		33:22:11		98	
26	20.684584		33:22:11		98	
27	20.685150		33:22:11		98	81
28	20.685695	CIMSYS_33:44:55	66:55:44:33:22:11	ACF-CAN	98	ACF-CAN(0): 0x0000000f c0 c1
29	20.686228	CIMSYS_33:44:55	11:22:33:44:55:66	ACF-CAN	962	ACF-CAN(0): 0x00000014 00 01

You will see the wireshark capture result as shown on the left. It will show the packet contains above encapsulated CAN frames.

Figure 16. **Encapsulated CAN frames**

For ETH2CAN, you can play it with CAN2ETH setup as is. If you connect multiple CAN channels, you can see more routed CAN frames.

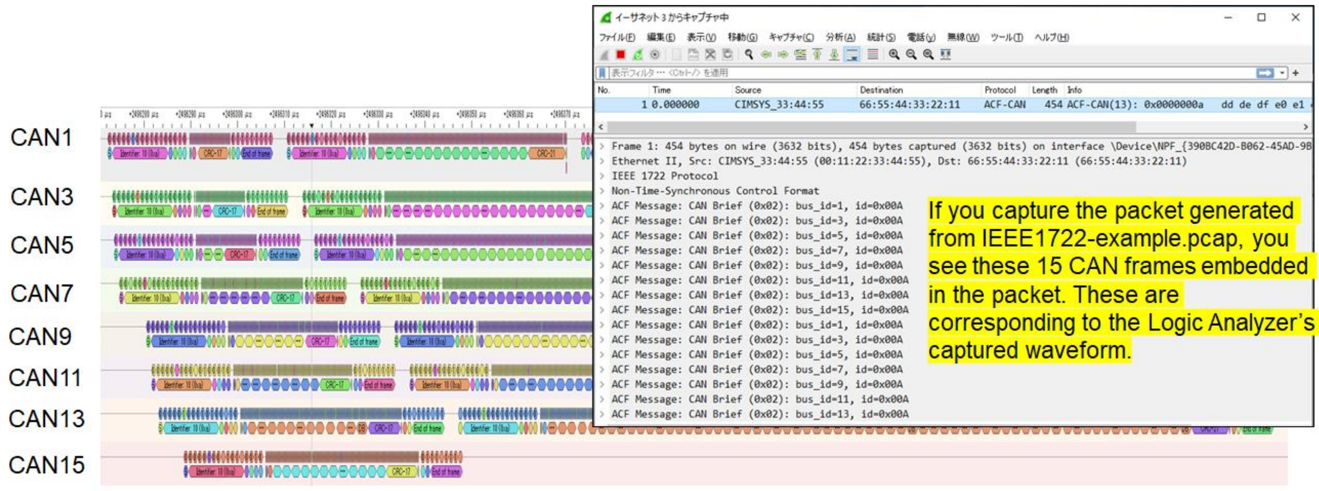
Running the same elf file as CAN2ETH (use same cmm also), if you send the packet generated from bundled PCAP “IEEE1722-example.pcap” to PFE_MAC1, LLCE parses it and unpacks the encapsulated CAN frames to each destination according to the ACF CAN msg information embedded in the packet (i.e. all odd CAN channels). You can play it with the CAN2ETH wiring setup as is. If you connect all odd CAN channels to the companion CAN channels (e.g. even channels), you can observe all unpacked CAN frames which are unpacked from the Ethernet frame.


Run packet tool to generate packets from pcap file.
PFE
MAC1
LLCE unpacks CAN frames and routes them to each destinations
LLCE
CAN0
CAN1
CAN2
CAN3

CAN14
CAN15
Connect CAN-H/CAN-L wires between odd CAN channels and even CAN channels

Figure 17. **ETH2CAN routing**

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


CAN1
CAN3
CAN5
CAN7
CAN9
CAN11
CAN13
CAN15
イーサネット 3 からキャプチャ中
ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(y) 無線(W) ツール(T) ヘルプ(H)
表示フィルタ … <Ctrl-/> を適用

```
No.    Time        Source             Destination          Protocol  Length Info
     1 0.000000    CIMSYS_33:44:55    66:55:44:33:22:11    ACF-CAN      454 ACF-CAN(13): 0x0000000a   dd de df e0 e1
```

```
> Frame 1: 454 bytes on wire (3632 bits), 454 bytes captured (3632 bits) on interface \Device\NPF_{390BC42D-B062-45AD-9B
> Ethernet II, Src: CIMSYS_33:44:55 (00:11:22:33:44:55), Dst: 66:55:44:33:22:11 (66:55:44:33:22:11)
> IEEE 1722 Protocol
> Non-Time-Synchronous Control Format
> ACF Message: CAN Brief (0x02): bus_id=1, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=3, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=5, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=7, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=9, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=11, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=13, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=15, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=1, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=3, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=5, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=7, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=9, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=11, id=0x00A
> ACF Message: CAN Brief (0x02): bus_id=13, id=0x00A
```

If you capture the packet generated from IEEE1722-example.pcap, you see these 15 CAN frames embedded in the packet. These are corresponding to the Logic Analyzer's captured waveform.

Figure 18. **Routed CAN frames**

4. Cnfigurating 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 September 2021. (i.e. S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.exe).

4.1. Importing the sample config

Run EB-Tresos Studio and import sample config.

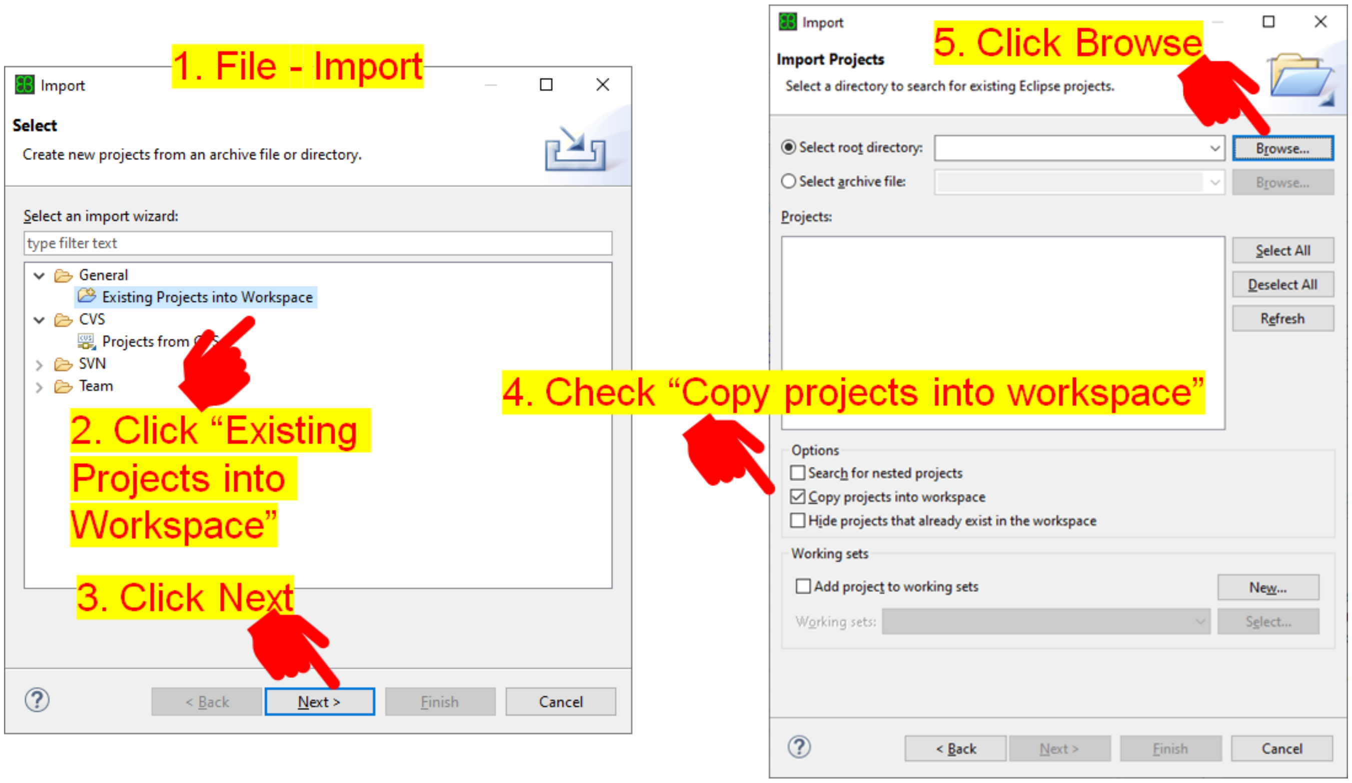

1. File - Import
Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Existing Projects into Workspace
CVS
Projects from CVS
SVN
Team
2. Click "Existing Projects into Workspace"
3. Click Next
< Back
Next >
Finish
Cancel
5. Click Browse
Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Browse...
Projects:
Select All
Deselect All
Refresh
4. Check "Copy projects into workspace"
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel

Figure 19. **Importing file**

In case of CAN2CAN, import the following file. Choose “tresos” and “Tresos_CAN2CAN_Project”.

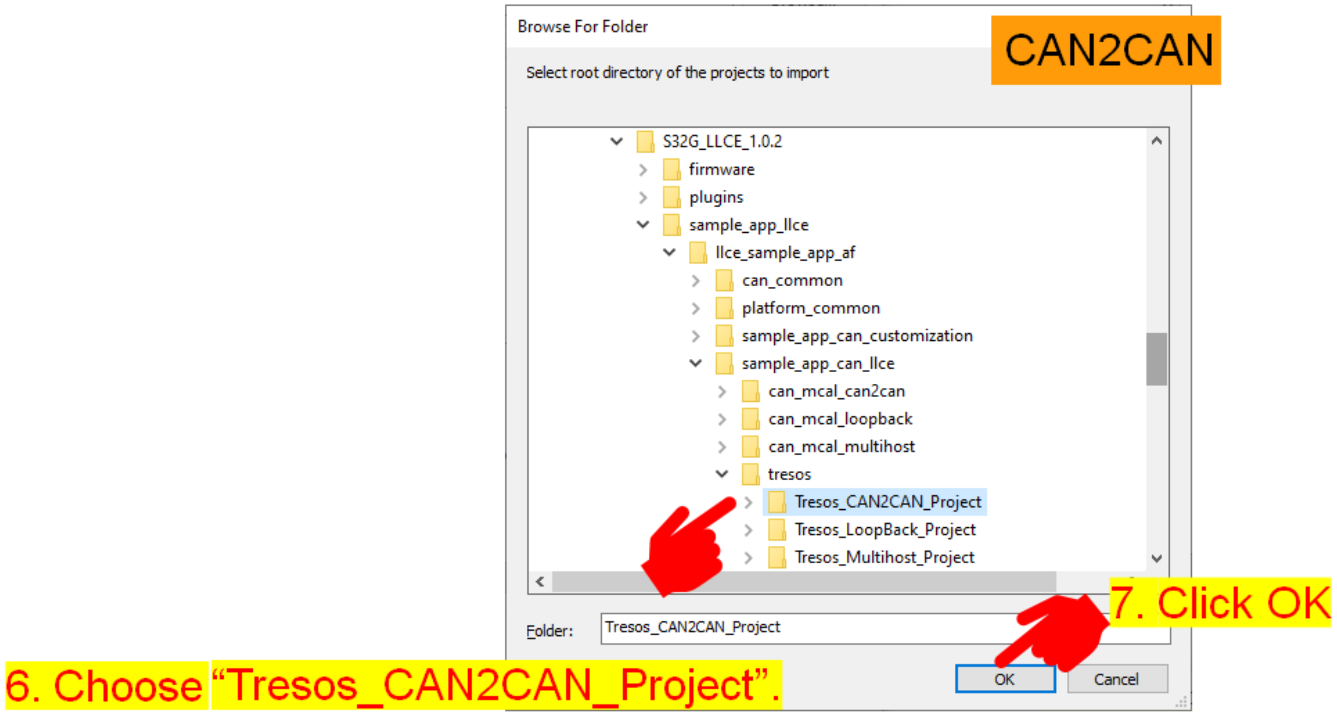


Figure 20. **Importing CAN2CAN config file**

In case of CAN2ETH and ETH2CAN, import the following file.

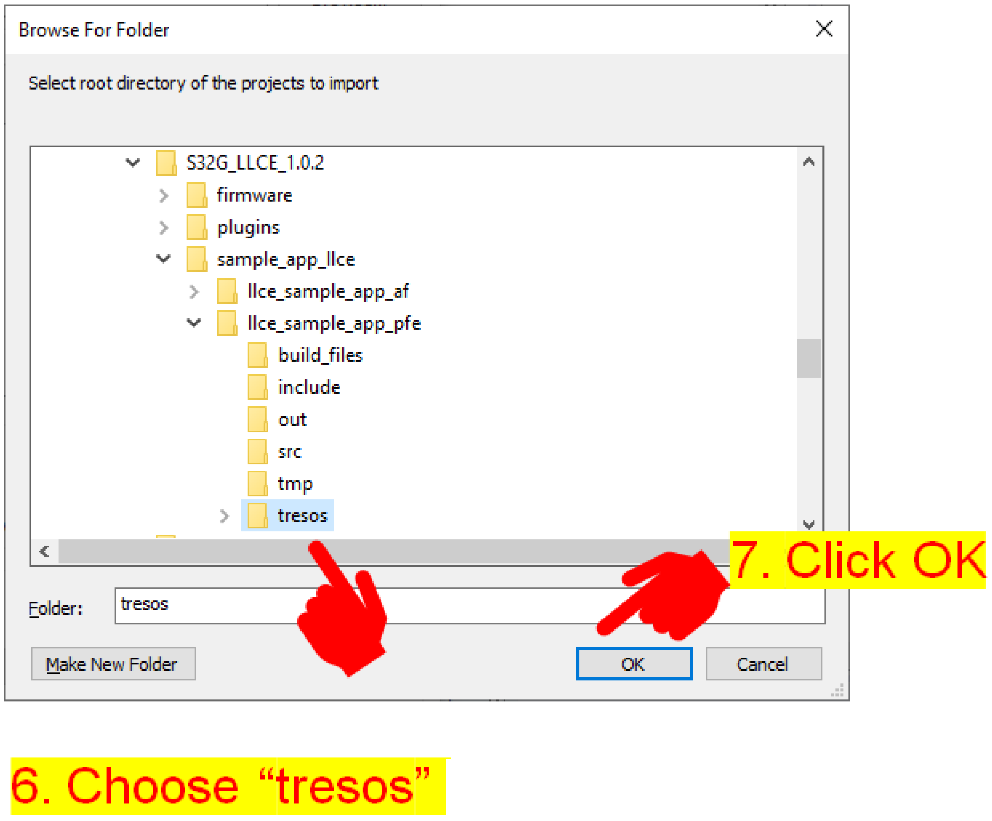
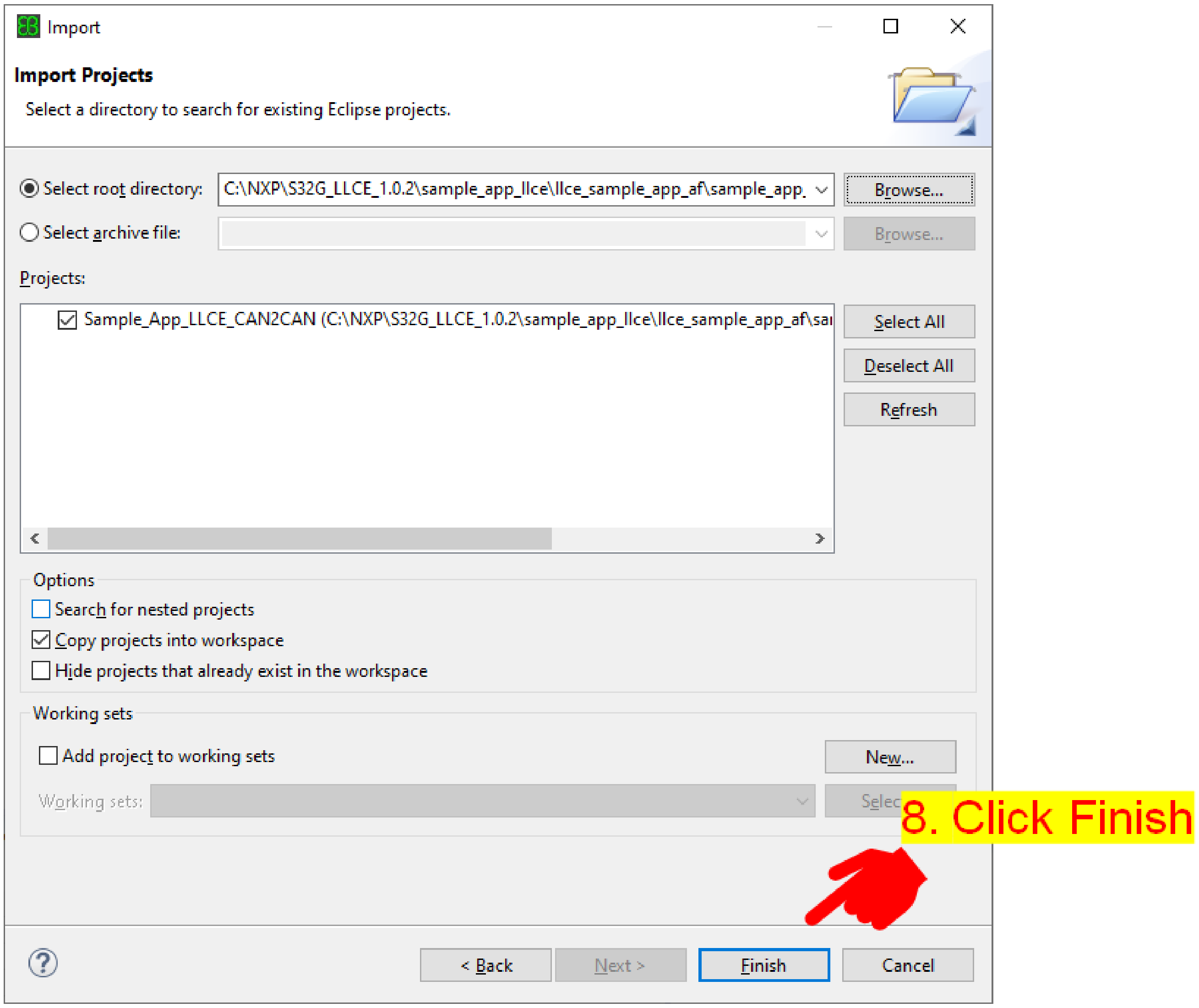


Figure 21. **Importing CAN2ETH and ETH2CAN config file**

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: C:\NXP\S32G_LLCE_1.0.2\sample_app_llce\llce_sample_app_af\sample_app_

Select archive file:

Projects:

☑ Sample_App_LLCE_CAN2CAN (C:\NXP\S32G_LLCE_1.0.2\sample_app_llce\llce_sample_app_af\sa

Options

☐ Search for nested projects

☑ Copy projects into workspace

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 22. **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.

