NXP Semiconductors
Product Brief

Document Number: 1
Rev. 1.7, 03/2023

Inter-Platform Communication Framework Product Brief

Contents

1. Software Product Overview

Inter-Platform Communication Framework (IPCF) is a subsystem which enables applications, running on multiple homogenous or heterogenous processing cores, located on the same chip or different chips, running on different operating systems (AUTOSAR, Linux, FreeRTOS, Zephyr, etc.), to communicate over various transport interfaces (Shared Memory, etc.).

IPCF is designed for NXP embedded systems and features low-latency and tiny-footprint. It exposes a zero-copy API that can be directly used by customers for maximum performance, minimum overhead and low CPU load. The driver ensures freedom from interference between local and remote shared memory by executing all writing operations only in local memory domain. Customers can enforce memory protection for their software with XRDC/SMPU peripherals.

Customers can choose to build exactly what they need in terms of HW, OS and transport interface.

1.1. Use-cases

The following diagram illustrates some use-cases addressed by IPCF.

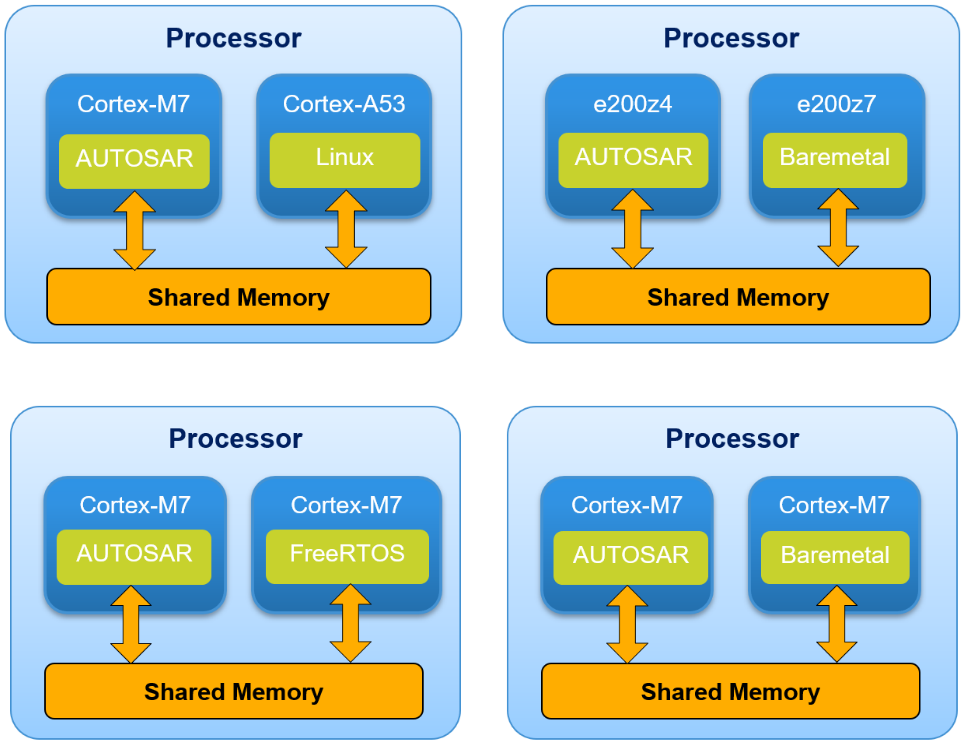

