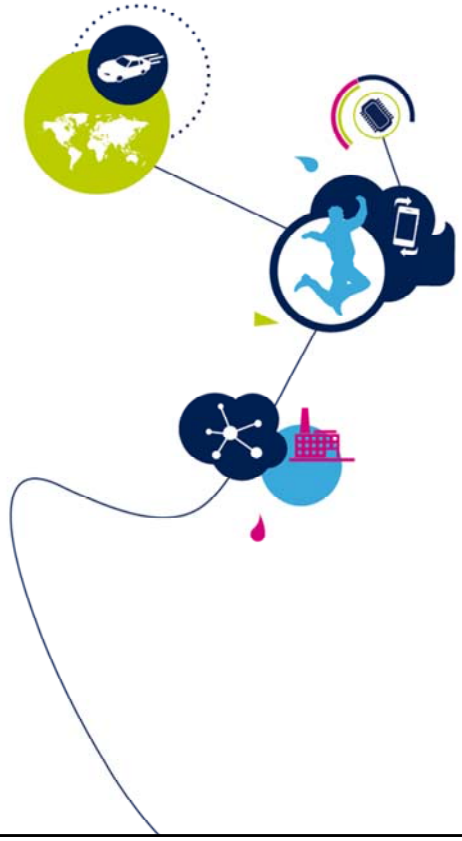
STM32MP1 - IPCC

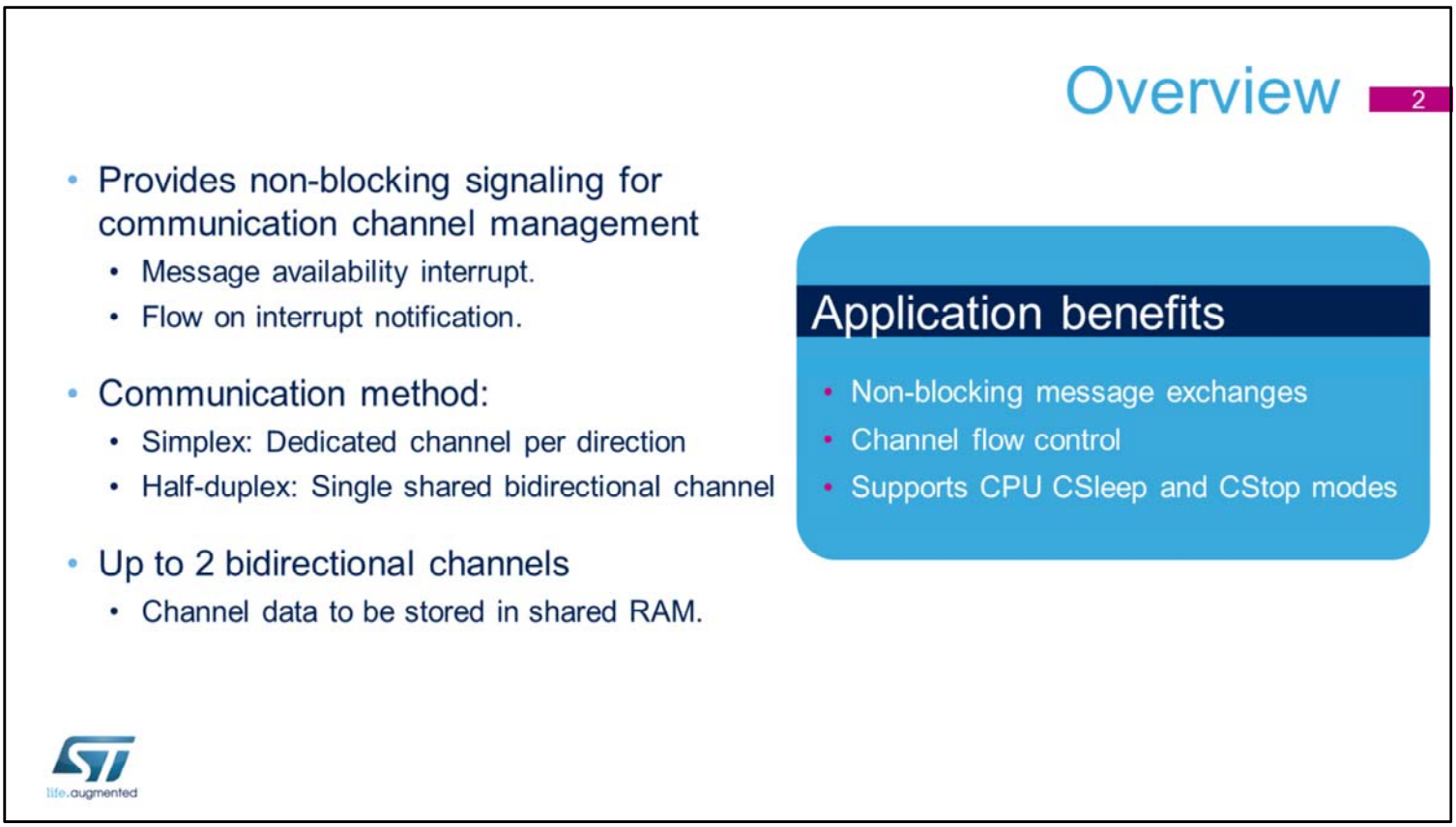
Inter-Processor communication controller