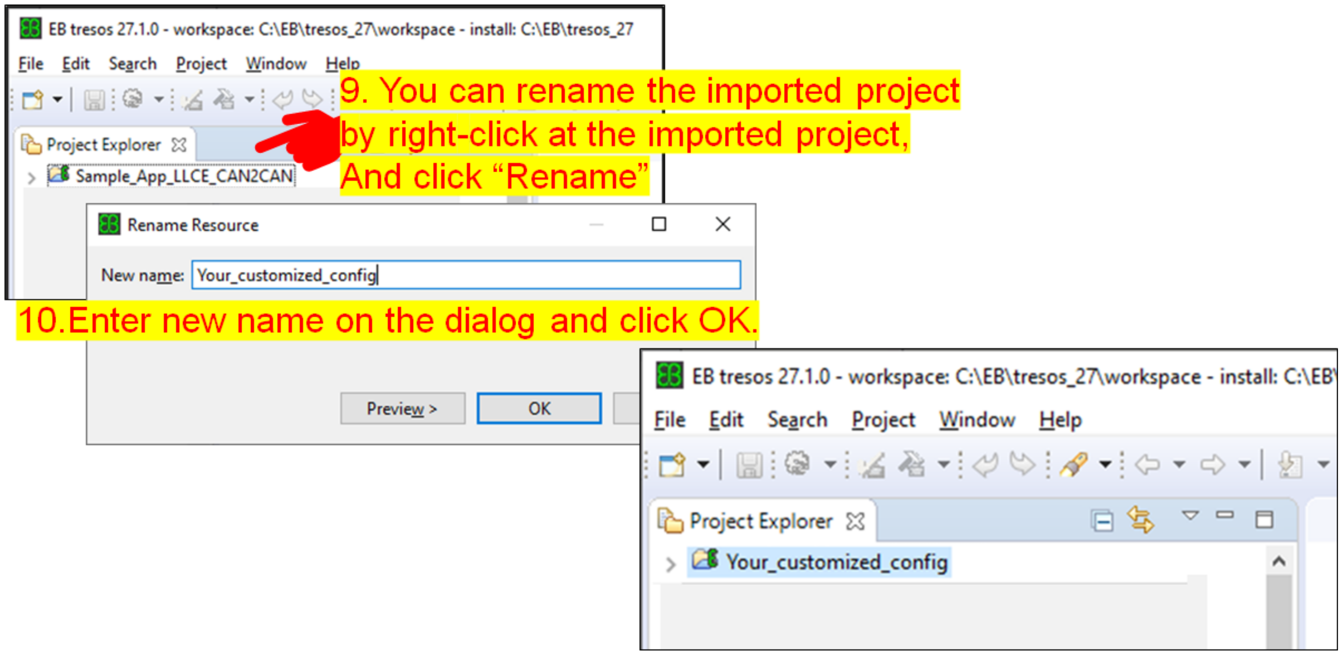

EB tresos 27.1.0 - workspace: C:\EB\tresos_27\workspace - install: C:\EB\tresos_27
File Edit Search Project Window Help
Project Explorer
Sample_App_LLCE_CAN2CAN
9. You can rename the imported project by right-click at the imported project, And click "Rename"
Rename Resource
New name: Your_customized_config
10.Enter new name on the dialog and click OK.
Preview >
OK
Project Explorer
Your_customized_config

Figure 23. **Renaming the imported project**

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

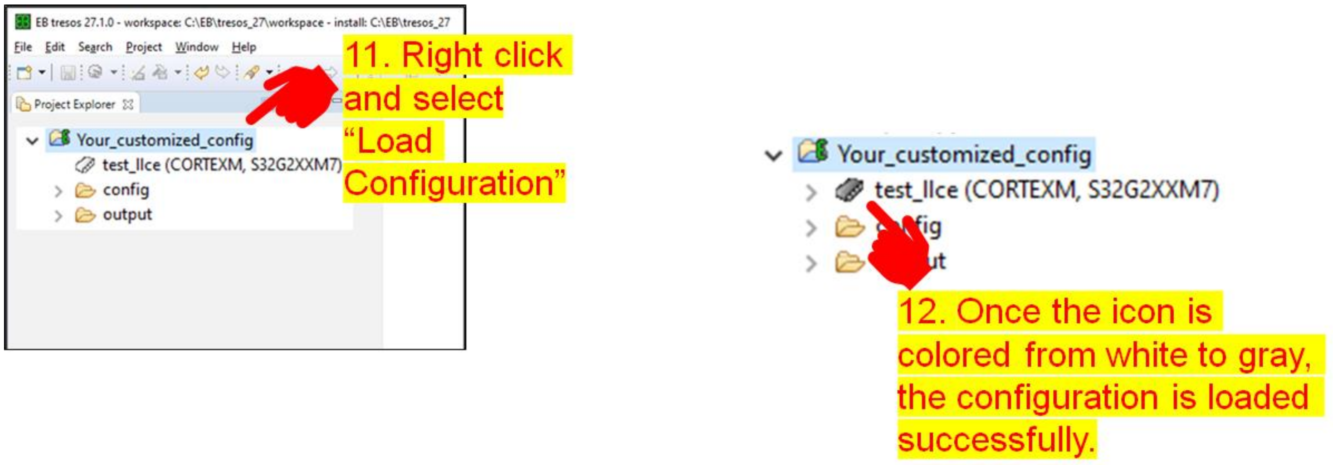
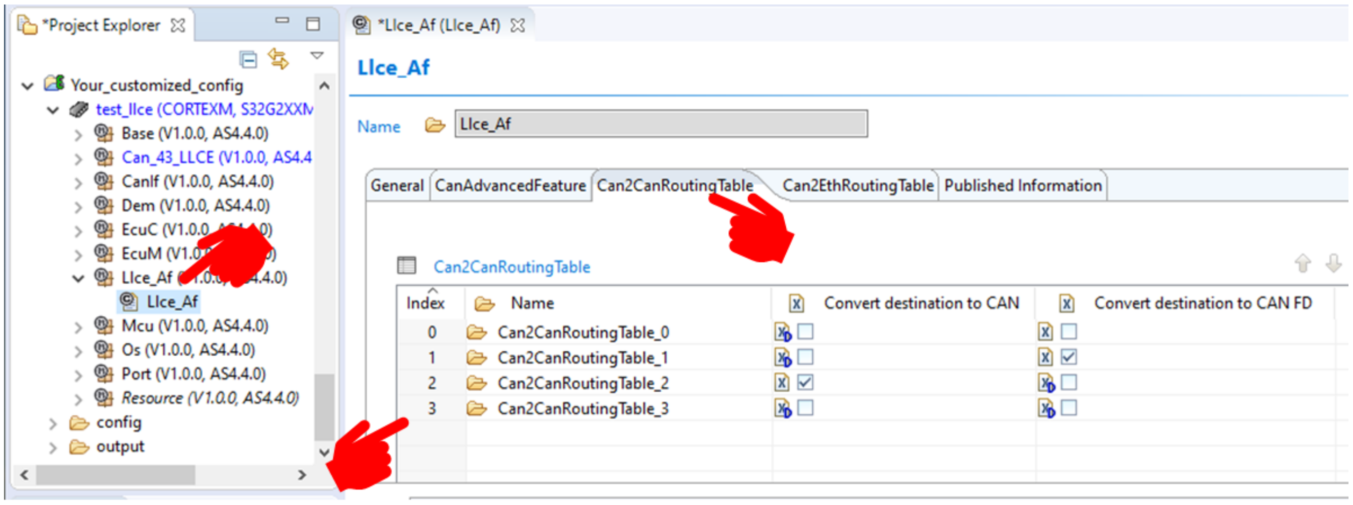

EB tresos 27.1.0 - workspace: C:\EB\tresos_27\workspace - install: C:\EB\tresos_27
File Edit Search Project Window Help
Project Explorer
Your_customized_config
test_llce (CORTEXM, S32G2XXM7)
config
output
11. Right click and select "Load Configuration"
Your_customized_config
test_llce (CORTEXM, S32G2XXM7)
12. Once the icon is colored from white to gray, the configuration is loaded successfully.

Figure 24. **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.



*Project Explorer

- Your_customized_config
  - test_llce (CORTEXM, S32G2XXM
    - Base (V1.0.0, AS4.4.0)
    - Can_43_LLCE (V1.0.0, AS4.4
    - CanIf (V1.0.0, AS4.4.0)
    - Dem (V1.0.0, AS4.4.0)
    - EcuC (V1.0.0, AS4.4.0)
    - EcuM (V1.0.0, AS4.4.0)
    - Llce_Af (V1.0.0, AS4.4.0)
      - Llce_Af
    - Mcu (V1.0.0, AS4.4.0)
    - Os (V1.0.0, AS4.4.0)
    - Port (V1.0.0, AS4.4.0)
    - Resource (V1.0.0, AS4.4.0)
  - config
  - output

*Llce_Af (Llce_Af)

Llce_Af

Name: Llce_Af

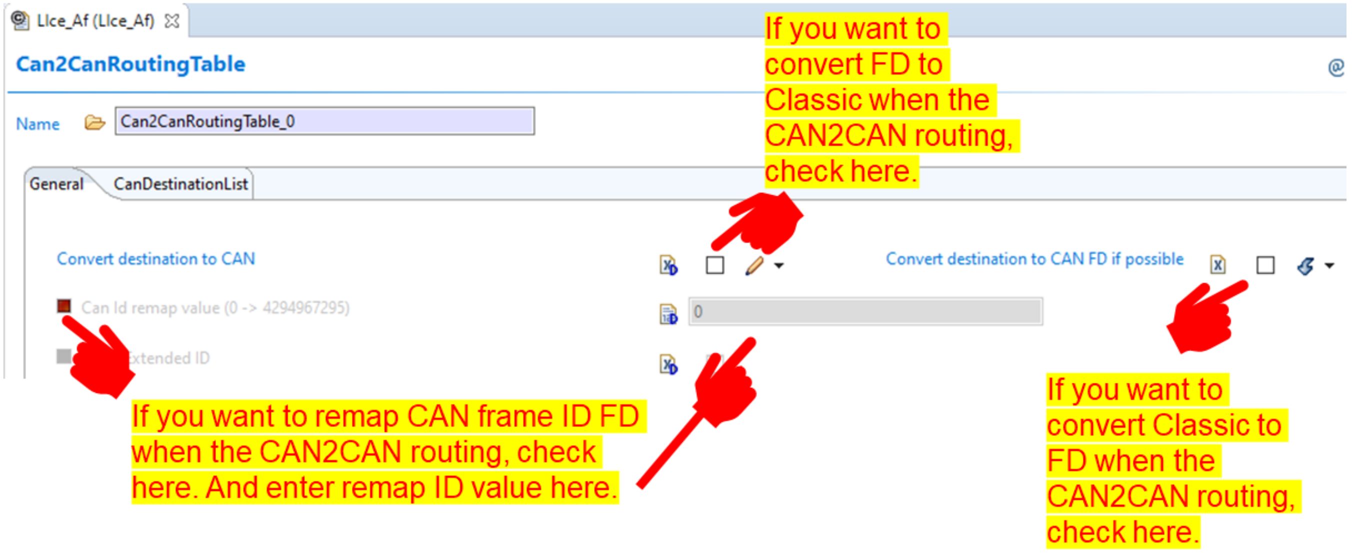
General | CanAdvancedFeature | Can2CanRoutingTable | Can2EthRoutingTable | Published Information

Can2CanRoutingTable

Index	Name	Convert destination to CAN	Convert destination to CAN FD
0	Can2CanRoutingTable_0	☐	☐
1	Can2CanRoutingTable_1	☐	✓
2	Can2CanRoutingTable_2	✓	☐
3	Can2CanRoutingTable_3	☐	☐

Figure 25. **Configuring Llce_Af**

Configure routing details in General Tab.



Llce_Af (Llce_Af)

Can2CanRoutingTable

Name: Can2CanRoutingTable_0

General | CanDestinationList

Convert destination to CAN ☐

Can Id remap value (0 -> 4294967295): 0

Extended ID

Convert destination to CAN FD if possible ☐

If you want to convert FD to Classic when the CAN2CAN routing, check here.

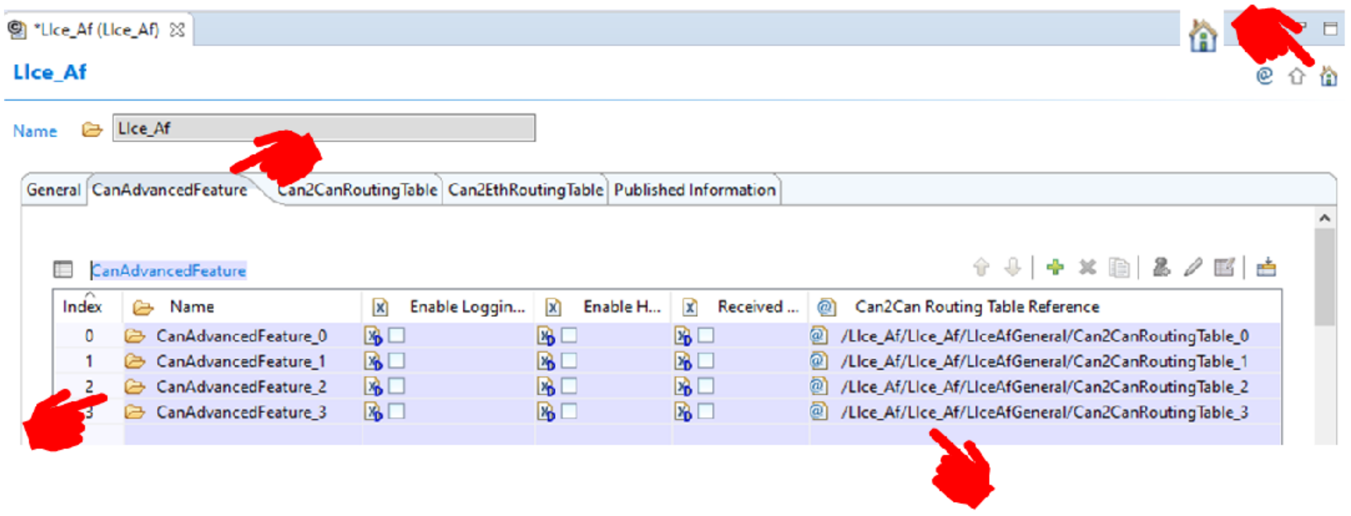
If you want to remap CAN frame ID FD when the CAN2CAN routing, check here. And enter remap ID value here.

If you want to convert Classic to FD when the CAN2CAN routing, check here.

Figure 26. **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.



*Llce_Af (Llce_Af)

Llce_Af

Name: Llce_Af

General | CanAdvancedFeature | Can2CanRoutingTable | Can2EthRoutingTable | Published Information

CanAdvancedFeature

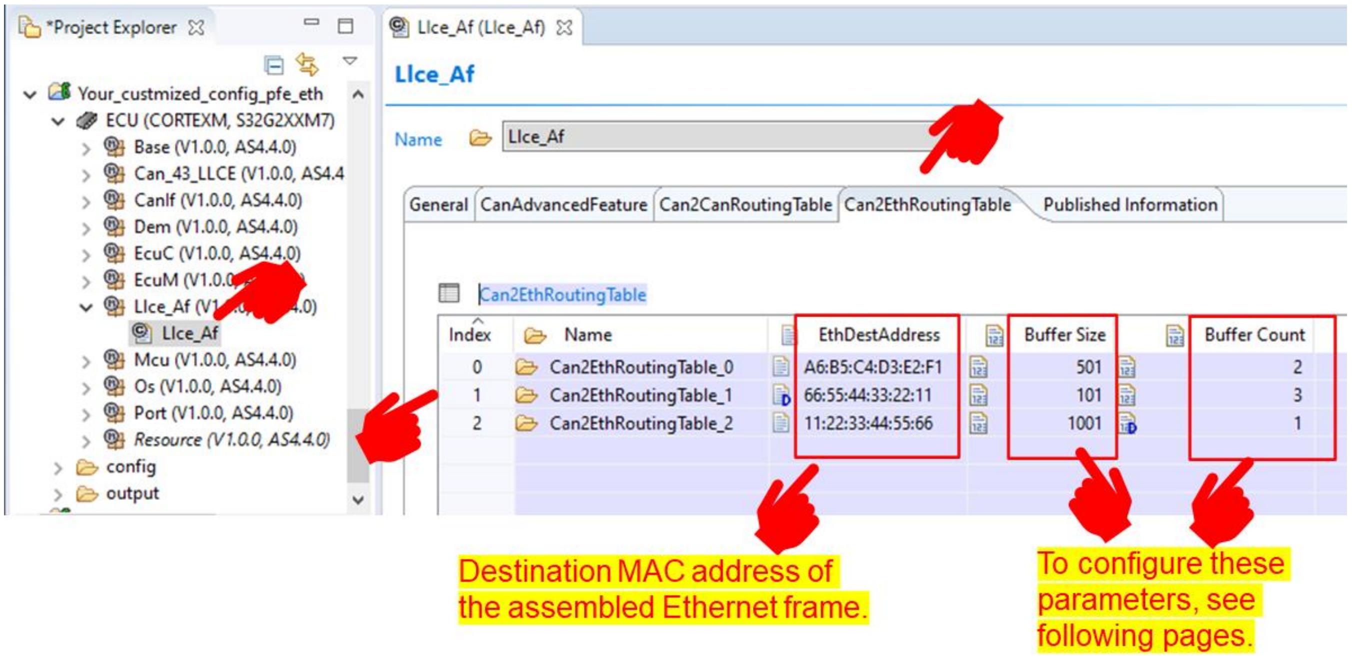
Index	Name	Enable Loggin...	Enable H...	Received ...	Can2Can Routing Table Reference
0	CanAdvancedFeature_0	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_0
1	CanAdvancedFeature_1	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_1
2	CanAdvancedFeature_2	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_2
3	CanAdvancedFeature_3	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_3

Figure 27. **CAN advanced feature table**

4.3. Configuring LIce_Af for CAN2ETH

In Configure Can2EthRoutingTable follow the steps to configure LIce_Af.

