**NXP Semiconductors**
Product Brief

Document Number: 1.7
Rev. 1.7, 10/2023

# S32G2/G3 LLCE Standard Firmware Product Brief

## Contents

## 1. Software Product Overview

**NXP Low Latency Communication Engine** (LLCE) controls the traditional automotive communication interfaces such as CAN, LIN, and FlexRay$^{TM}$. The LLCE offloads the Host CPU from all interface level tasks along with validating, authenticating of frames for security and also handles the encryption/decryption process of frames with the help of on-chip Hardware Security Module (HSE).

It provides a platform to develop software to customize the processing of communication interfaces and handshaking with the host CPU.

LLCE is aimed to offer:

- Low latency processing of communication interface
- Offloading of host CPU for all interface related tasks
- Direct data transfer to/from HSE for security related tasks

The communication interfaces handled by LLCE are as below:

- 16 CAN interfaces, capable of CAN2.0 and CAN FD (Flexible Data rate) (5 Mbps)
- 4 LIN interfaces, capable of 20 Kbps each
- 1 FlexRay interface (20 Mbps)

NXP Low Latency Communication Engine runs a multicore application that controls the hardware communication interfaces and is called “firmware” Its architecture is described in Figure 1.

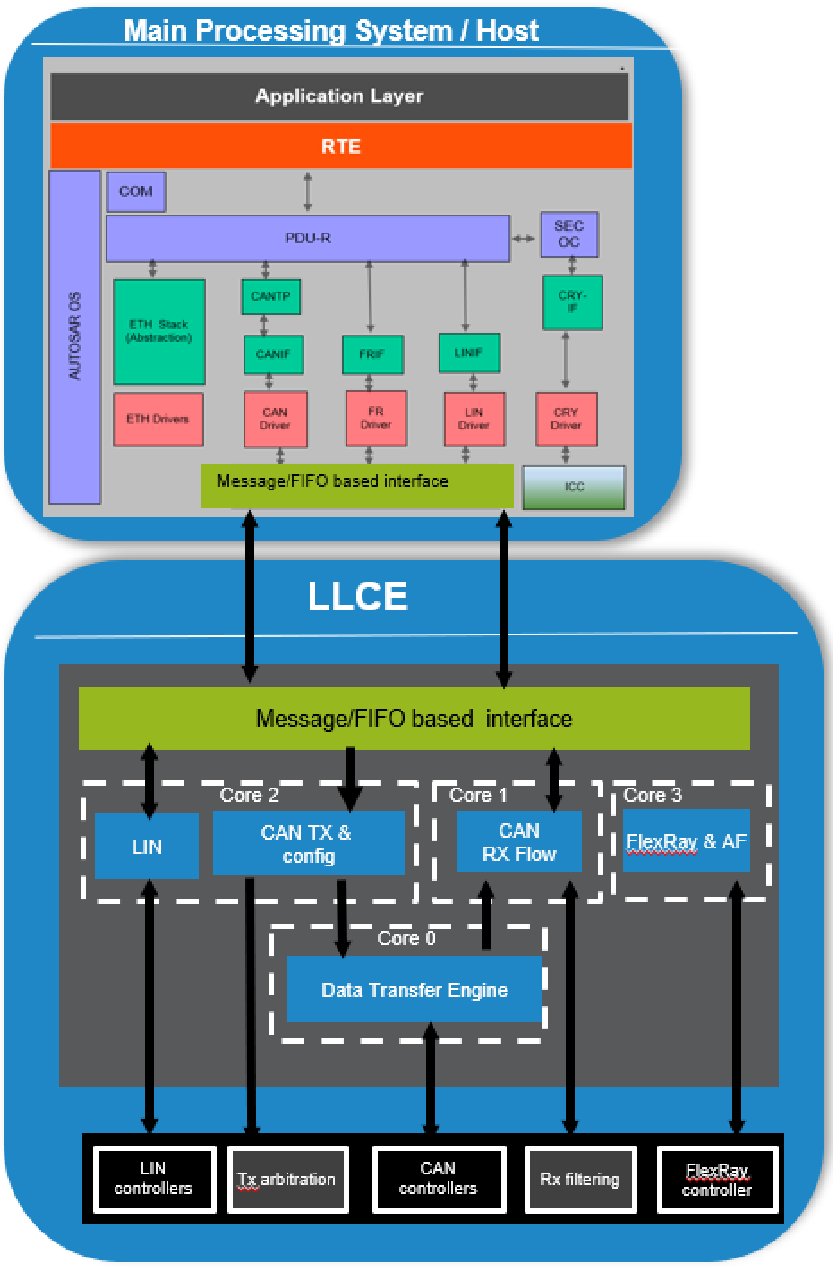

Figure 1. **LLCE Architecture Diagram**

# 2. Software Content

NXP solution is a fully programmable engine with firmware that supports:

- Offload of host CPU for all communication interface related tasks
    - Reduced interrupt load on the host core
    - Advanced software filtering
- Flexible control and data interface exposed to the Host cores
- Hardware acceleration for filtering and prioritization of messages

The firmware services are integrated into the AUTOSAR® Communication stack via AUTOSAR® MCAL drivers that are provided by NXP:

- CAN_LLCE
- LIN_LLCE
- FR_LLCE

AUTOSAR® drivers for LLCE run in parallel with the CAN/LIN/FR drivers for the standard communication modules (from the NXP MCAL package).

AUTOSAR® CAN driver for LLCE was developed according to AUTOSAR® multicore concept supporting up to 2 cores.