Revision 1.0



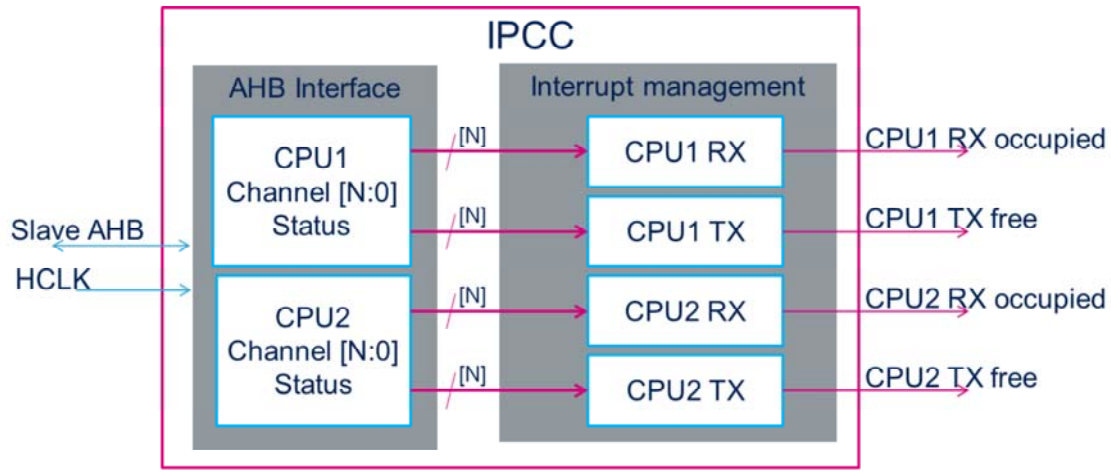
life.augmented

Hello, and welcome to this presentation of the STM32 Inter-Processor communication controller (IPCC) module. It covers the main features of the module which is used to signal message exchange between CPUs.



The Inter-Processor communication controller module integrated inside STM32 products provides interrupt signaling allowing microcontrollers to exchange messages in a non-blocking way. This module allows for simplex communication, where a dedicated channel is used to send a message from Processor A to Processor B. It also allows for half duplex (message - response) communication using a single shared channel to communicate between Processor A and Processor B. Applications benefit from non-blocking interrupt-based message exchanges and channel flow control. The Inter-Processor communication controller is able to wake up a CPU from CSleep, CStop and CStandby modes.
The channel data location is not part of the Inter-Processor communication controller and is stored in shared memory.

Block diagram



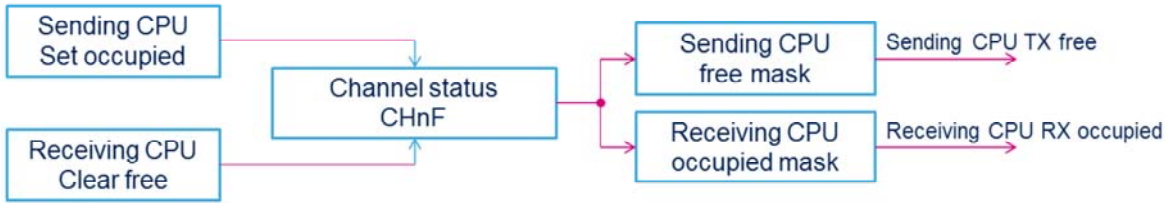
The Inter-Processor communication controller is an AHB slave module. It consists of an "AHB interface" containing the registers and an “Interrupt management” part.

Each channel has a single status flag to indicate the Send status for one CPU and the Receive status for the other CPU. Each CPU has its own status mask and set or clear register bit for each channel. The register area is split into 2 regions, one per CPU. Each region contains the registers associated with the CPU, preventing read-modify-write access conflicts.