1. Double click LIce_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.



*Project Explorer

- Your_custmized_config_pfe_eth
  - ECU (CORTEXM, S32G2XXM7)
    - Base (V1.0.0, AS4.4.0)
    - Can_43_LLCE (V1.0.0, AS4.4
    - CanIf (V1.0.0, AS4.4.0)
    - Dem (V1.0.0, AS4.4.0)
    - EcuC (V1.0.0, AS4.4.0)
    - EcuM (V1.0.0, AS4.4.0)
    - Llce_Af (V1.0.0, AS4.4.0)
      - Llce_Af
    - Mcu (V1.0.0, AS4.4.0)
    - Os (V1.0.0, AS4.4.0)
    - Port (V1.0.0, AS4.4.0)
    - *Resource (V1.0.0, AS4.4.0)*
  - config
  - output

Llce_Af (Llce_Af)

Llce_Af

Name: Llce_Af

General | CanAdvancedFeature | Can2CanRoutingTable | Can2EthRoutingTable | Published Information

Can2EthRoutingTable

Index	Name	EthDestAddress	Buffer Size	Buffer Count
0	Can2EthRoutingTable_0	A6:B5:C4:D3:E2:F1	501	2
1	Can2EthRoutingTable_1	66:55:44:33:22:11	101	3
2	Can2EthRoutingTable_2	11:22:33:44:55:66	1001	1

Destination MAC address of the assembled Ethernet frame.
To configure these parameters, see following pages.

Figure 28. **Configuring LIce_Af**

Use the following formula to calculate the buffer size.

If the customer want to pack ***N*** ACF msg / packet, the Bufer size should be equal or larger than

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

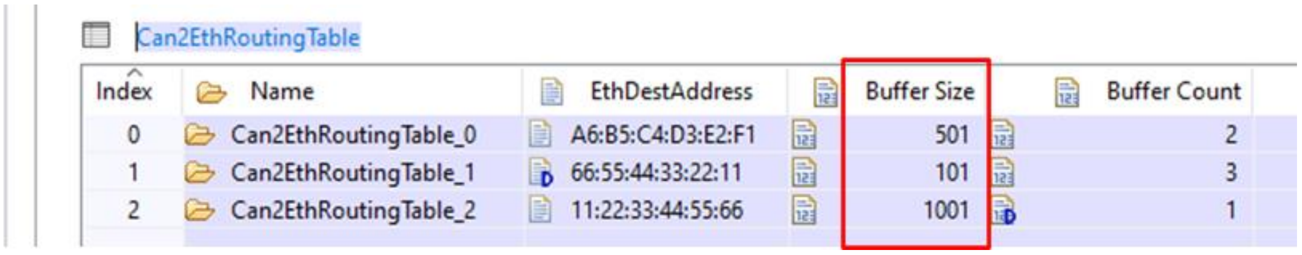
and less than

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

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), 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).



Can2EthRoutingTable

Index	Name	EthDestAddress	Buffer Size	Buffer Count
0	Can2EthRoutingTable_0	A6:B5:C4:D3:E2:F1	501	2
1	Can2EthRoutingTable_1	66:55:44:33:22:11	101	3
2	Can2EthRoutingTable_2	11:22:33:44:55:66	1001	1

Figure 29. **Calculating buffer size**

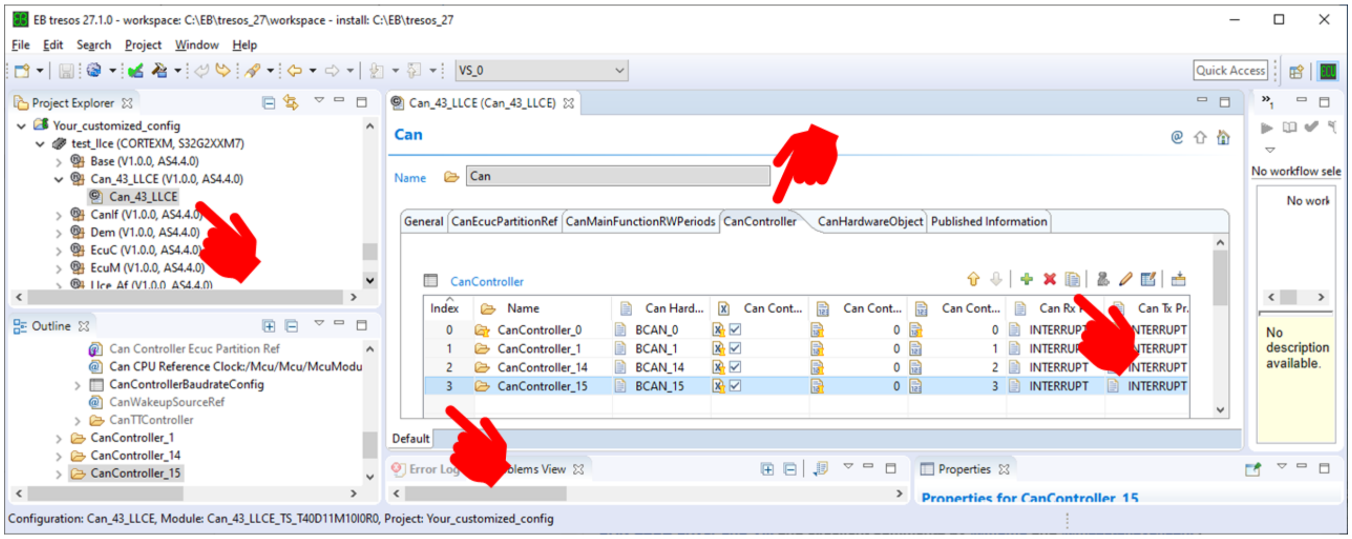
The buffer count depends on a multitude of factors. It is not that easy to calculate exact values without some experimentation.

- 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. 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.


EB tresos 27.1.0 - workspace: C:\EB\tresos_27\workspace - install: C:\EB\tresos_27

Can

Name: Can

General | CanEcucPartitionRef | CanMainFunctionRWPeriods | CanController | CanHardwareObject | Published Information

CanController

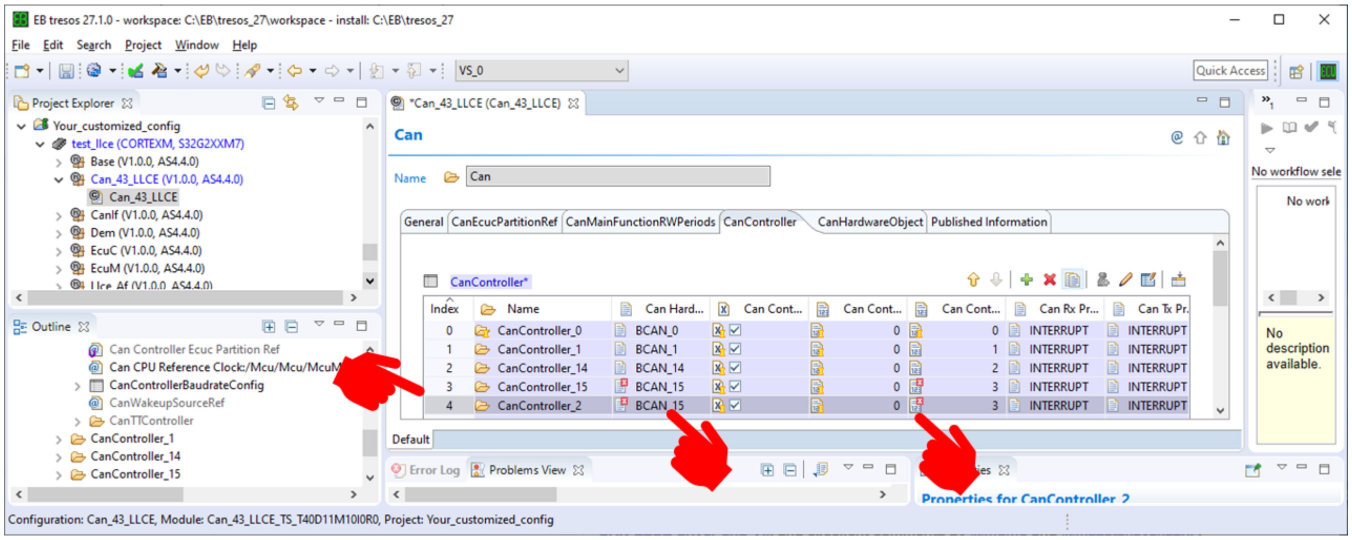
Index	Name	Can Hard...	Can Cont...	Can Cont...	Can Cont...	Can Rx P...	Can Tx Pr...
0	CanController_0	BCAN_0	✓	0	0	INTERRUPT	INTERRUPT
1	CanController_1	BCAN_1	✓	0	1	INTERRUPT	INTERRUPT
2	CanController_14	BCAN_14	✓	0	2	INTERRUPT	INTERRUPT
3	CanController_15	BCAN_15	✓	0	3	INTERRUPT	INTERRUPT

Properties for CanController_15

Configuration: Can_43_LLCE, Module: Can_43_LLCE_TS_T40D11M10I0R0, Project: Your_customized_config

Figure 30.   **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.


EB tresos 27.1.0 - workspace: C:\EB\tresos_27\workspace - install: C:\EB\tresos_27

Can

Name: Can

General | CanEcucPartitionRef | CanMainFunctionRWPeriods | CanController | CanHardwareObject | Published Information

CanController*

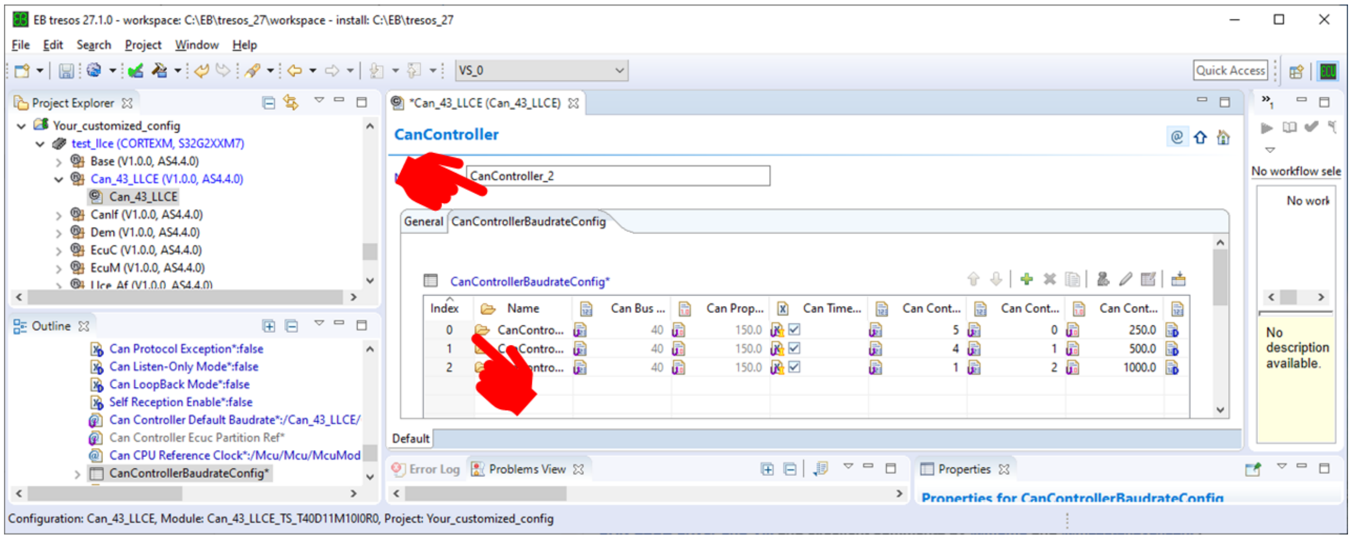
Index	Name	Can Hard...	Can Cont...	Can Cont...	Can Cont...	Can Rx Pr...	Can Tx Pr...
0	CanController_0	BCAN_0	✓	0	0	INTERRUPT	INTERRUPT
1	CanController_1	BCAN_1	✓	0	1	INTERRUPT	INTERRUPT
2	CanController_14	BCAN_14	✓	0	2	INTERRUPT	INTERRUPT
3	CanController_15	BCAN_15	✓	0	3	INTERRUPT	INTERRUPT
4	CanController_2	BCAN_15	✓	0	3	INTERRUPT	INTERRUPT

Properties for CanController_2

Configuration: Can_43_LLCE, Module: Can_43_LLCE_TS_T40D11M10I0R0, Project: Your_customized_config

Figure 31.   **Configuring BCAN (two)**

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


EB tresos 27.1.0 - workspace: C:\EB\tresos_27\workspace - install: C:\EB\tresos_27
File Edit Search Project Window Help
VS_0
Quick Access
Project Explorer
Outline
Error Log
Problems View
Properties
No workflow sele
No work
No description available.
Default

Your_customized_config
- test_llce (CORTEXM, S32G2XXM7)
  - Base (V1.0.0, AS4.4.0)
  - Can_43_LLCE (V1.0.0, AS4.4.0)
    - Can_43_LLCE
  - CanIf (V1.0.0, AS4.4.0)
  - Dem (V1.0.0, AS4.4.0)
  - EcuC (V1.0.0, AS4.4.0)
  - EcuM (V1.0.0, AS4.4.0)
  - Llce_Af (V1.0.0, AS4.4.0)

Outline:
- Can Protocol Exception*:false
- Can Listen-Only Mode*:false
- Can LoopBack Mode*:false
- Self Reception Enable*:false
- Can Controller Default Baudrate*:/Can_43_LLCE/
- Can Controller Ecuc Partition Ref*
- Can CPU Reference Clock*:/Mcu/Mcu/McuMod
- CanControllerBaudrateConfig*

*Can_43_LLCE (Can_43_LLCE)

CanController

CanController_2

General | CanControllerBaudrateConfig

CanControllerBaudrateConfig*

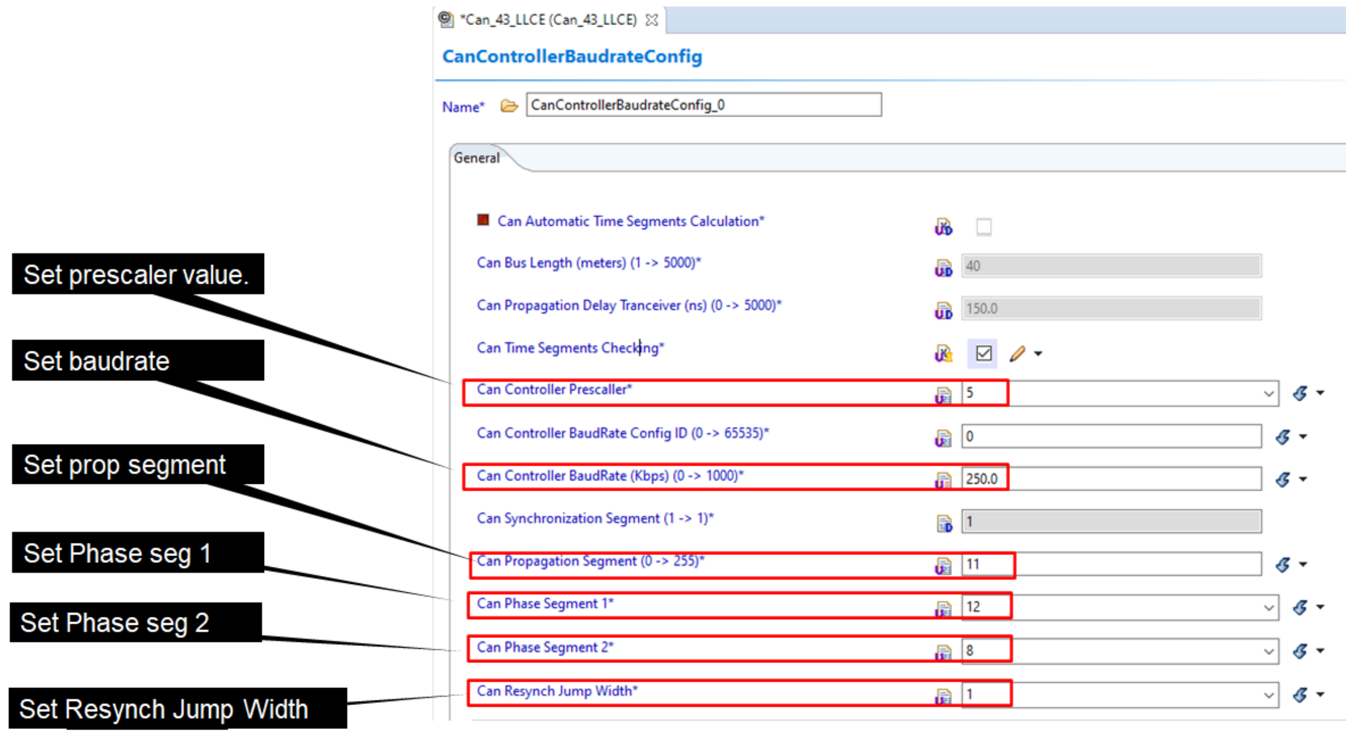
Index	Name	Can Bus ...	Can Prop...	Can Time...	Can Cont...	Can Cont...	Can Cont...
0	CanContro...	40	150.0	☑	5	0	250.0
1	CanContro...	40	150.0	☑	4	1	500.0
2	[illegible]ntro...	40	150.0	☑	1	2	1000.0

Properties for CanControllerBaudrateConfig

Configuration: Can_43_LLCE, Module: Can_43_LLCE_TS_T40D11M10I0R0, Project: Your_customized_config

Figure 32. **Configuring BCAN (three)**

10. Configure baud rate parameters.



*Can_43_LLCE (Can_43_LLCE)