Figure 1 IPCF use cases on multiple homogenous or heterogenous processing cores

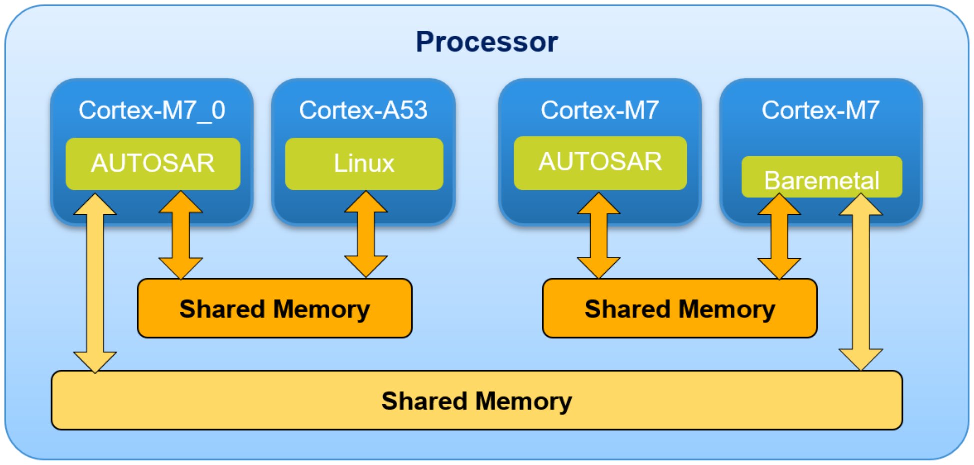

Figure 2 IPCF use case with multiple instances

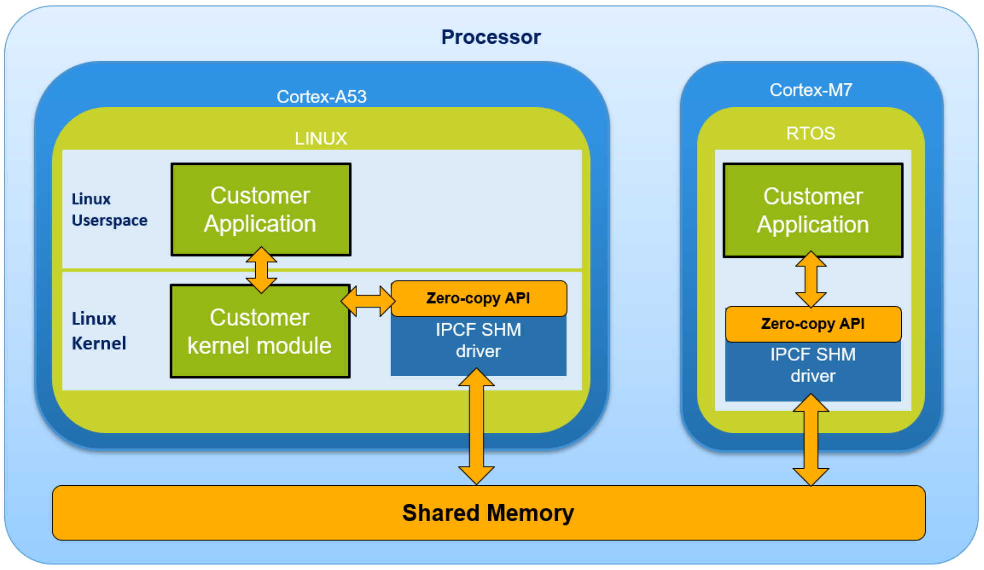

Figure 3 IPCF use case for Linux kernel

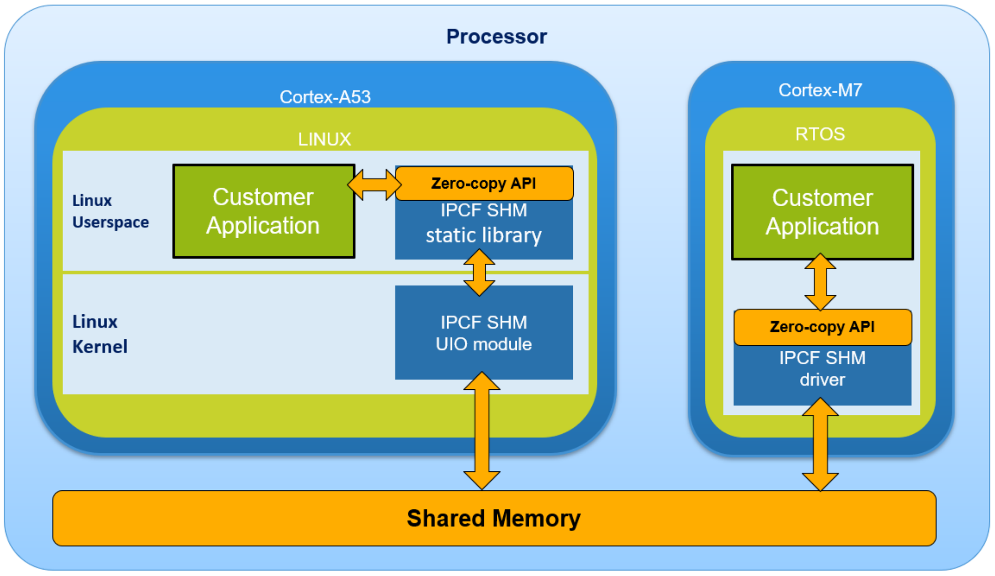

Figure 4 IPCF use case for Linux user-space

2. Software Content

IPCF software package contains the communication driver over shared memory supporting AUTOSAR, FreeRTOS, Zephyr and baremetal. For Linux support IPCF software package is published on GitHub (GitHub NXP Repo).

The driver is accompanied by sample applications which demonstrate a ping-pong message communication (for more details see the samples readme file).

IPCF software package contains a Configuration Tool component used in NXP S32 Design Studio for quick and easy to use configuration. This component is installed over NXP S32 Real Time Drivers releases and can be used with all Real Time Drivers components.

IPCF Shared Memory Driver for Linux and sample application are integrated as out-of-tree kernel modules in NXP Auto Linux BSP. The IPCF driver is provided in Yocto images from NXP Auto Linux BSP (fsl-image-auto) but can be also build manually.

IPCF Shared Memory Driver for Linux also provides a user-space static library in case the customer application requires IPCF usage from user-space.

The source code and samples of IPCF implementation for Linux drivers are published on:
https://github.com/nxp-auto-linux/ipc-shm
https://github.com/nxp-auto-linux/ipc-shm-us

IPCF software package also contains:

- Release Notes (information about release):
  - Supported platforms
  - Software dependencies
  - Validated compilers
  - Instructions about installation steps
  - New features
  - Known limitations
  - Licensing and support

- IPCF Driver User Manual:
  - Installation instruction for different integration
  - Driver usage, compile and configuration for different OSes and platforms
  - Instructions to compile, build and run the sample application
  - Describe driver APIs

- Quality Package - delivered to customers for RTM releases

2.1. Architecture

IPCF driver contains next layers:

- Shared memory generic implementation that is HW and OS agnostic
- Queue component implementation used in IPCF driver
- HW abstraction component: abstraction over various HW IP modules (MSCM, INTC, MRU)
- OS abstraction component: OS agnostic API for common OS services