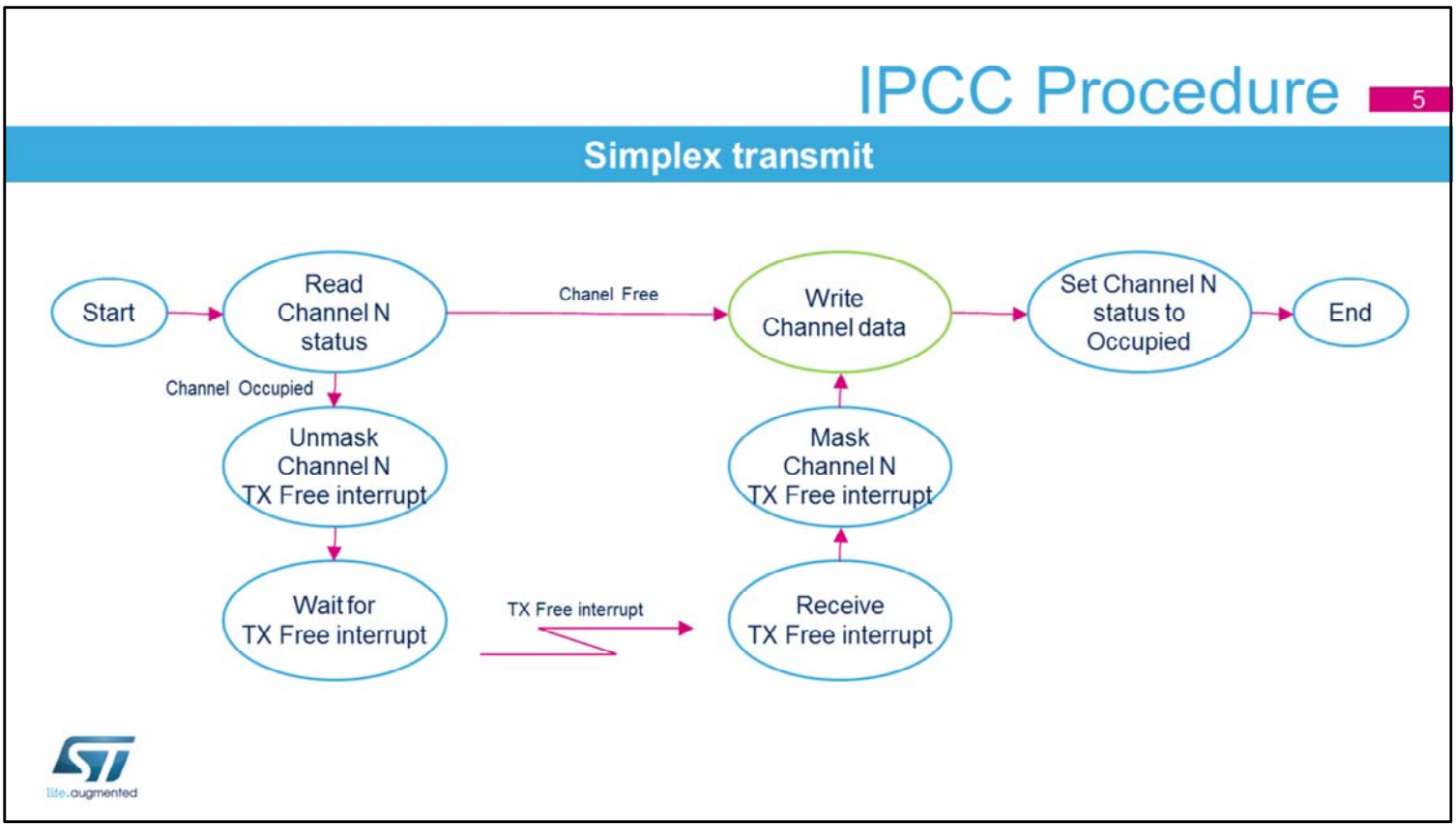
Dedicated interrupts are provided for each CPU.

Channel management

- Each channel has the following functionalities:
  - an associated direction either:
    - sent from CPU1 and received by CPU2 OR sent from CPU2 and received by CPU1
  - a single read-only channel status flag (bit CHnF)
    - which can be set to Occupied by the sending CPU (bit CHnS)
    - and can be cleared to Free by the receiving CPU (bit CHnC)
  - interrupt masks
    - a sending CPU channel free mask (bit CHnFM)
    - a receiving CPU channel occupied mask (bit CHnOM)