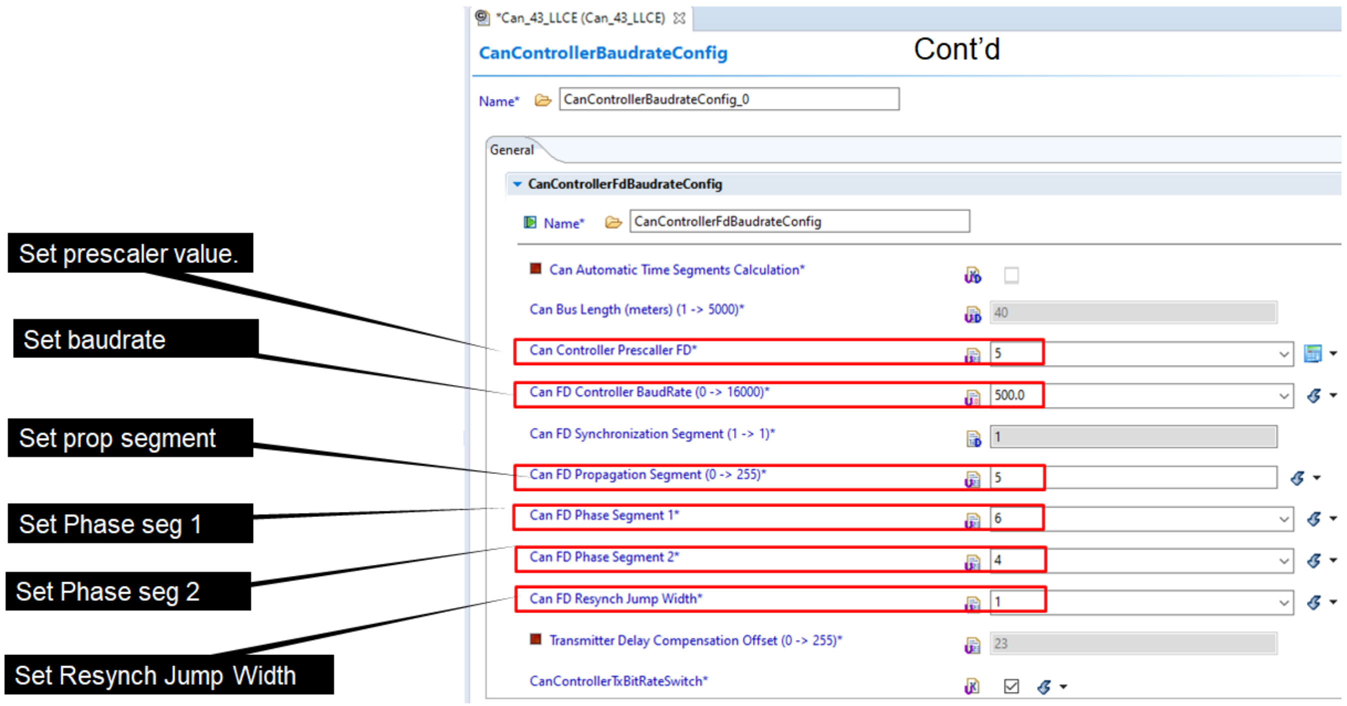
CanControllerBaudrateConfig

Name* CanControllerBaudrateConfig_0

General

Can Automatic Time Segments Calculation*	☐
Can Bus Length (meters) (1 -> 5000)*	40
Can Propagation Delay Tranceiver (ns) (0 -> 5000)*	150.0
Can Time Segments Checking*	☑
Can Controller Prescaller*	5
Can Controller BaudRate Config ID (0 -> 65535)*	0
Can Controller BaudRate (Kbps) (0 -> 1000)*	250.0
Can Synchronization Segment (1 -> 1)*	1
Can Propagation Segment (0 -> 255)*	11
Can Phase Segment 1*	12
Can Phase Segment 2*	8
Can Resynch Jump Width*	1

Figure 33. **Baud rate setting**



CanControllerBaudrateConfig

Name* CanControllerBaudrateConfig_0

CanControllerFdBaudrateConfig

Name* CanControllerFdBaudrateConfig

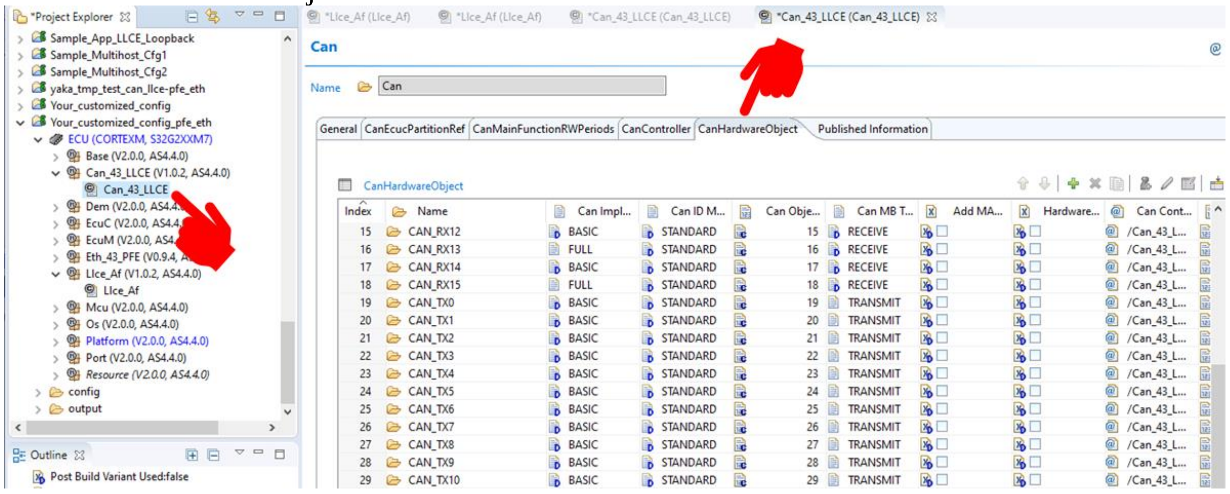
Can Automatic Time Segments Calculation*	☐
Can Bus Length (meters) (1 -> 5000)*	40
Can Controller Prescaller FD*	5
Can FD Controller BaudRate (0 -> 16000)*	500.0
Can FD Synchronization Segment (1 -> 1)*	1
Can FD Propagation Segment (0 -> 255)*	5
Can FD Phase Segment 1*	6
Can FD Phase Segment 2*	4
Can FD Resynch Jump Width*	1
Transmitter Delay Compensation Offset (0 -> 255)*	23
CanControllerTxBitRateSwitch*	☑

Figure 34. **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.



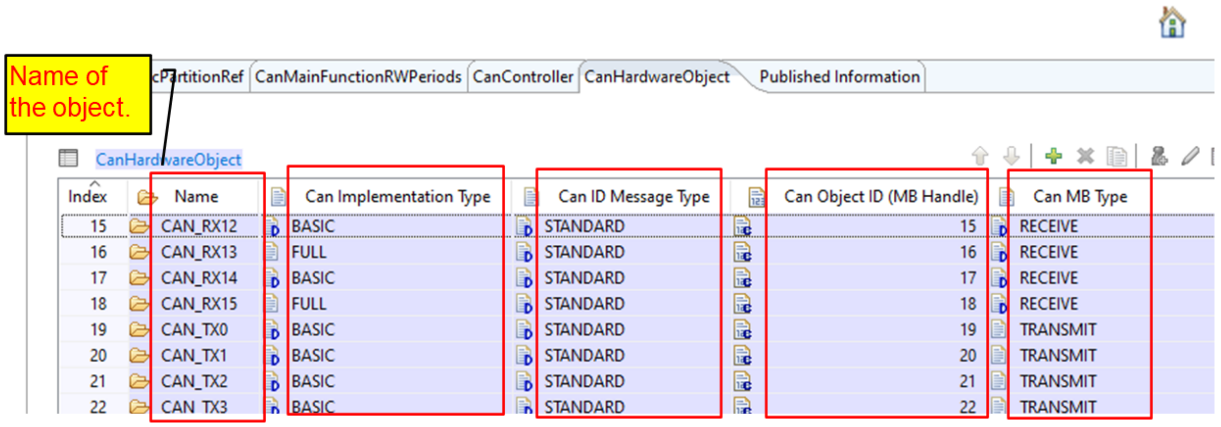
Name Can

CanHardwareObject

Index	Name	Can Impl...	Can ID M...	Can Obje...	Can MB T...	Add MA...	Hardware...	Can Cont...
15	CAN_RX12	BASIC	STANDARD	15	RECEIVE	☐	☐	/Can_43_L...
16	CAN_RX13	FULL	STANDARD	16	RECEIVE	☐	☐	/Can_43_L...
17	CAN_RX14	BASIC	STANDARD	17	RECEIVE	☐	☐	/Can_43_L...
18	CAN_RX15	FULL	STANDARD	18	RECEIVE	☐	☐	/Can_43_L...
19	CAN_TX0	BASIC	STANDARD	19	TRANSMIT	☐	☐	/Can_43_L...
20	CAN_TX1	BASIC	STANDARD	20	TRANSMIT	☐	☐	/Can_43_L...
21	CAN_TX2	BASIC	STANDARD	21	TRANSMIT	☐	☐	/Can_43_L...
22	CAN_TX3	BASIC	STANDARD	22	TRANSMIT	☐	☐	/Can_43_L...
23	CAN_TX4	BASIC	STANDARD	23	TRANSMIT	☐	☐	/Can_43_L...
24	CAN_TX5	BASIC	STANDARD	24	TRANSMIT	☐	☐	/Can_43_L...
25	CAN_TX6	BASIC	STANDARD	25	TRANSMIT	☐	☐	/Can_43_L...
26	CAN_TX7	BASIC	STANDARD	26	TRANSMIT	☐	☐	/Can_43_L...
27	CAN_TX8	BASIC	STANDARD	27	TRANSMIT	☐	☐	/Can_43_L...
28	CAN_TX9	BASIC	STANDARD	28	TRANSMIT	☐	☐	/Can_43_L...
29	CAN_TX10	BASIC	STANDARD	29	TRANSMIT	☐	☐	/Can_43_L...

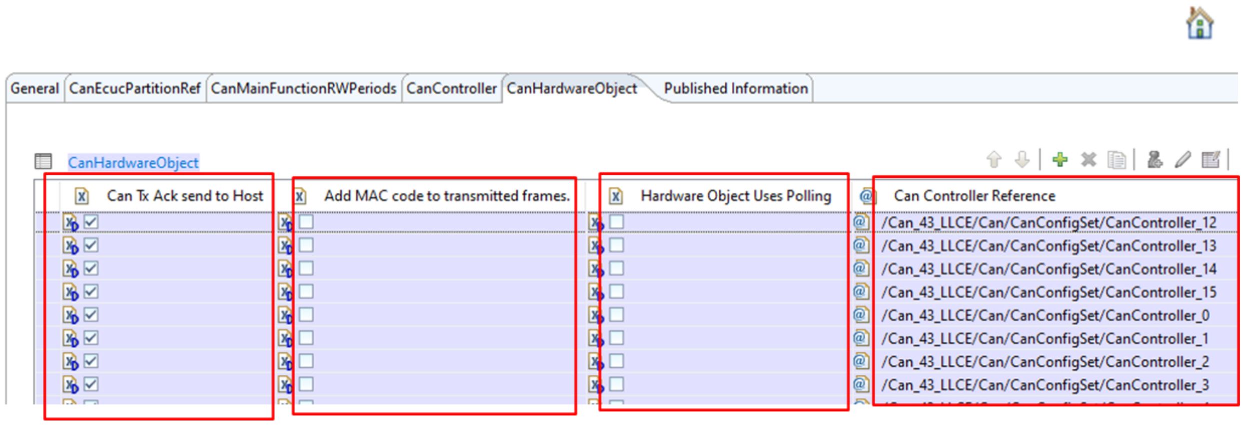
3. Select ID mask BASIC : ID mask enabled. FULL: Exact ID match.
4. Select CAN frame ID type STANDARD / EXTENDED.
5. Object Handle ID. Should start with 0 and continue without any gaps.

6. Select MB Type. RX or TX.



Index	Name	Can Implementation Type	Can ID Message Type	Can Object ID (MB Handle)	Can MB Type
15	CAN_RX12	BASIC	STANDARD	15	RECEIVE
16	CAN_RX13	FULL	STANDARD	16	RECEIVE
17	CAN_RX14	BASIC	STANDARD	17	RECEIVE
18	CAN_RX15	FULL	STANDARD	18	RECEIVE
19	CAN_TX0	BASIC	STANDARD	19	TRANSMIT
20	CAN_TX1	BASIC	STANDARD	20	TRANSMIT
21	CAN_TX2	BASIC	STANDARD	21	TRANSMIT
22	CAN_TX3	BASIC	STANDARD	22	TRANSMIT

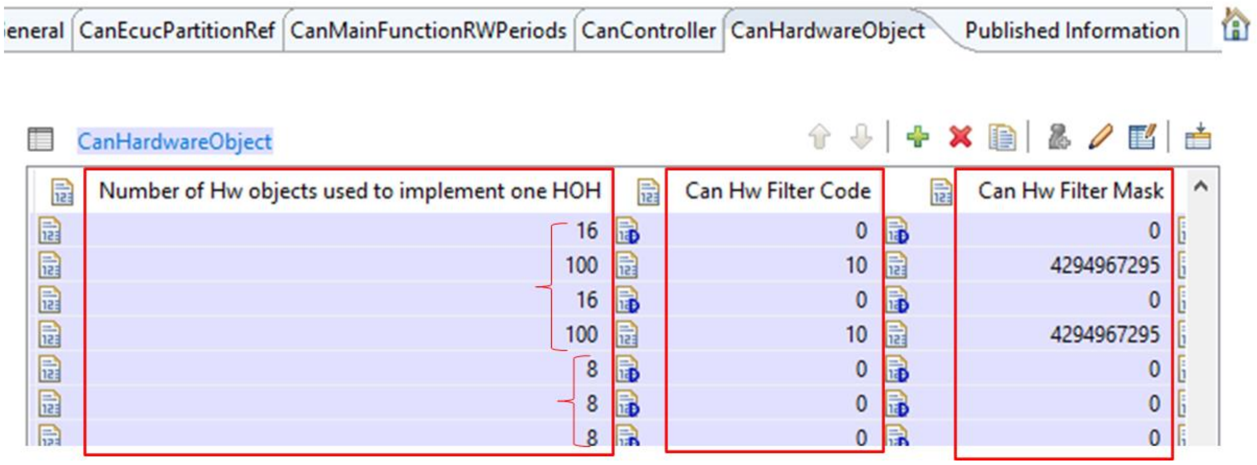
7. Enable feature that sends ack to host.
8. MAC feature: Not available for standard enablement FW.
9. Enables polling of the object.
10. Specify which CanController has the object



Can Tx Ack send to Host	Add MAC code to transmitted frames.	Hardware Object Uses Polling	Can Controller Reference
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_12
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_13
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_14
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_15
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_0
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_1
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_2
✓	☐	☐	/Can_43_LLCE/Can/CanConfigSet/CanController_3

11. 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.
12. For HRH, 2016 objects (i.e. msg.buf.) can be used in total.
13. For HTH, each can controller has 16 objects (I,e, msg buf). The value configured here is for Tx from host. Remaining 8 objects are for CAN2CAN and ETH2CAN.
14. Specify (together with the filter mask) the frame ID that passes the hardware filter for the RX object.

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

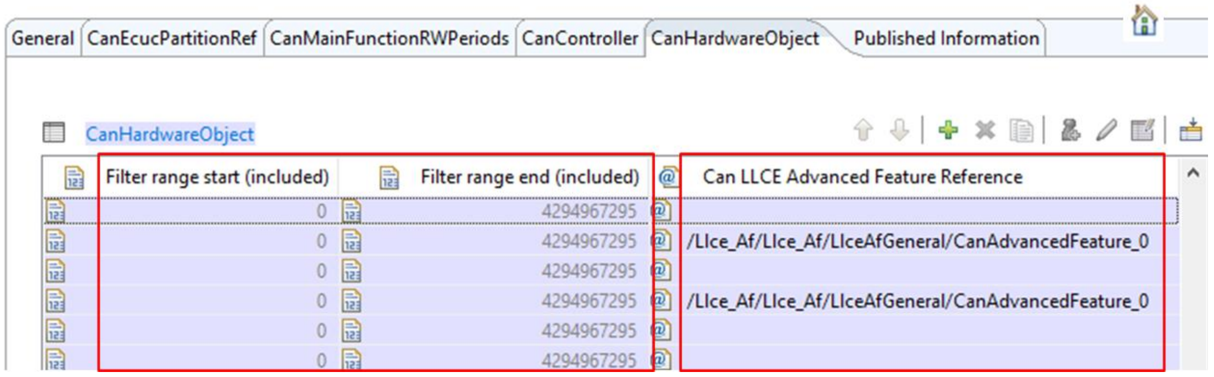


eneral | CanEcucPartitionRef | CanMainFunctionRWPeriods | CanController | CanHardwareObject | Published Information

CanHardwareObject

Number of Hw objects used to implement one HOH	Can Hw Filter Code	Can Hw Filter Mask
16	0	0
100	10	4294967295
16	0	0
100	10	4294967295
8	0	0
8	0	0
8	0	0

16. 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.
17. 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.



General | CanEcucPartitionRef | CanMainFunctionRWPeriods | CanController | CanHardwareObject | Published Information

CanHardwareObject

Filter range start (included)	Filter range end (included)	Can LLCE Advanced Feature Reference
0	4294967295	
0	4294967295	/Llce_Af/Llce_Af/LlceAfGeneral/CanAdvancedFeature_0
0	4294967295	
0	4294967295	/Llce_Af/Llce_Af/LlceAfGeneral/CanAdvancedFeature_0
0	4294967295	
0	4294967295	

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

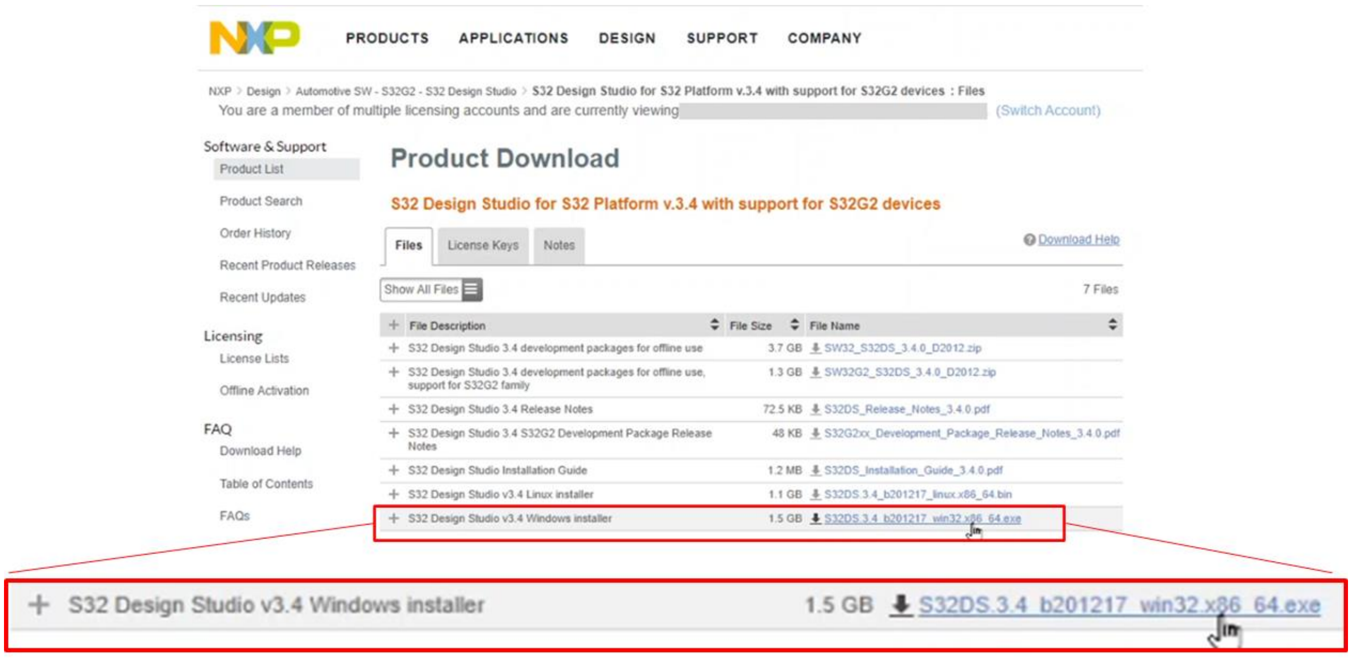
This section is based on the sample app config of the latest release as of September 2021. (i.e. S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.exe).

