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

![Overview diagram](image)

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

If you do not have the latest PFE MCAL 4.4 driver 0.9.4 and RTD package, download both of them.
After the installation of PFE MCAL and RTD, put the folders and files into the tresos/plugins as shown below.
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.
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.

Figure 12. Modifying config.mk

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.

![Diagram showing CAN2CAN Routing][1]

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.

```
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.

---

[1]: #/image.png
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).
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.

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.
If you capture the odd CAN buses with Logic Analyzer, you can see the routed CAN frames.
4. Configuring 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
2. Click “Existing Projects into Workspace”
3. Click Next
4. Check “Copy projects into workspace”
5. Click Browse

![Figure 19. Importing file](image)

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

Using CAN2CAN, CAN2ETH and ETH2CAN Features of LLCE on S32G, Rev. 0, 11/2021

NXP Semiconductors

15

Figure 20. Importing CAN2CAN config file

6. Choose “Tresos_CAN2CAN_Project”.

7. Click OK

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

Figure 21. Importing CAN2ETH and ETH2CAN config file

6. Choose “tresos”
To import sample configuration, browse and select the root directory, click on finish. Refer to the following screenshot.

![Importing sample configuration](image)

8. Click Finish

To rename the imported project config right click and select Rename. Enter the new name in the dialog box and click OK.
Every time you start configuration on the Tresos studio, you need to load 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 25. Configuring LLCE_Af

Configure routing details in General Tab.

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.
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 27. CAN advanced feature table

Figure 28. Configuring Llce_Af

Destination MAC address of the assembled Ethernet frame. To configure these parameters, see following pages.
Use the following formula to calculate the buffer size.
If the customer want to pack $N$ ACF msg / packet, the Buffer size should be equal or larger than
$26+(N-1)*(8+\text{can\_msg\_payload}) -1 + 72$
and less than
$26+N*(8+\text{can\_msg\_payload}) -1+72$

**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. $26+10*(8+4) -1+72$).

![Figure 29. Calculating buffer size](image)

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.
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.

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 ).
10. Configure baud rate parameters.

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

- Set prescaler value.
- Set baudrate
- Set prop segment
- Set Phase seg 1
- Set Phase seg 2
- Set Resynch Jump Width

**Figure 33. 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. 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

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.

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.

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.

![Figure 35. Downloading S32DS3.4](image)

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

![Figure 36. Update for support of S32G2 family](image)
Download the S32G_LLCE_GATEWAY_1.0.2_HF2_D2105 from update site.

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

Add the downloaded three zip files in the S32DS3.4.
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
2. Select following extensions.
   - GCC 9.2 build 1649
   - Platform pkg.
   - Platform Tools pkg.
   - S32G2xx Dev. Pkg.

3. Click “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.

4. Click Next

5. Select “I accept…”

6. Click Finish

7. Click OK to restart S32DS

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.

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.

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.

4. To set up configuration tools Select Can_Llce_DS_Can2Can_S32G274A_M7.
5. Update the code by clicking on Update Code. Once code is updated click on Ok.

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.
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”.

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.

10. Click C code view icon.

11 Right click the project and select Clean Project. After that, select Build Project.
3. You can add/delete these for your CAN2CAN use case. In order to configure routing details, click the index.

![Figure 48. Configuring LLCE_Af for CAN2CAN](image)

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 on plus button and enter remap ID value.

![Figure 49. CAN2CAN routing table](image)

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.

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

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

10. Configure baud rate parameters for data phase.

13. Configure baud rate parameters for data phase

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.

![Image showing the process of clicking on Can43_LLCE_1 and selecting CanConfigSet and CanHardwareObject tabs]

2. Select CanConfigSet tab.

3. Select CanHardwareObject tab.

![Image showing the configuration of the Can hardware object]

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.

![Image showing the configuration of the Can hardware object 2]

Figure 56. Configuring Can hardware object 2

4. In this window you can configure message buffer related settings.
Figure 57. **Message buffer configuration**

- 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.
Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

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

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

© 2021 NXP B.V.