To configure the Advanced features supported in the firmware another AUTOSAR® plugin is provided:

- LLCE_AF

Configuration of all drivers can be done with Elektrobit Tresos tool or S32 Design Studio Configuration Tool.

NXP LLCE firmware can support advanced extensions created by NXP or by the customers. Example of such advanced extensions include:

- CAN to CAN frame router
- CAN Frame ID remapping during internal routing
- Fast autonomous startup mode
- Routing received CAN frames to PCIe interface
- CAN tunnelling over Ethernet using IEEE1722a and UDP encapsulation protocol
    - CAN to Ethernet
    - Ethernet to CAN

## 2.1. LLCE Standard

LLCE enablement consist of binary images for the code running on each of the LLCE cores. Support for 16 CAN and 4 LIN communication controllers as it is expected by the AUTOSAR® 4.4 drivers on the host side.

- Timestamping of received and transmitted frames
- S32G2: Support for up to 1024 filters and 2000 message buffers
- S32G3: Support for up to 4096 filters, 1732 long message buffers and 2364 short message buffers

### 2.1.1. Release package content

- AUTOSAR® drivers for CAN, LIN, FlexRay
- Firmware binaries for the 4 cores (DTE, PPE_RX, PPE_TX, FRPE)
- LLCE firmware user guide
- LLCE host interface header files + data structures
- Sample applications (CAN, LIN, Flexray)
- Quick Start Guide
- Release notes
- Quality Package - delivered to customers for RTM releases

## 2.2. High-Level Architecture of the LLCE Firmware

### 2.2.1. Host Interface

Interface with host cores is done through the LLCE host interface. It consists of messages exchanged via shared memory, hardware FIFOs and Core2Core hardware communication module.

- Host side applications interacts with LLCE firmware by using 3 different interfaces for each type of protocols: CAN, LIN and FlexRay. In case of CAN protocol, the LLCE firmware can be used by maximum 2 different host applications simultaneously.
- The host interface for each bus is composed from independent hardware (HW) elements.
- All the source files servicing each bus behavior are compiled together and the execution is distributed between multiple internal cores