5.1. Installing S32DS 3.4, RTD and LLCE drivers

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

- S32 Design Studio v3.4 installer
- S32 Design Studio 3.4 Update 1, support for S32G2 Family
- S32G2 Real Time Drivers Version 2.0.0 HF02 Update Site
- S32G_LLCE_GATEWAY_1.0.2_HF2_D2105 Update Site

Go Flexera, download the S32DS3.4 installer and install it.



NXP > Design > Automotive SW - S32G2 - S32 Design Studio > S32 Design Studio for S32 Platform v.3.4 with support for S32G2 devices : Files

You are a member of multiple licensing accounts and are currently viewing [redacted] (Switch Account)

Product Download

S32 Design Studio for S32 Platform v.3.4 with support for S32G2 devices

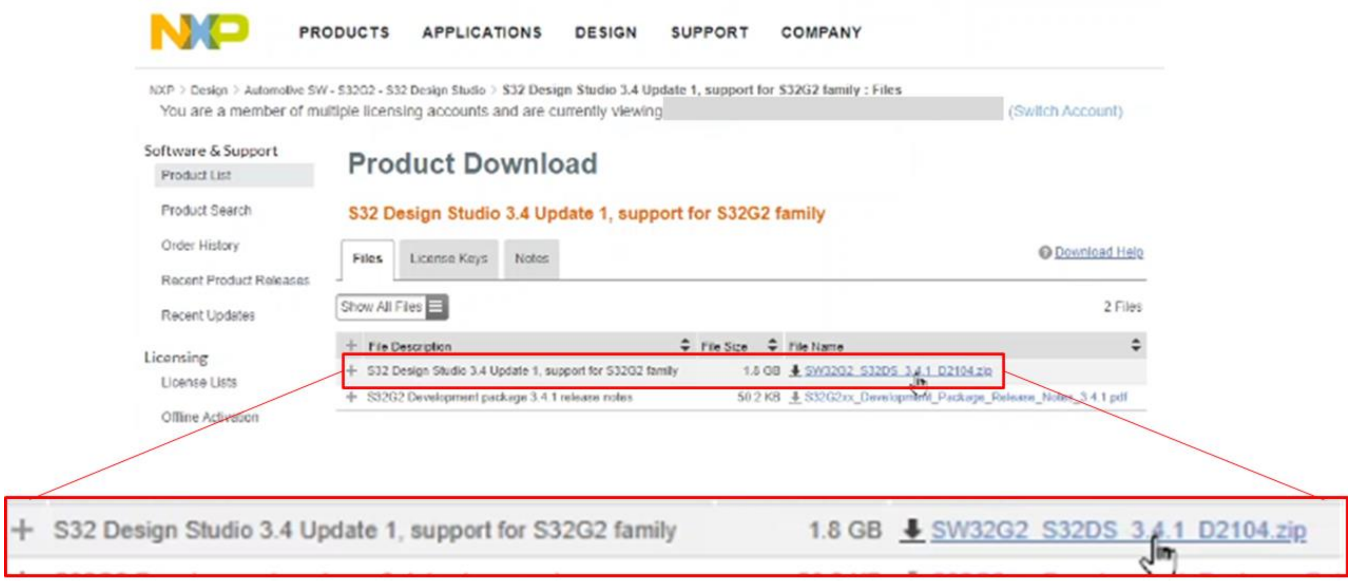
7 Files

File Description	File Size	File Name
S32 Design Studio 3.4 development packages for offline use	3.7 GB	SW32_S32DS_3.4.0_D2012.zip
S32 Design Studio 3.4 development packages for offline use, support for S32G2 family	1.3 GB	SW32G2_S32DS_3.4.0_D2012.zip
S32 Design Studio 3.4 Release Notes	72.5 KB	S32DS_Release_Notes_3.4.0.pdf
S32 Design Studio 3.4 S32G2 Development Package Release Notes	48 KB	S32G2xx_Development_Package_Release_Notes_3.4.0.pdf
S32 Design Studio Installation Guide	1.2 MB	S32DS_Installation_Guide_3.4.0.pdf
S32 Design Studio v3.4 Linux installer	1.1 GB	S32DS.3.4_b201217_linux.x86_64.bin
S32 Design Studio v3.4 Windows installer	1.5 GB	S32DS.3.4_b201217_win32.x86_64.exe

S32 Design Studio v3.4 Windows installer	1.5 GB	S32DS.3.4_b201217_win32.x86_64.exe

Figure 35. **Downloading S32DS3.4**

Download the S32 Design Studio 3.4 update 1 which has support for S32G2 family.



NXP > Design > Automotive SW - S32G2 - S32 Design Studio > S32 Design Studio 3.4 Update 1, support for S32G2 family : Files

You are a member of multiple licensing accounts and are currently viewing [redacted] (Switch Account)

Product Download

S32 Design Studio 3.4 Update 1, support for S32G2 family

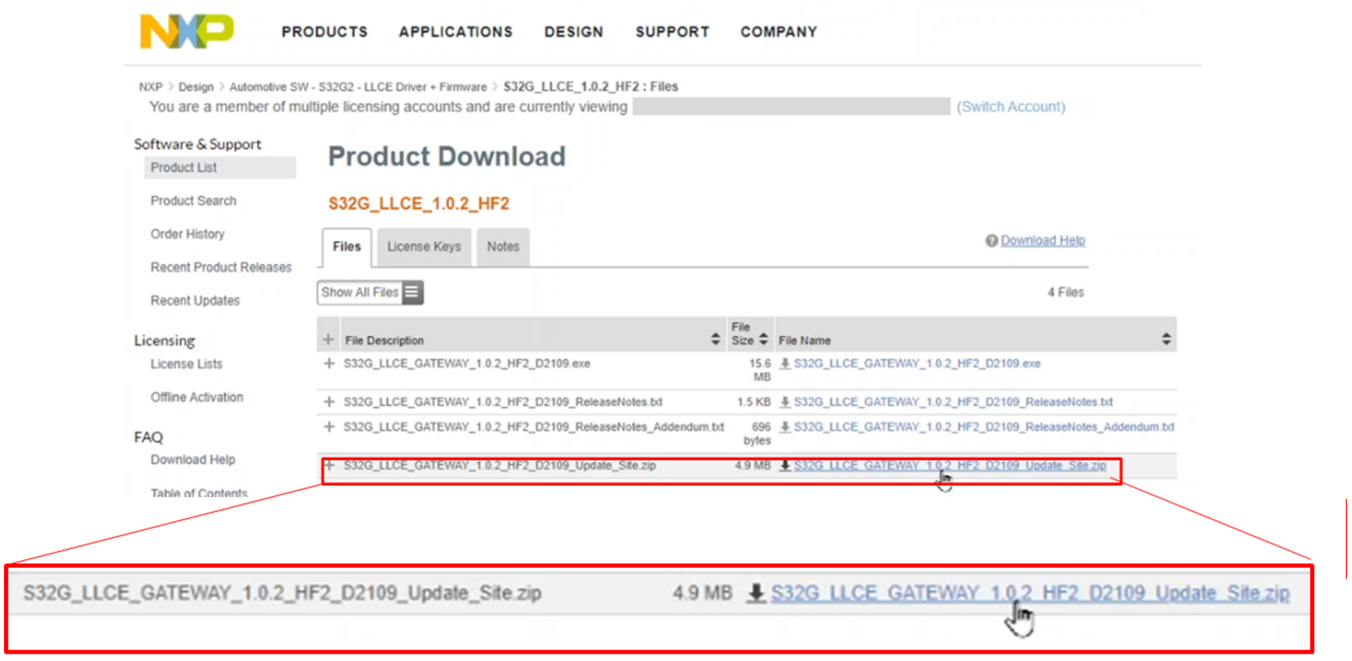
2 Files

File Description	File Size	File Name
S32 Design Studio 3.4 Update 1, support for S32G2 family	1.8 GB	SW32G2_S32DS_3.4.1_D2104.zip
S32G2 Development package 3.4.1 release notes	50.2 KB	S32G2xx_Development_Package_Release_Notes_3.4.1.pdf

S32 Design Studio 3.4 Update 1, support for S32G2 family	1.8 GB	SW32G2_S32DS_3.4.1_D2104.zip

Figure 36. **Update for support of S32G2 family**

Download the S32G_LLCE_GATEWAY_1.0.2_HF2_D2105 from update site.



NXP > Design > Automotive SW - S32G2 - LLCE Driver + Firmware > S32G_LLCE_1.0.2_HF2 : Files

You are a member of multiple licensing accounts and are currently viewing [redacted] (Switch Account)

Software & Support: Product List, Product Search, Order History, Recent Product Releases, Recent Updates
Licensing: License Lists, Offline Activation
FAQ: Download Help, Table of Contents

Product Download

S32G_LLCE_1.0.2_HF2

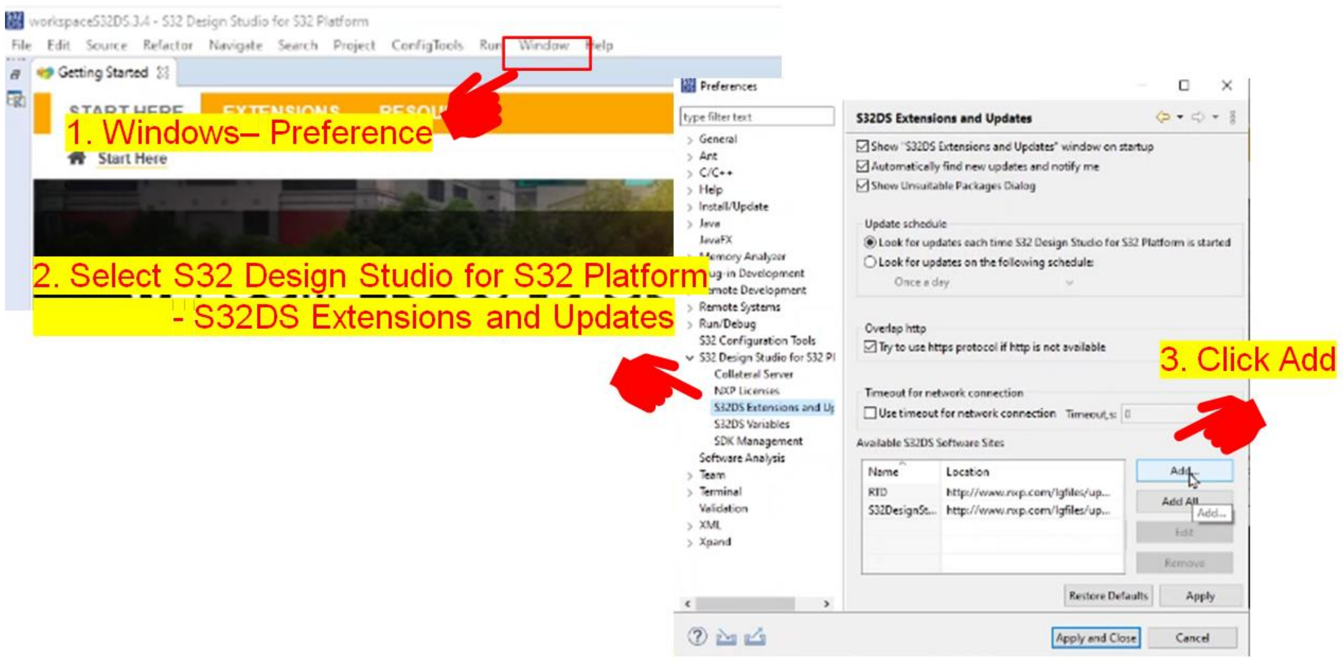
Files | License Keys | Notes — Download Help

Show All Files — 4 Files

File Description	File Size	File Name
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.exe	15.6 MB	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109.exe
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_ReleaseNotes.txt	1.5 KB	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_ReleaseNotes.txt
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_ReleaseNotes_Addendum.txt	696 bytes	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_ReleaseNotes_Addendum.txt
S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_Update_Site.zip	4.9 MB	S32G_LLCE_GATEWAY_1.0.2_HF2_D2109_Update_Site.zip

Figure 37. **Downloading S32G_LLCE_GATEWAY_1.0.2_HF2_D2105**

Add the downloaded three zip files in the S32DS3.4.



S32DS Extensions and Updates

- [✓] Show "S32DS Extensions and Updates" window on startup
- [✓] Automatically find new updates and notify me
- [✓] Show Unsuitable Packages Dialog

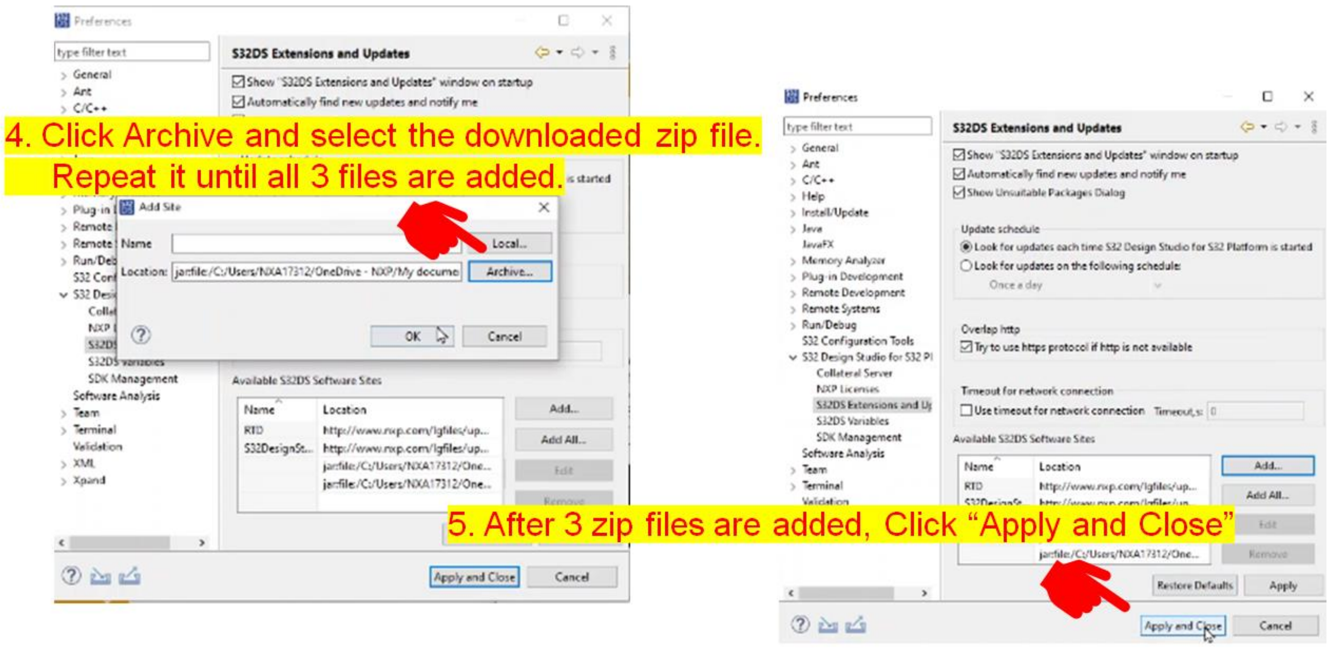
Update schedule
- [✓] Look for updates each time S32 Design Studio for S32 Platform is started
- [ ] Look for updates on the following schedule: Once a day

Overlap http
- [✓] Try to use https protocol if http is not available

Timeout for network connection
- [ ] Use timeout for network connection Timeout, s: 0

Available S32DS Software Sites

Name	Location
RTD	http://www.nxp.com/lgfiles/up...
S32DesignSt...	http://www.nxp.com/lgfiles/up...


4. Click Archive and select the downloaded zip file. Repeat it until all 3 files are added.

5. After 3 zip files are added, Click "Apply and Close"

Location: jar:file:/C:/Users/NXA17312/OneDrive - NXP/My docume

Name	Location
RTD	http://www.nxp.com/lgfiles/up...
S32DesignSt...	http://www.nxp.com/lgfiles/up...
	jar:file:/C:/Users/NXA17312/One...
	jar:file:/C:/Users/NXA17312/One...

Figure 38. **Adding the downloaded zip files**

Installing the S32 Design Studio 3.4 Update 1 with support for S32G2 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

- Platform pkg.
- Platform Tools pkg.
- S32G2xx Dev. Pkg.