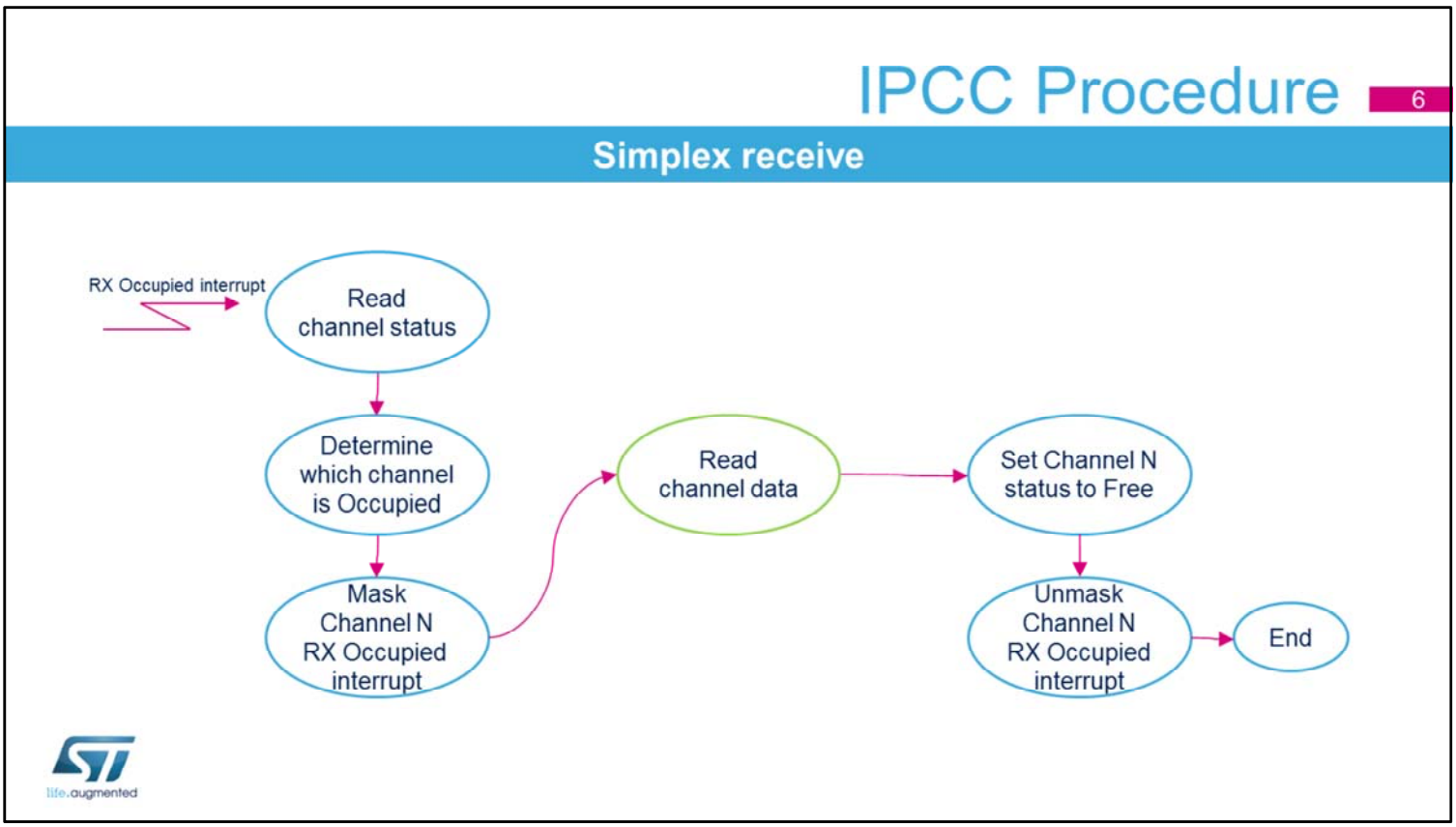


A channel is associated with a direction, from a sending CPU to a receiving CPU.
The sending CPU can signal a channel to be occupied by setting the channel status to Occupied using its Set Channel N register bit (CHnS).
If the receiving CPU has unmasked its Channel Occupied interrupt in its Channel N Occupied Mask register bit (CHnOM), an RX Occupied interrupt (message available) is generated for the receiving CPU.
The receiving CPU can signal a channel to be free by setting the channel status to Free using its Clear Channel N register bit (CHnC).
If the sending CPU has unmasked its Channel Free interrupt in its Channel N Free Mask register bit (CHnFM), a TX Free interrupt (flow on) is generated for the sending CPU.