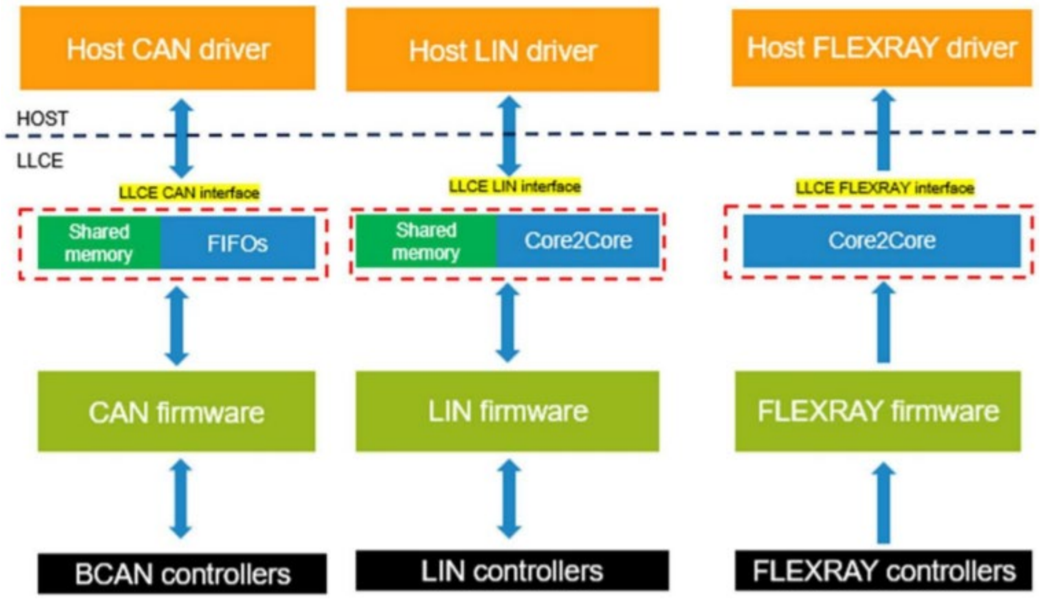

Figure 2. **LLCE interface with host**

### 2.2.2. **CAN protocol**

High level architecture of the LLCE CAN communication firmware is presented in *Figure 3*.

- LLCE CAN firmware is distributed and runs on all 4 internal cores
- Interactions between host applications and CAN firmware is done by using multiple custom interfaces composed from different shared memory areas and HW FIFOs
- HW FIFOs are used also as inter-core communication mechanism inside LLCE
- Data Transfer Engine (DTE) core run fully in polling mode in order to get all frames from all BCANs