3. Click “Install/Update 4 item(s)”.

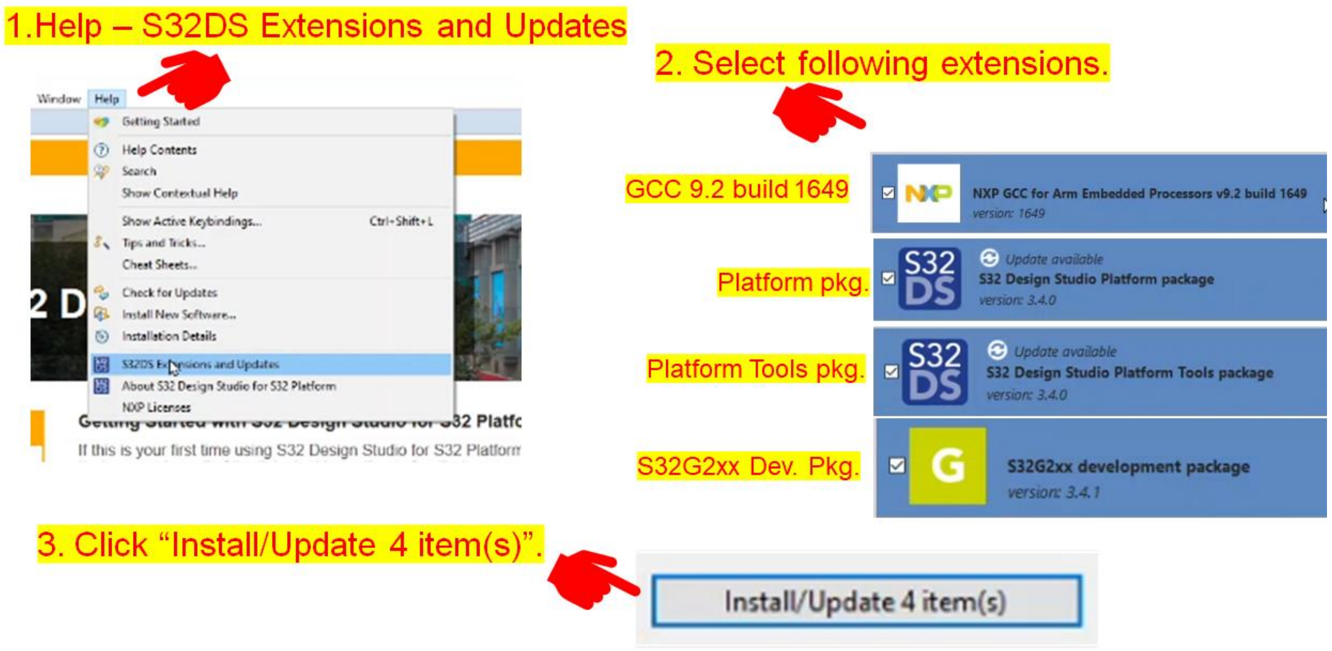

1.Help – S32DS Extensions and Updates
Window Help
Getting Started
Help Contents
Search
Show Contextual Help
Show Active Keybindings... Ctrl+Shift+L
Tips and Tricks...
Cheat Sheets...
Check for Updates
Install New Software...
Installation Details
S32DS Extensions and Updates
About S32 Design Studio for S32 Platform
NXP Licenses
2. Select following extensions.
GCC 9.2 build 1649
NXP GCC for Arm Embedded Processors v9.2 build 1649
version: 1649
Platform pkg.
Update available
S32 Design Studio Platform package
version: 3.4.0
Platform Tools pkg.
Update available
S32 Design Studio Platform Tools package
version: 3.4.0
S32G2xx Dev. Pkg.
S32G2xx development package
version: 3.4.1
3. Click “Install/Update 4 item(s)”.
Install/Update 4 item(s)

Figure 39. **Steps to update support for S32G2 family**

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

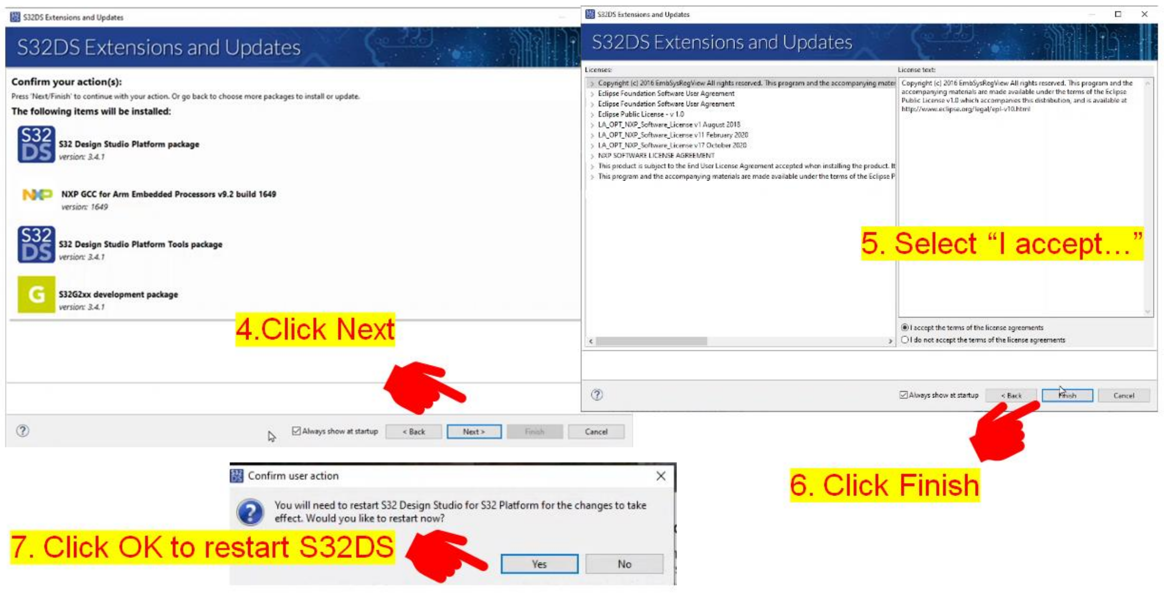
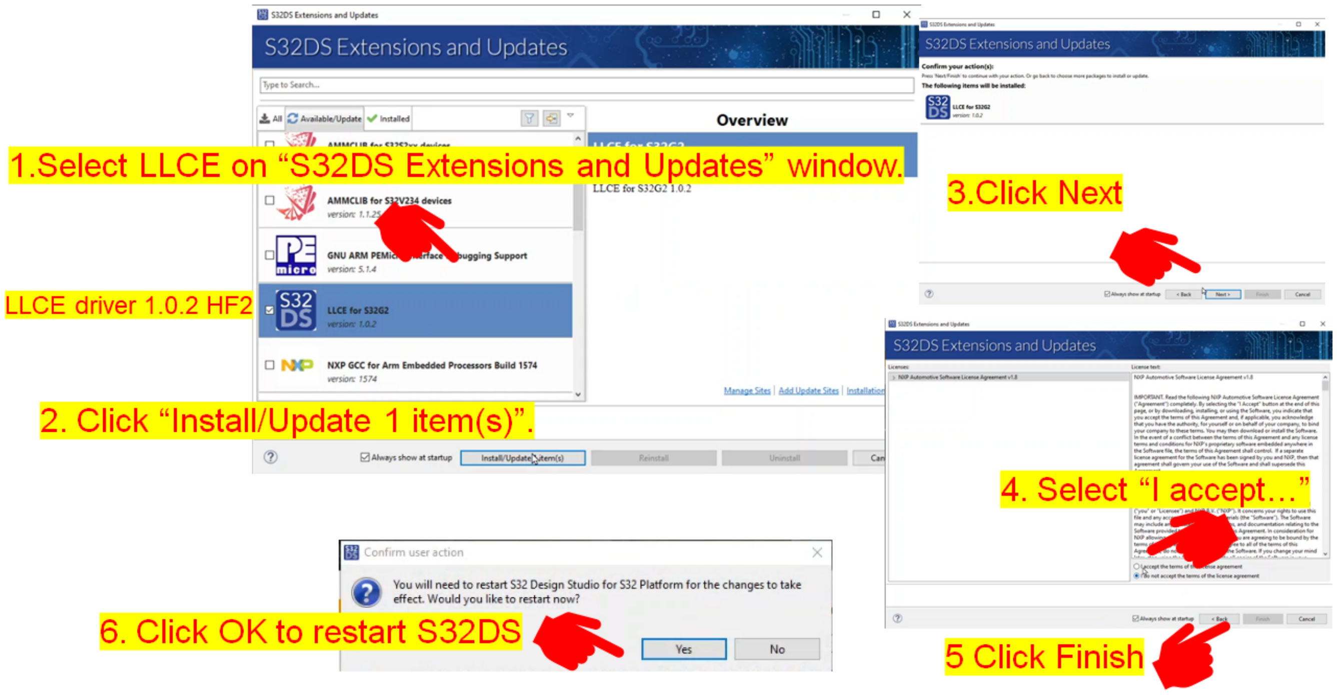

S32DS Extensions and Updates
Confirm your action(s):
Press 'Next/Finish' to continue with your action. Or go back to choose more packages to install or update.
The following items will be installed:
S32 Design Studio Platform package
version: 3.4.1
NXP GCC for Arm Embedded Processors v9.2 build 1649
version: 1649
S32 Design Studio Platform Tools package
version: 3.4.1
S32G2xx development package
version: 3.4.1
4.Click Next
Always show at startup
< Back
Next >
Finish
Cancel
Licenses:
License text:
5. Select “I accept…”
I accept the terms of the license agreements
I do not accept the terms of the license agreements
6. Click Finish
Confirm user action
You will need to restart S32 Design Studio for S32 Platform for the changes to take effect. Would you like to restart now?
7. Click OK to restart S32DS
Yes
No

Figure 40. **Steps to finish S32DS for S32G2 family**

5.2. Installing LLCE driver on S32DS

To install LLCE driver follow these steps:

1. Select LLCE on “S32DS Extensions and Updates” window.
2. Click “Install/Update 1 item(s)” and click Next.
3. Select “I accept…” and click on Finish. A pop up window appears to restart S32DS, click Yes.



You will need to restart S32 Design Studio for S32 Platform for the changes to take effect. Would you like to restart now?

Figure 41. **Steps to install LLCE driver**

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_S32G274A_M7 and click on Finish.

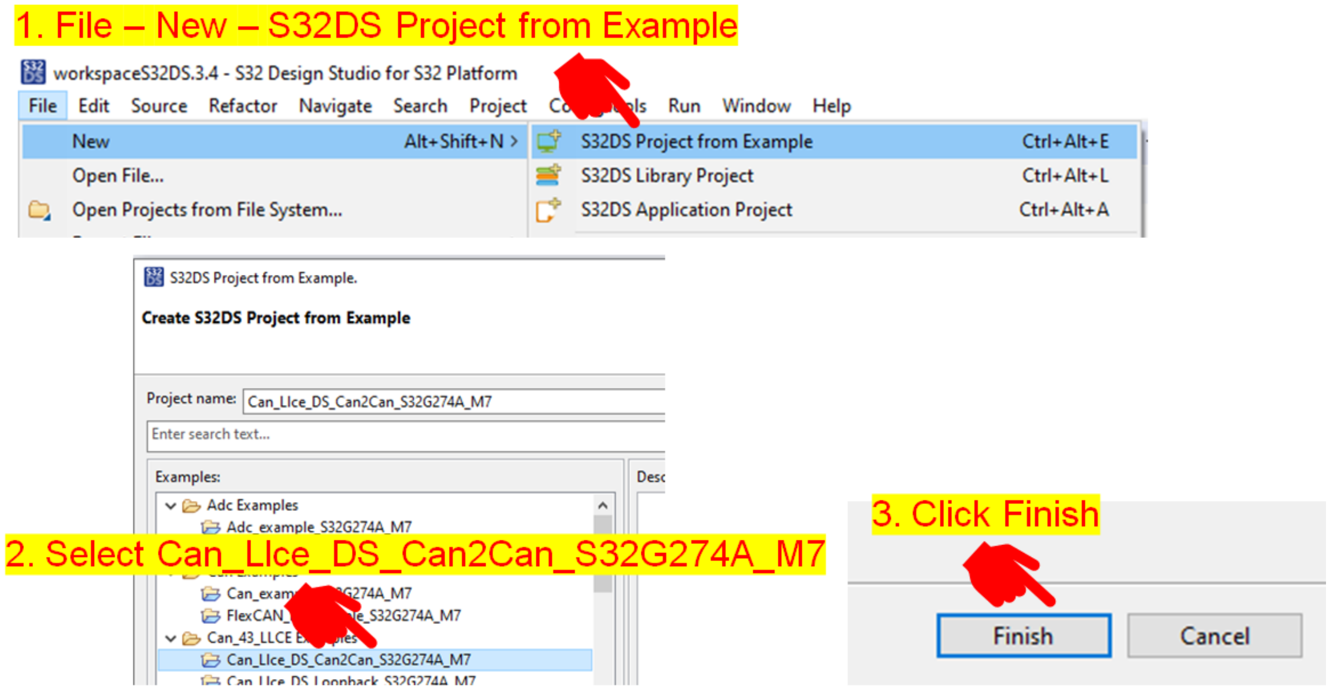

1. File – New – S32DS Project from Example
workspaceS32DS.3.4 - S32 Design Studio for S32 Platform
File Edit Source Refactor Navigate Search Project ConfigTools Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
S32DS Project from Example Ctrl+Alt+E
S32DS Library Project Ctrl+Alt+L
S32DS Application Project Ctrl+Alt+A
S32DS Project from Example.
Create S32DS Project from Example
Project name: Can_Llce_DS_Can2Can_S32G274A_M7
Enter search text...
Examples:
Adc Examples
Adc_example_S32G274A_M7
Can_example_S32G274A_M7
FlexCAN_example_S32G274A_M7
Can_43_LLCE Examples
Can_Llce_DS_Can2Can_S32G274A_M7
Can_Llce_DS_Loopback_S32G274A_M7
2. Select Can_Llce_DS_Can2Can_S32G274A_M7
3. Click Finish
Finish
Cancel

Figure 42. **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.

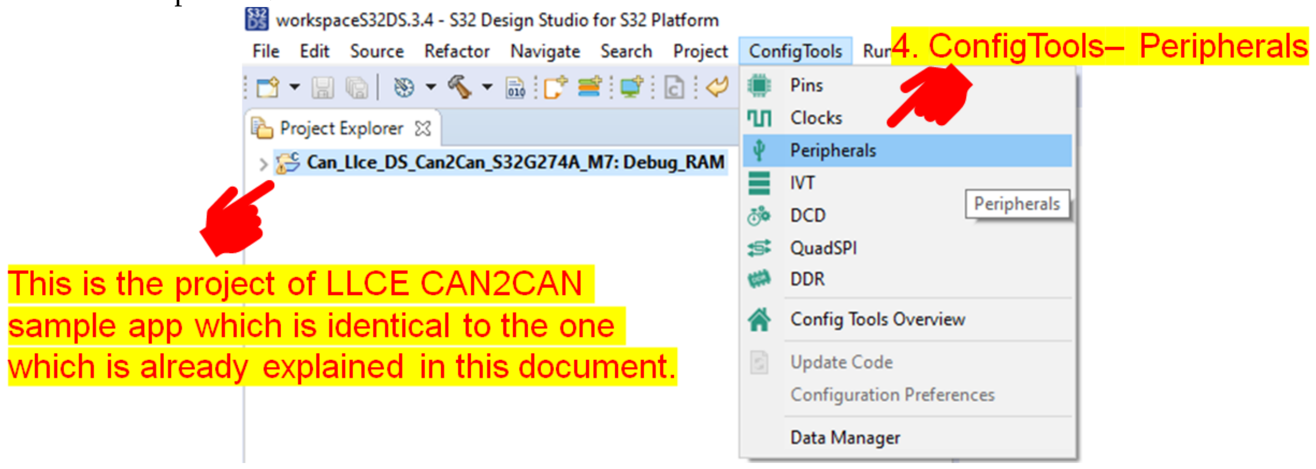

workspaceS32DS.3.4 - S32 Design Studio for S32 Platform
File Edit Source Refactor Navigate Search Project ConfigTools Run
4. ConfigTools– Peripherals
Project Explorer
Can_Llce_DS_Can2Can_S32G274A_M7: Debug_RAM
Pins
Clocks
Peripherals
IVT
DCD
QuadSPI
DDR
Config Tools Overview
Update Code
Configuration Preferences
Data Manager
Peripherals
This is the project of LLCE CAN2CAN sample app which is identical to the one which is already explained in this document.