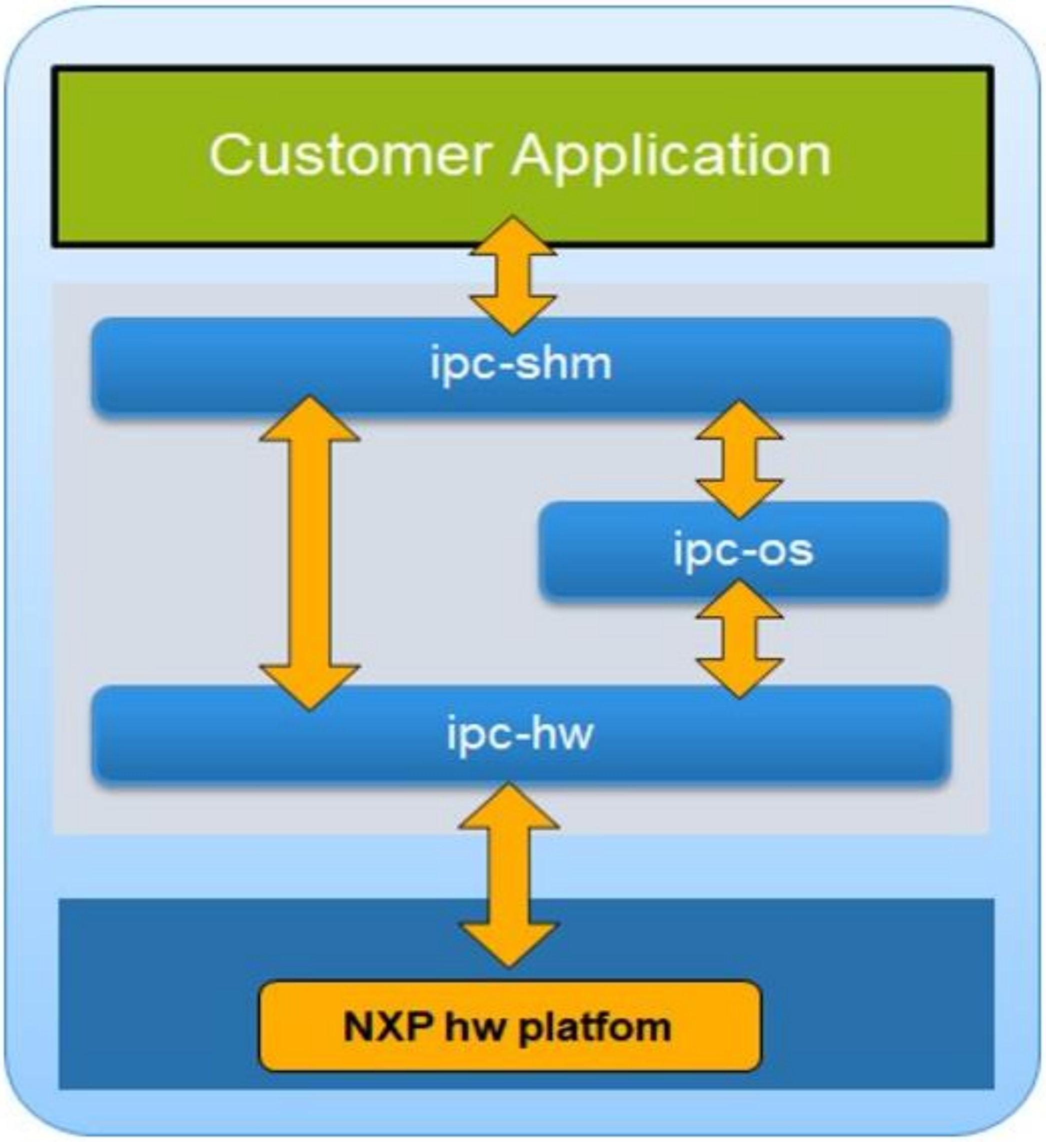

Figure 5 IPCF System Architecture

2.2. Details

IPCF driver uses for buffer management:

- unmanaged channel data flow: buffer management is disabled, and application owns the entire channel memory; use-case example: video streaming or non-critical data exchange
- managed channel data flow: memory is split in buffer pools and buffer management is controlled by driver; use-case example: CAN forwarding or flash update

IPCF driver supports the following inter-core notification methods:

- intercore interrupts or MRU (hardware notification)
- polling method (sending and receiving is managed by user)

IPCF driver reduces the receive interrupt overhead with interrupt coalescing technique, avoiding storming interrupt. When an receive interrupt is triggered the code disables the interrupt, processes all buffer descriptors in receive FIFO and then reenables the interrupt.

3. Supported Targets

The software described in this document is intended to be used with the following microcontroller and microprocessors devices of NXP Semiconductors:

- Vehicle networking : S32G2, S32G3
- ADAS Vision: S32V2xx
- ADAS Radar: S32R45, S32R294, S32R41, SAF85XX
- GPIS: S32K3xx
- VDS: S32Z2, S32E2

4. Quality, Standards Compliance and Testing Approach

IPCF product is developed according to NXP Software Development Processes that are Automotive-SPICE (tailored for open source), ISO26262, IATF16949 and ISO9001 compliant.

The compilers used for driver validation are listed in the release notes document.

5. Document Information

Revision	Date	Changes Description
0.1	26-Jul-2018	Initial version
1.0	29-Aug-2018	Minor updates + bump rev to 1.0
1.1	16-July-2019	Updates for RRU2
1.2	31-Aug-2020	Updated supported targets
1.3	03-Dec-2020	Updated supported targets
1.4	03-Feb-2021	Updated diagrams and content in chapter 2
1.5	06-Jun-2022	Updated supported targets and figure 5
1.6	22-Nov-2022	Updated supported targets R41 & SAF85xx
1.7	2-Mar-2023	Updated supported targets & location for Linux support

Document Number: 1
Rev. 1.7
03/2023