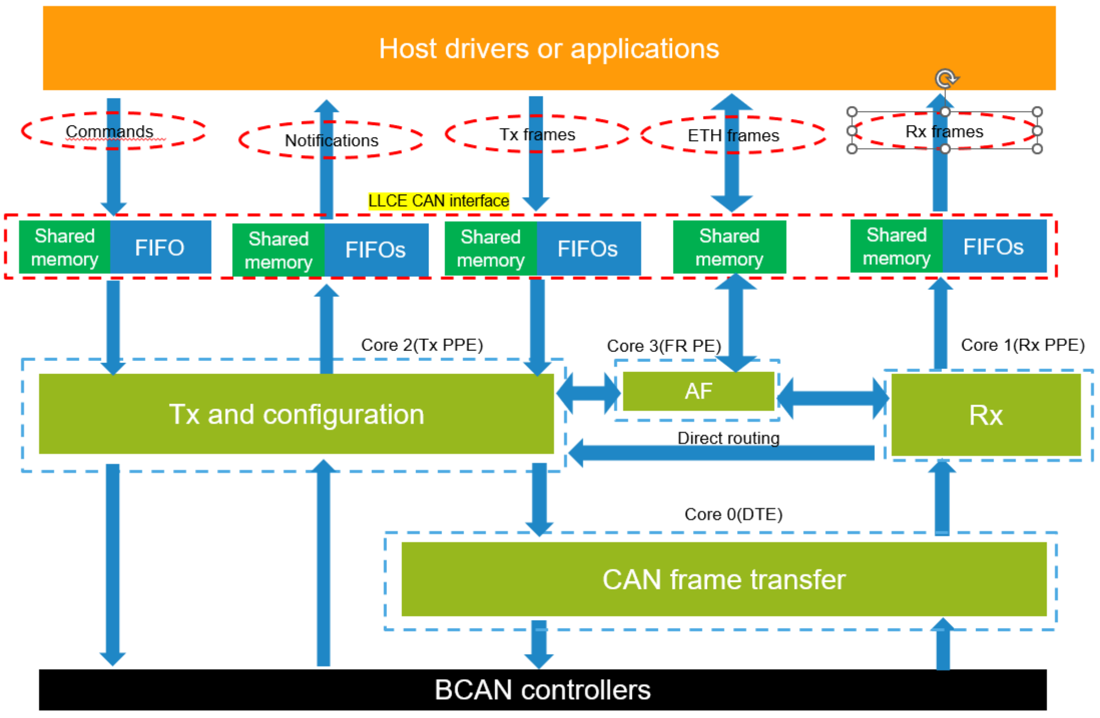

Figure 3. **LLCE Firmware architecture for CAN**

## 2.2.3. **LIN Protocol**

High level architecture of the LLCE LIN communication firmware is presented in *Figure 4*.

- LLCE LIN firmware is running fully on the Tx PPE core
- LIN firmware behave like a master or slave on the LIN bus
- LIN firmware reacts only by responding to the host commands
- Host driver writes into shared memory the command parameters and notify LIN firmware by raising a flag inside Core2Core module
- Support LIN protocol processing on the host side by forwarding  interrupts

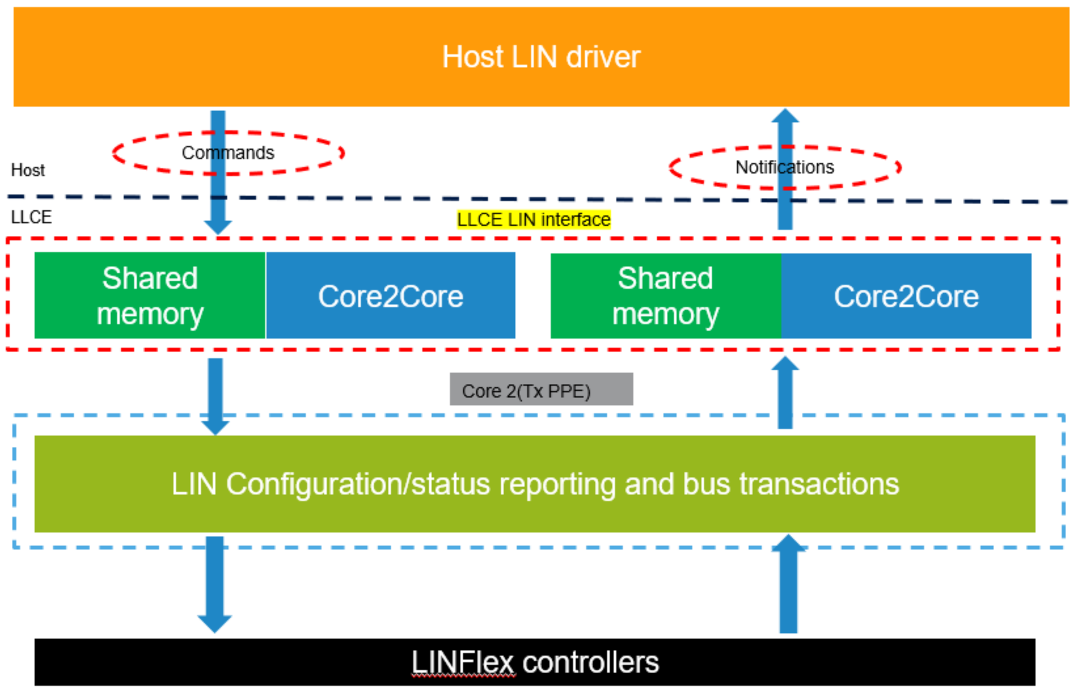

Figure 4. **LLCE Firmware architecture for LIN**

## 2.2.4. FlexRay Protocol

High level architecture of the LLCE FlexRay communication firmware is presented in *Figure 5*.

Due to the fact that the Flexray interrupts are not routed directly to the host cores, it was needed that the LLCE firmware to implement a mechanism in order to forward the Flexray interrupts to host applications.