Figure 43. **Selecting Peripheral**

4. To set up configuration tools Select Can_Llce_DS_Can2Can_S32G274A_M7.

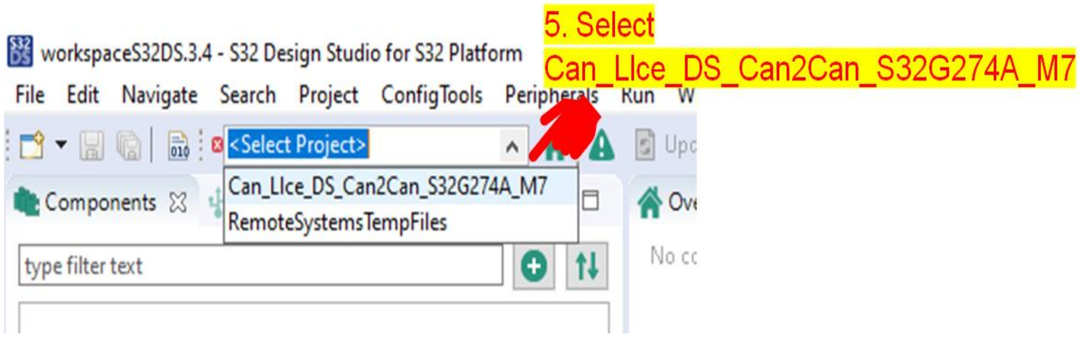
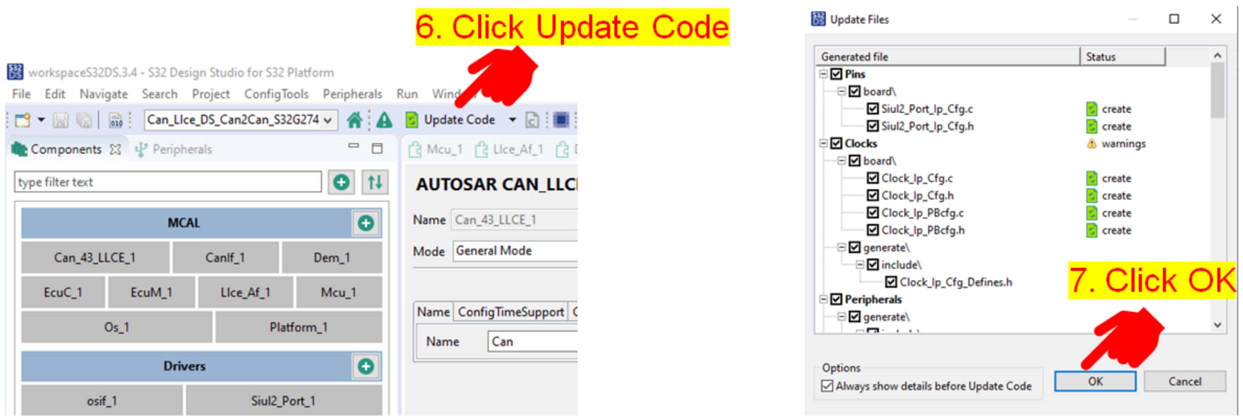

5. Select
Can_Llce_DS_Can2Can_S32G274A_M7
workspaceS32DS.3.4 - S32 Design Studio for S32 Platform
File Edit Navigate Search Project ConfigTools Peripherals Run W
<Select Project>
Can_Llce_DS_Can2Can_S32G274A_M7
RemoteSystemsTempFiles
Components
type filter text

Figure 44. **Configuring tools**

5. Update the code by clicking on Update Code. Once code is updated click on Ok.



Generated file	Status
Pins	
board\	
Siul2_Port_Ip_Cfg.c	create
Siul2_Port_Ip_Cfg.h	create
Clocks	warnings
board\	
Clock_Ip_Cfg.c	create
Clock_Ip_Cfg.h	create
Clock_Ip_PBcfg.c	create
Clock_Ip_PBcfg.h	create
generate\	
include\	
Clock_Ip_Cfg_Defines.h	
Peripherals	
generate\	

Figure 45. **Updating code**

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

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

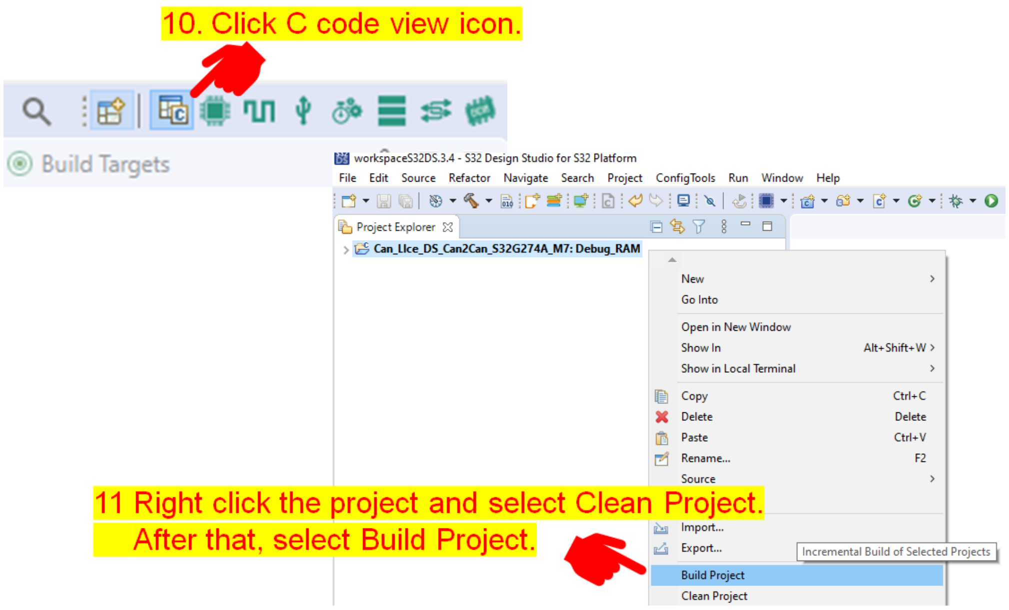
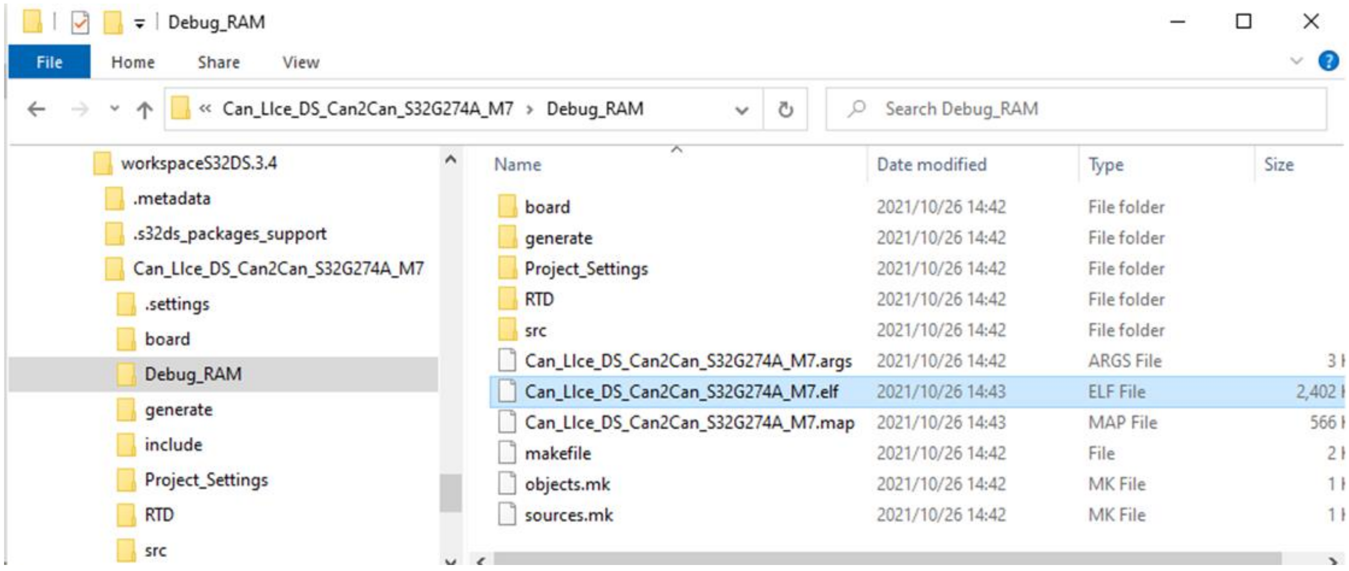

10. Click C code view icon.
Build Targets
workspaceS32DS.3.4 - S32 Design Studio for S32 Platform
File Edit Source Refactor Navigate Search Project ConfigTools Run Window Help
Project Explorer
Can_Llce_DS_Can2Can_S32G274A_M7: Debug_RAM
New
Go Into
Open in New Window
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Delete Delete
Paste Ctrl+V
Rename... F2
Source
11 Right click the project and select Clean Project.
After that, select Build Project.
Import...
Export...
Incremental Build of Selected Projects
Build Project
Clean Project

Figure 46. **Creating a new project**

The Elf file can be found in your workspace inside the folder "Can_Llce_DS_Can2Can_S32G274A_M7/Debug_RAM".



Name	Date modified	Type	Size
board	2021/10/26 14:42	File folder	
generate	2021/10/26 14:42	File folder	
Project_Settings	2021/10/26 14:42	File folder	
RTD	2021/10/26 14:42	File folder	
src	2021/10/26 14:42	File folder	
Can_Llce_DS_Can2Can_S32G274A_M7.args	2021/10/26 14:42	ARGS File	3
Can_Llce_DS_Can2Can_S32G274A_M7.elf	2021/10/26 14:43	ELF File	2,402
Can_Llce_DS_Can2Can_S32G274A_M7.map	2021/10/26 14:43	MAP File	566
makefile	2021/10/26 14:42	File	2
objects.mk	2021/10/26 14:42	MK File	1
sources.mk	2021/10/26 14:42	MK File	1

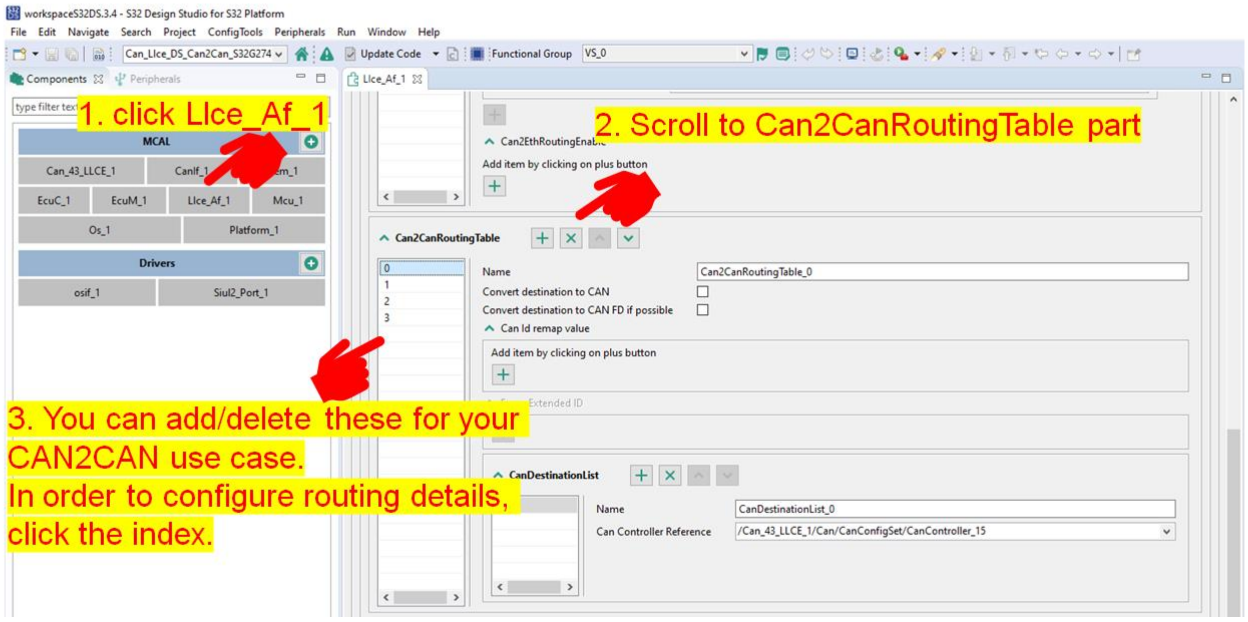
Figure 47. **Elf file location**

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.

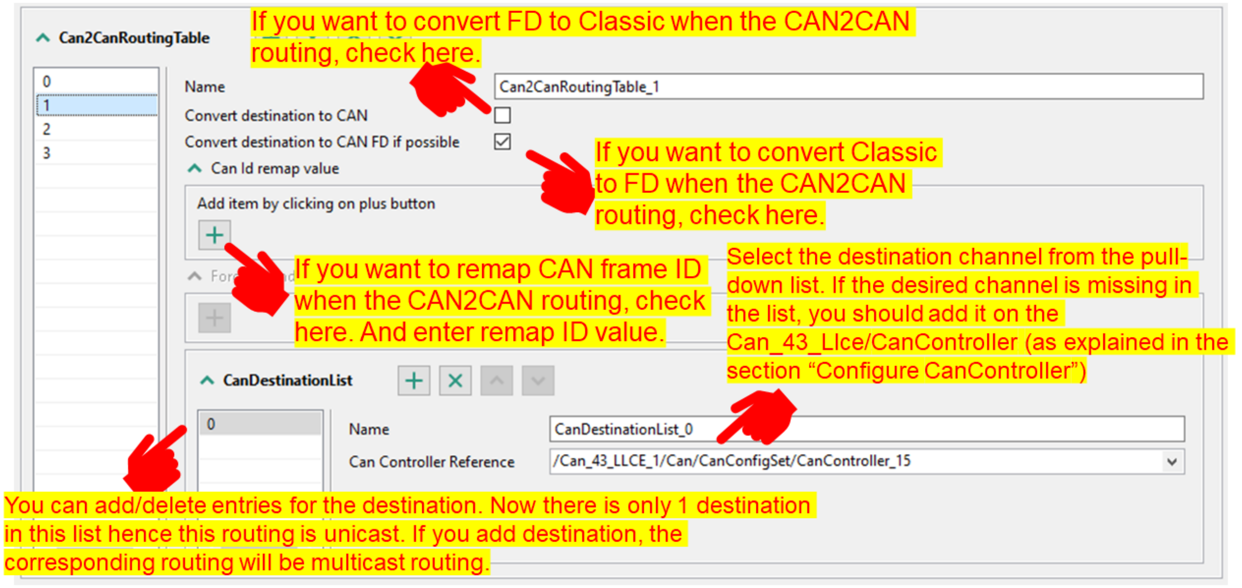
3. You can add/delete these for your CAN2CAN use case. In order to configure routing details, click the index.


1. click 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.

Name: Can2CanRoutingTable_0
Convert destination to CAN: ☐
Convert destination to CAN FD if possible: ☐
Name: CanDestinationList_0
Can Controller Reference: /Can_43_LLCE_1/Can/CanConfigSet/CanController_15

Figure 48. **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.


If you want to convert FD to Classic when the CAN2CAN routing, check here.

If you want to convert Classic to FD when the CAN2CAN routing, check here.

If you want to remap CAN frame ID when the CAN2CAN routing, check here. And enter remap ID value.

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”)

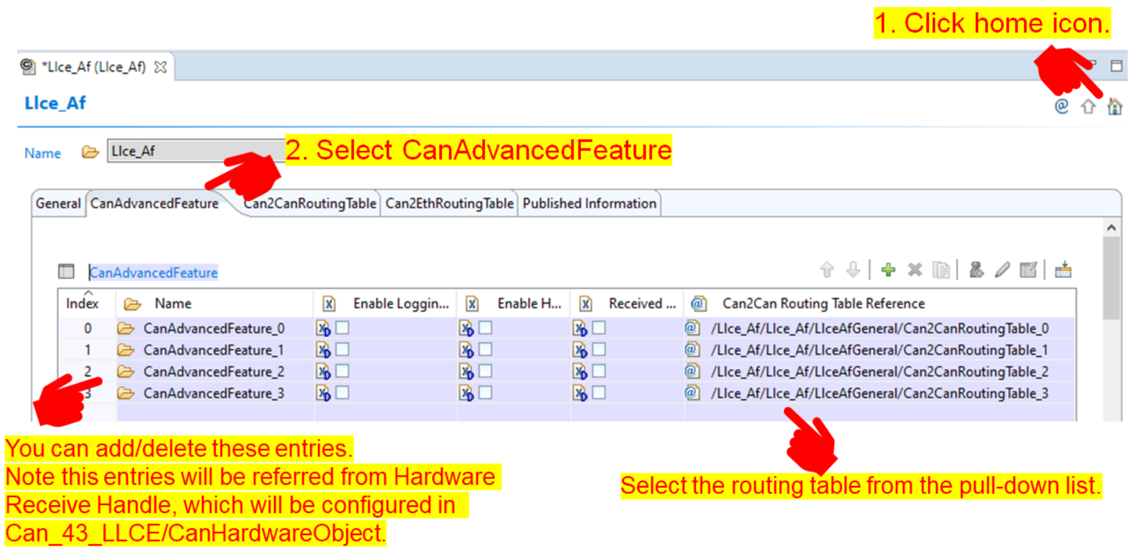
You can add/delete entries for the destination. Now there is only 1 destination in this list hence this routing is unicast. If you add destination, the corresponding routing will be multicast routing.

Name: Can2CanRoutingTable_1
Convert destination to CAN: ☐
Convert destination to CAN FD if possible: ☑
Name: CanDestinationList_0
Can Controller Reference: /Can_43_LLCE_1/Can/CanConfigSet/CanController_15

Figure 49. **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.
8. To ensure the CAN2CAN routing is referred from CanAdvancedFeature table click on the Home icon.



Index	Name	Enable Loggin...	Enable H...	Received ...	Can2Can Routing Table Reference
0	CanAdvancedFeature_0	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_0
1	CanAdvancedFeature_1	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_1
2	CanAdvancedFeature_2	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_2
3	CanAdvancedFeature_3	☐	☐	☐	/Llce_Af/Llce_Af/LlceAfGeneral/Can2CanRoutingTable_3

You can add/delete these entries.
Note this entries will be referred from Hardware Receive Handle, which will be configured in Can_43_LLCE/CanHardwareObject.

Select the routing table from the pull-down list.

Figure 50. **CanAdvancedFeature table**

9. Select the CanAdvancedFeature tab.
10. You can select the routing table from the pull down list. You can also add/delete the entries.

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.

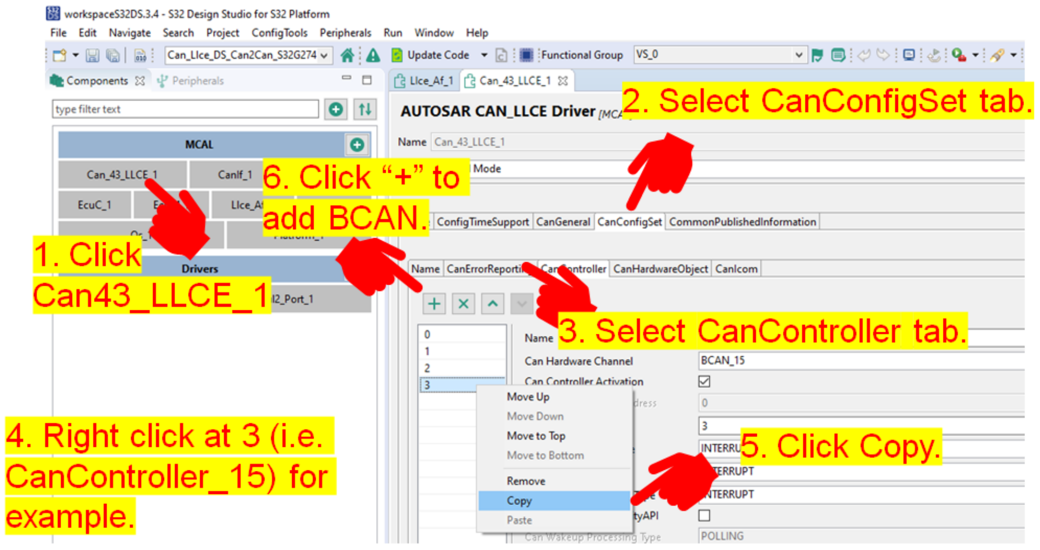

2. Select CanConfigSet tab.
6. Click "+" to add BCAN.
1. Click Can43_LLCE_1
3. Select CanController tab.
4. Right click at 3 (i.e. CanController_15) for example.
5. Click Copy.
Move Up
Move Down
Move to Top
Move to Bottom
Remove
Copy
Paste

Figure 51. **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.

