



## Linux Mailbox framework overview



---

## Contents

---

---



A quality version of this page, approved on *30 January 2020*, was based off this revision.

This article gives information about the Linux<sup>®</sup> mailbox framework. The mailbox framework is involved in interprocessor communication in heterogeneous multicore systems.

## Contents

1 Framework purpose .....	4
2 System overview .....	5
2.1 Component description .....	5
2.2 API description .....	5
3 Configuration .....	6
4 Device tree configuration .....	7
5 How to trace and debug the framework .....	8
5.1 How to trace .....	8
6 References .....	8



---

## 1 Framework purpose

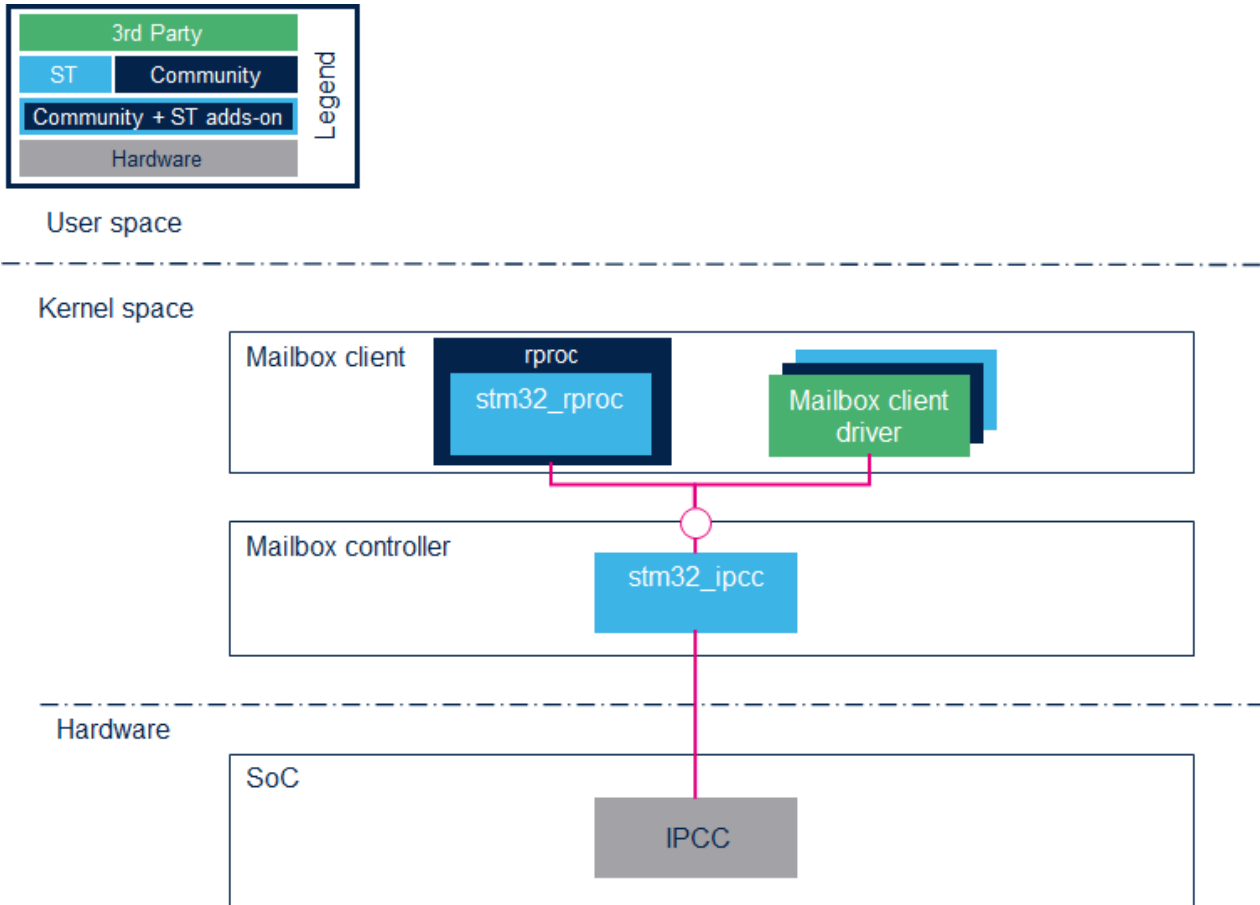
---

The mailbox is used in interprocessor communication to exchange messages or signals between the host and the coprocessor cores. The mailbox framework is based on:

- A **mailbox controller** that is platform dependent:
  - It is in charge of configuring and handling IRQ from the IPCC peripheral.
  - It provides a generic API to the mailbox client.
- A **mailbox client** that is in charge of the message to send or receive.

A general presentation of the mailbox framework is available in the Linux mailbox documentation <sup>[1]</sup>.

## 2 System overview



### 2.1 Component description

- **Mailbox controller**  
The mailbox controller is the `stm32_ipcc`. It configures and controls the IPCC peripheral
- **Mailbox client**  
The user can define his own mailbox client.  
For example, the `RPMMsg` framework uses mailbox for the interprocessor communication.  
In this case the mailbox client is the `remoteproc` driver that forwards services from/to the `RPMMsg` framework.

### 2.2 API description

The APIs are described in the Linux documentation:

- Mailbox client API <sup>[2]</sup>
- Mailbox controller API <sup>[3]</sup>



---

### 3 Configuration

---

Activate **stm32 IPCC** mailbox in kernel configuration using the Linux Menuconfig tool: [Menuconfig](#) or [how to configure kernel](#)

```
Device drivers --->
  *- Mailbox Hardware Support --->
    <*> STM32 IPCC Mailbox
```



## 4 Device tree configuration

The mailbox device node must be declared and enabled in the Linux kernel device tree. Here is an extract of the STM32MP1 evaluation board device tree:

```
ipcc: mailbox@4c001000 {
    compatible = "st,stm32-ipcc";
    #mbox-cells = <1>;
    reg = <0x4c001000 0x400>;
    interrupts-extended = <&intc GIC_SPI 100 IRQ_TYPE_NONE>,
                        <&intc GIC_SPI 101 IRQ_TYPE_NONE>,
                        <&exti 62 1>;
    interrupt-names = "rx", "tx", "wakeup";
    clocks = <&rcc_clk IPCC>;
    wakeup-source;

    Status = "okay";
};
```

Then client has to reserve channels. Here is an example of channel allocation for the remoteproc node:

```
&m4_rproc {
    memory-region = <&ipc_share>;
    mbox-names = <&ipcc 0>, <&ipcc 1>, <&ipcc 2>;
    mbox-names = "vq0", "vq1", "init_shdn";
    status = "okay";
};
```



---

## 5 How to trace and debug the framework

---

### 5.1 How to trace

Dynamic debug traces can be added using the following commands:

```
echo -n 'file stm32-ipcc.c +p' > /sys/kernel/debug/dynamic_debug/control  
echo -n 'file mailbox.c +p' > /sys/kernel/debug/dynamic_debug/control
```

---

## 6 References

---

- [Linux Mailbox documentation](#)
- [Mailbox client API](#)
- [Mailbox controller API](#)

Linux® is a registered trademark of Linus Torvalds.

Inter-Processor Communication Controller

Application programming interface

Remote Processor Messaging

Generic Interrupt Controller

Serial Peripheral Interface