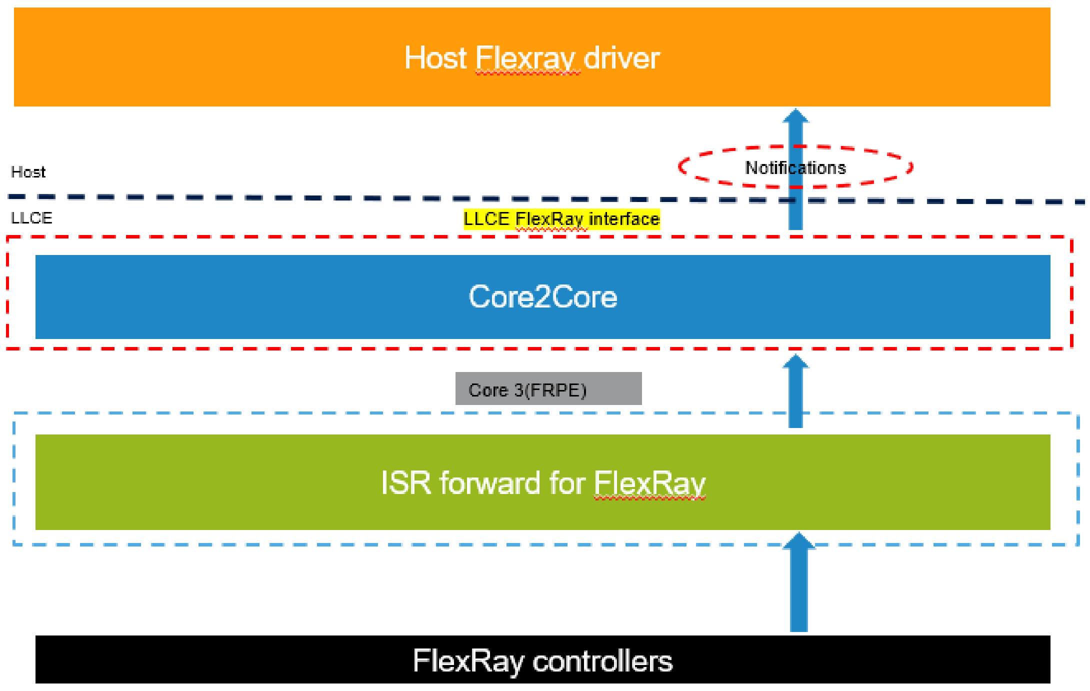

Figure 5. **LLCE Firmware architecture for FlexRay**

### 2.2.5. SPI Protocol

Due to the fact that the SPI interrupts are not routed directly to the host cores, it needed that the LLCE firmware to implement a mechanism in order to forward the SPI interrupts to host applications. This is similar with the Flexray mechanism.

# 3. Supported Targets

The software described in this document is intended to be used with NXP Semiconductors S32G2/G3 devices.

# 4. Quality, Standards Compliance and Testing Approach

LLCE Firmware product is developed according to NXP Software Development Processes that are Automotive-SPICE, ISO21434, IATF16949 and ISO 9001 compliant.

LLCE Firmware (SW) Releases starting with Beta are accompanied by SW quality packages containing at minimum the following deliverables:

- Requirements Specification and Traceability Matrix (features-design-code-test);
- Test Specification;
- Test Report;
- Static Analysis Report (MISRA, CERTC, CWE) Report;
- Code Coverage Report.

SW Testing approach is documented in LLCE Firmware Test Strategy document that contain the following information and can be shared with customers in request.

- Testing scope and objectives;
- Test levels: unit tests, unit integration tests;
- Test types: functional, non-functional, regression tests, robustness, performance tests, conformance testing (MISRA 2012);
- Test techniques: white-box, black-box tests;
- Test cases organization and prioritization;
- Test deliverables (test report, test specification, code coverage report, traceability matrix, static analysis report).

## 5. Document Information

Table 1. **Sample revision history**

| Revision number | Date | Substantive changes |
|---|---|---|
| 1 | 10/2021 | Initial release |
| 1.6 | 02/2023 | Updated for S32G3 |
| 1.7 | 10/2023 | Added new features |

Document Number: 1.7
Rev. 1.7
10/2023