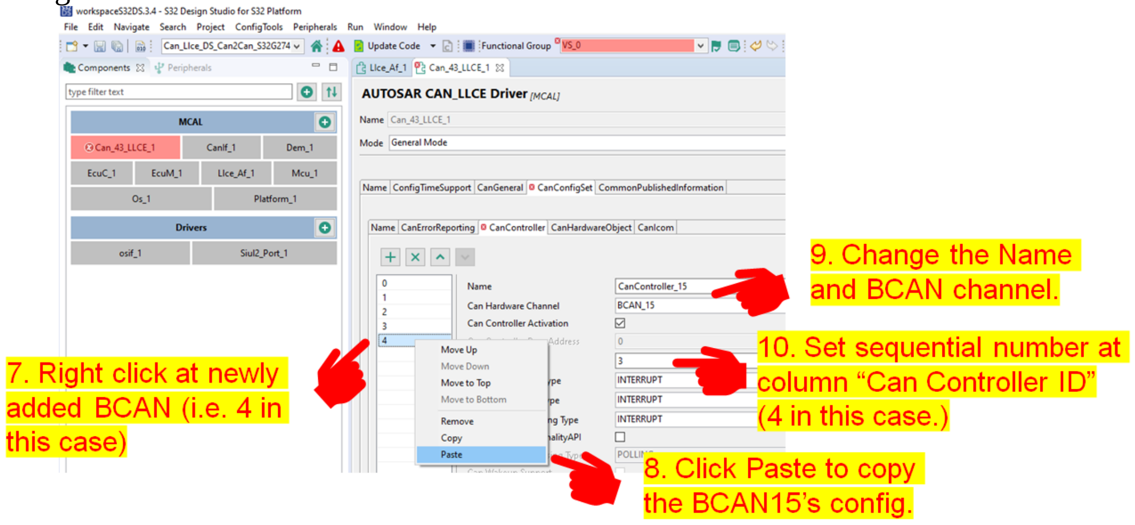
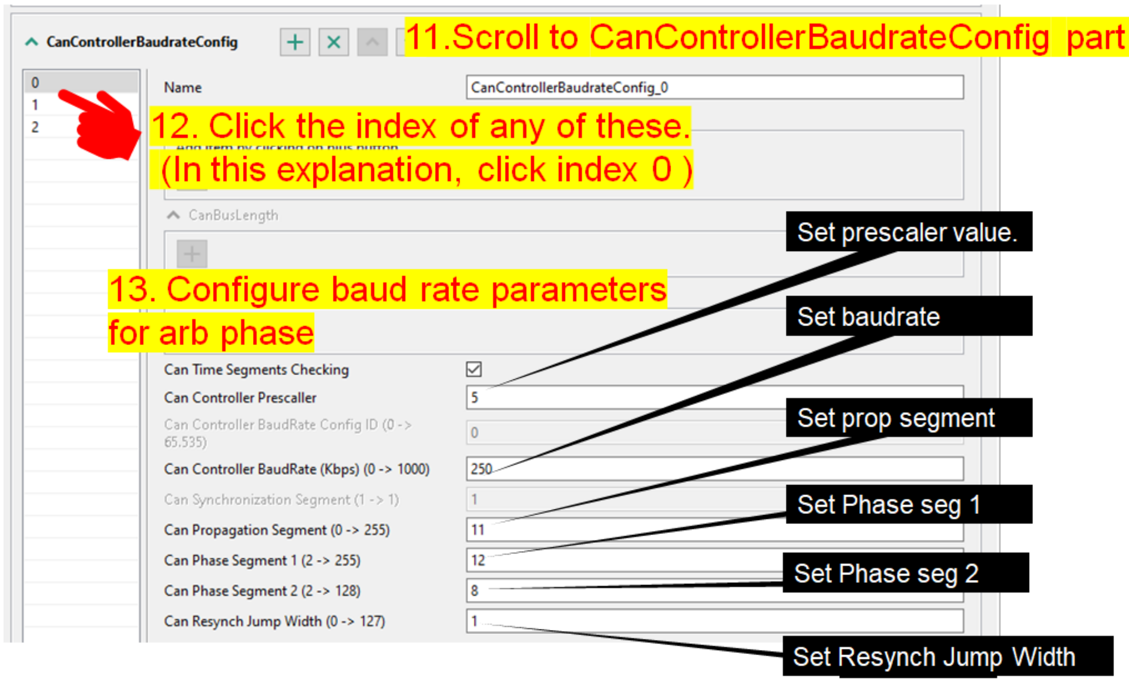

9. Change the Name and BCAN channel.
10. Set sequential number at column "Can Controller ID" (4 in this case.)
7. Right click at newly added BCAN (i.e. 4 in this case)
8. Click Paste to copy the BCAN15's config.
Move Up
Move Down
Move to Top
Move to Bottom
Remove
Copy
Paste

Figure 52. **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).

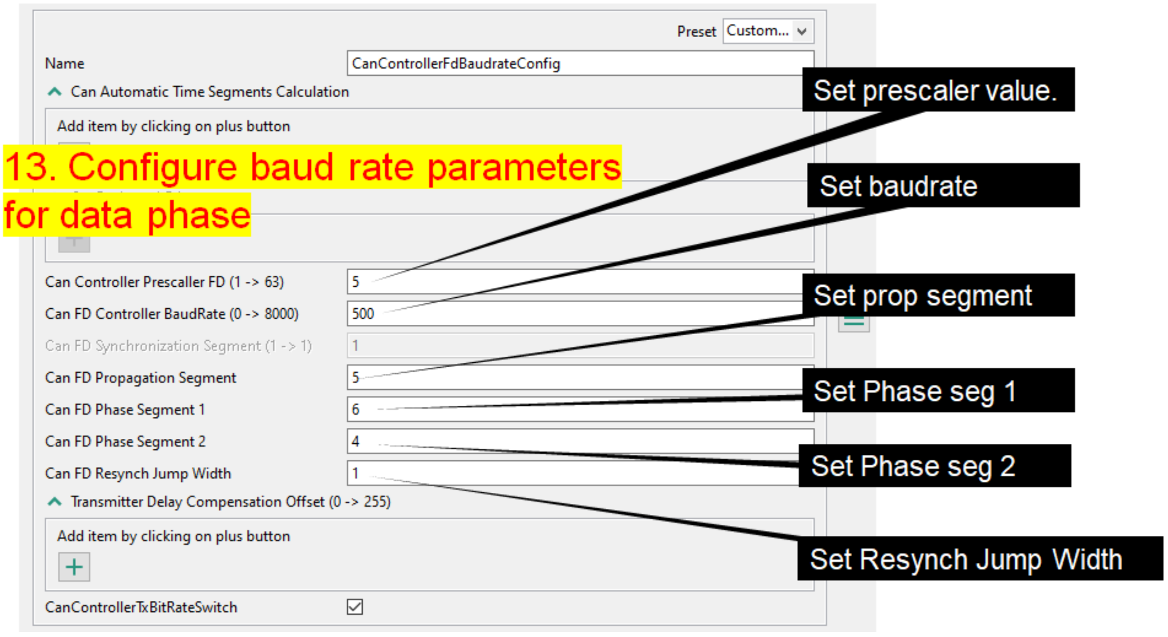
9. Configure baud rate parameters for arb phase.


11.Scroll to CanControllerBaudrateConfig part.
12. Click the index of any of these.
(In this explanation, click index 0)
13. Configure baud rate parameters for arb phase
Set prescaler value.
Set baudrate
Set prop segment
Set Phase seg 1
Set Phase seg 2
Set Resynch Jump Width
CanControllerBaudrateConfig
0
1
2
CanBusLength

Name	CanControllerBaudrateConfig_0
Can Time Segments Checking	✓
Can Controller Prescaller	5
Can Controller BaudRate Config ID (0 -> 65.535)	0
Can Controller BaudRate (Kbps) (0 -> 1000)	250
Can Synchronization Segment (1 -> 1)	1
Can Propagation Segment (0 -> 255)	11
Can Phase Segment 1 (2 -> 255)	12
Can Phase Segment 2 (2 -> 128)	8
Can Resynch Jump Width (0 -> 127)	1

Figure 53. **Configuring Baud rate setting**

10. Configure baud rate parameters for data phase.


13. Configure baud rate parameters for data phase
Set prescaler value.
Set baudrate
Set prop segment
Set Phase seg 1
Set Phase seg 2
Set Resynch Jump Width
Preset
Custom...
Can Automatic Time Segments Calculation
Add item by clicking on plus button
Transmitter Delay Compensation Offset (0 -> 255)
Add item by clicking on plus button

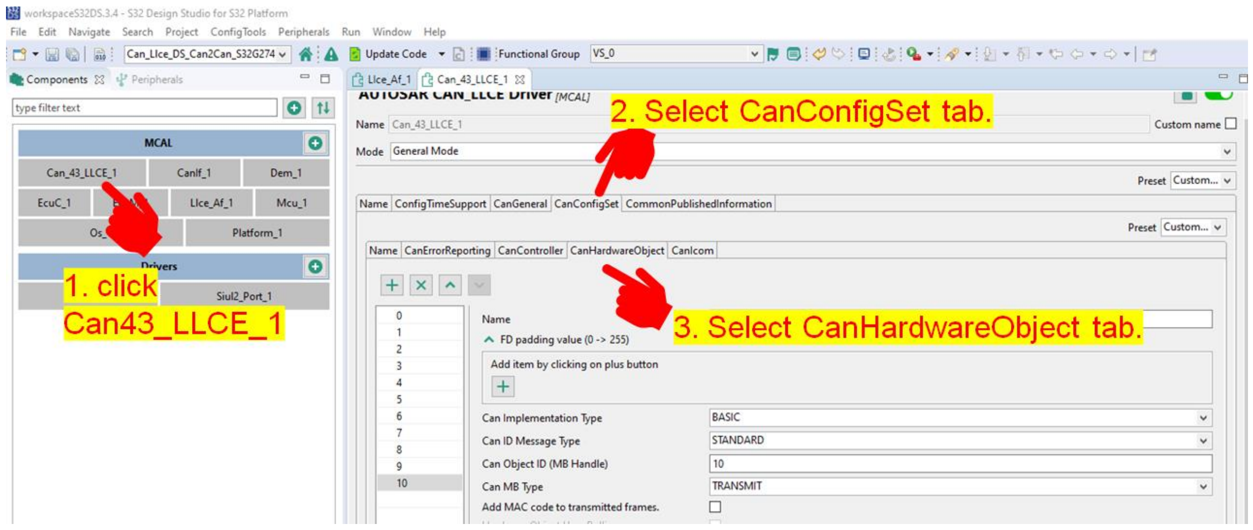
Name	CanControllerFdBaudrateConfig
Can Controller Prescaller FD (1 -> 63)	5
Can FD Controller BaudRate (0 -> 8000)	500
Can FD Synchronization Segment (1 -> 1)	1
Can FD Propagation Segment	5
Can FD Phase Segment 1	6
Can FD Phase Segment 2	4
Can FD Resynch Jump Width	1
CanControllerTxBitRateSwitch	✓

Figure 54. **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.



AUTOSAR CAN_LLCE Driver [MCAL]

Name: Can_43_LLCE_1

Mode: General Mode

FD padding value (0 -> 255)

Add item by clicking on plus button

Can Implementation Type: BASIC

Can ID Message Type: STANDARD

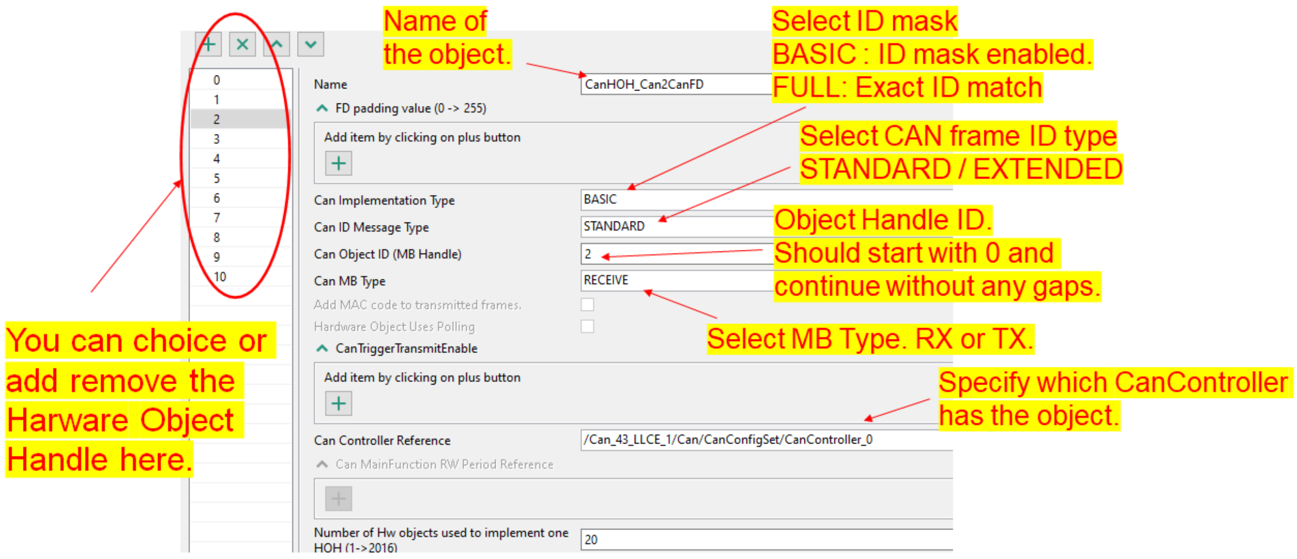
Can Object ID (MB Handle): 10

Can MB Type: TRANSMIT

Add MAC code to transmitted frames.

Figure 55. **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.


Name of the object.
Select ID mask
BASIC : ID mask enabled.
FULL: Exact ID match
Select CAN frame ID type
STANDARD / EXTENDED
Object Handle ID.
Should start with 0 and continue without any gaps.
Select MB Type. RX or TX.
Specify which CanController has the object.
You can choice or add remove the Harware Object Handle here.
0
1
2
3
4
5
6
7
8
9
10

Name: CanHOH_Can2CanFD

FD padding value (0 -> 255)

Add item by clicking on plus button

Can Implementation Type: BASIC

Can ID Message Type: STANDARD

Can Object ID (MB Handle): 2

Can MB Type: RECEIVE

Add MAC code to transmitted frames.

Hardware Object Uses Polling

CanTriggerTransmitEnable

Add item by clicking on plus button

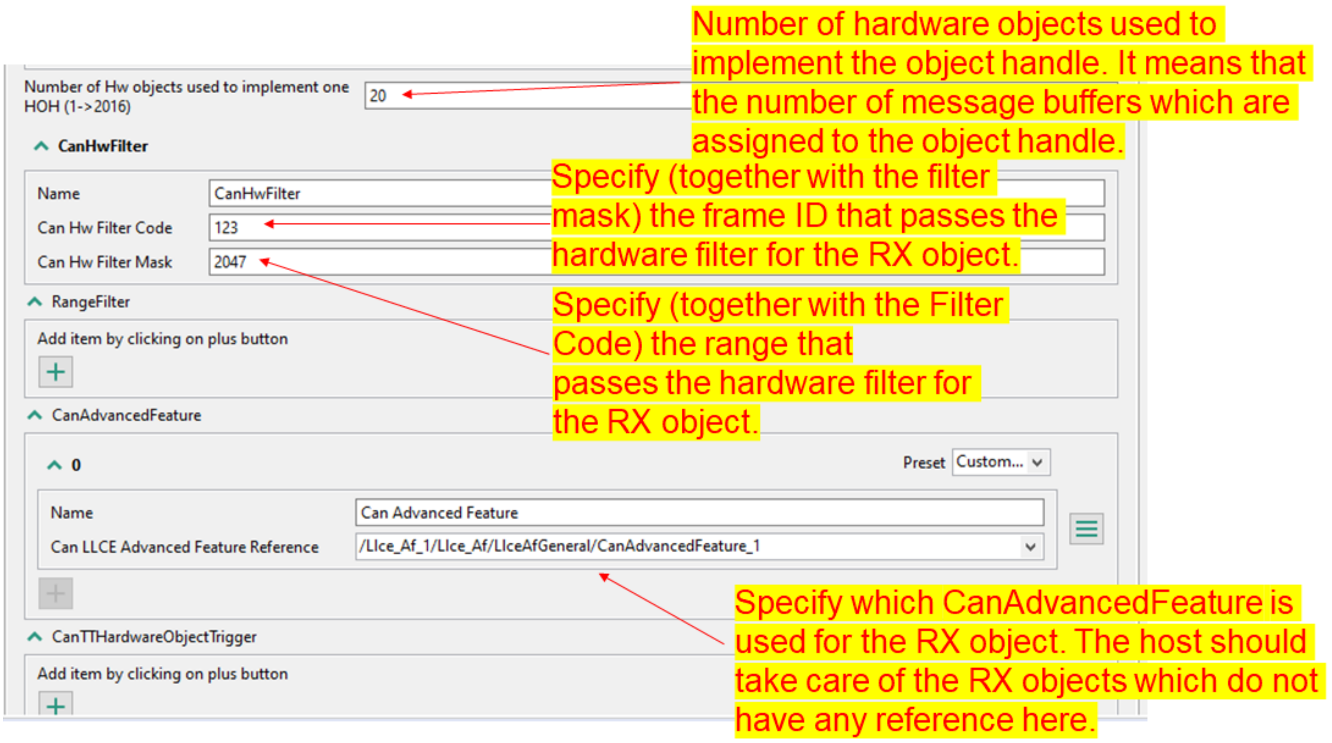
Can Controller Reference: /Can_43_LLCE_1/Can/CanConfigSet/CanController_0

Can MainFunction RW Period Reference

Number of Hw objects used to implement one HOH (1->2016): 20

Figure 56. **Configuring Can hardware object 2**

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


Number of Hw objects used to implement one HOH (1->2016): 20

CanHwFilter

Name: CanHwFilter

Can Hw Filter Code: 123

Can Hw Filter Mask: 2047

RangeFilter

Add item by clicking on plus button

CanAdvancedFeature

0 — Preset: Custom...

Name: Can Advanced Feature

Can LLCE Advanced Feature Reference: /Llce_Af_1/Llce_Af/LlceAfGeneral/CanAdvancedFeature_1

CanTTHardwareObjectTrigger

Add item by clicking on plus button

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.

Specify (together with the filter mask) the frame ID that passes the hardware filter for the RX object.

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

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.

Figure 57. **Message buffer configuration**

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN13423
Rev. 0
11/2021