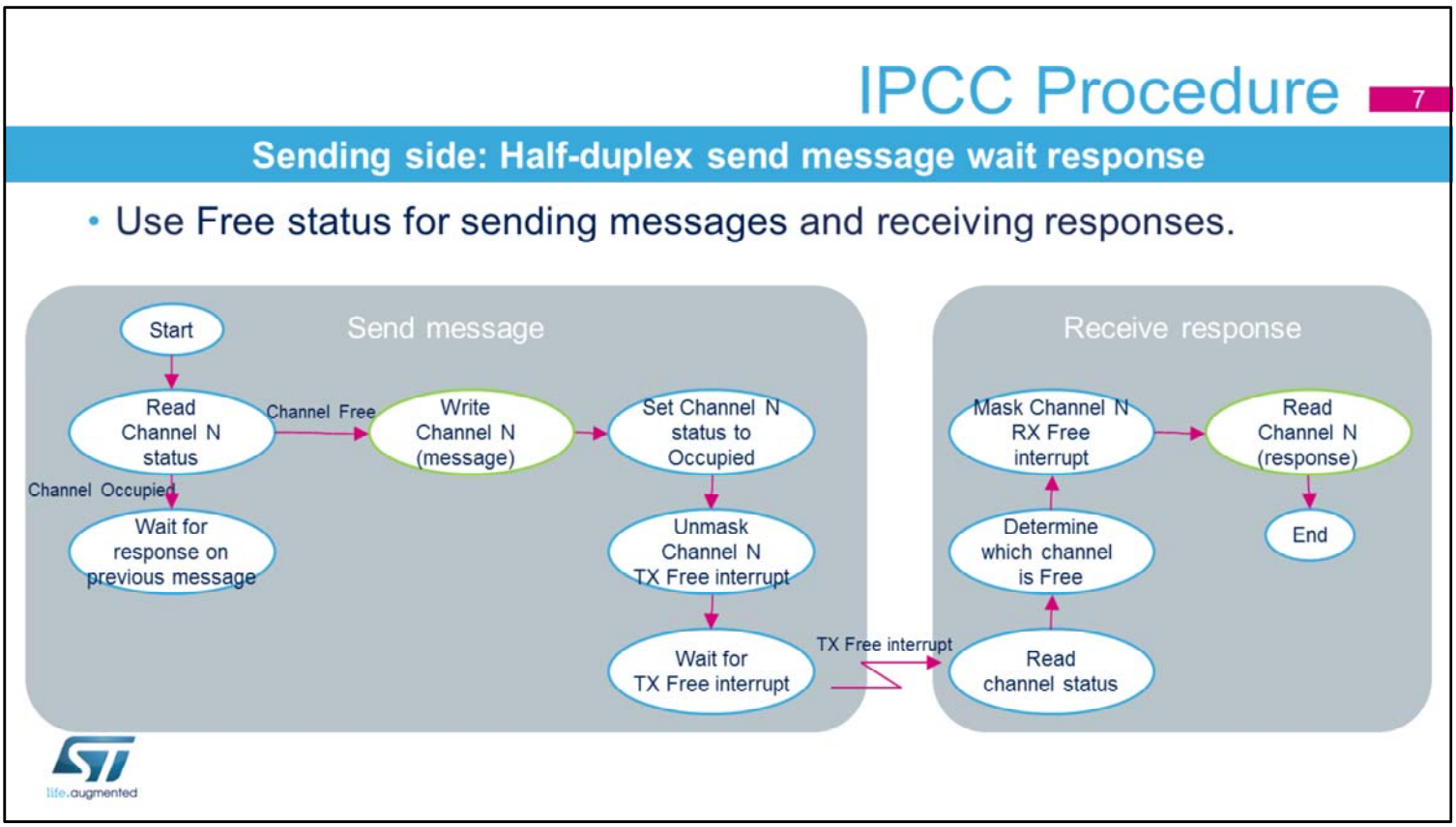


The simplex procedure allows the transfer of a message from a sending side to a receiving side via a dedicated channel.
To transmit a message using the simplex procedure, the Channel Status flag is checked. When the Channel Status flag indicates Channel Occupied (flow off), i.e. due to the receiving side not having freed the channel data buffer from a previous message, the Channel Free interrupt is unmasked. Once the channel is freed by the receiving side, a Channel Free interrupt (flow on) is generated. When the Channel Free interrupt is generated, the Channel Free interrupt is masked and the message can be written in the channel data buffer. Subsequently the Channel Status flag is set to Occupied, which triggers a Channel Occupied interrupt for the receiving side.
When the Channel Status flag is checked to be free, a

message can directly be written in the channel data buffer.



When a Channel Occupied interrupt is generated, the receiving side determines which channel is occupied and mask the appropriate Channel Occupied interrupt. Subsequently, the message can be read from the channel data buffer. Once read, the channel status flag is cleared to Free (flow on) and the Channel Occupied interrupt is unmasked.



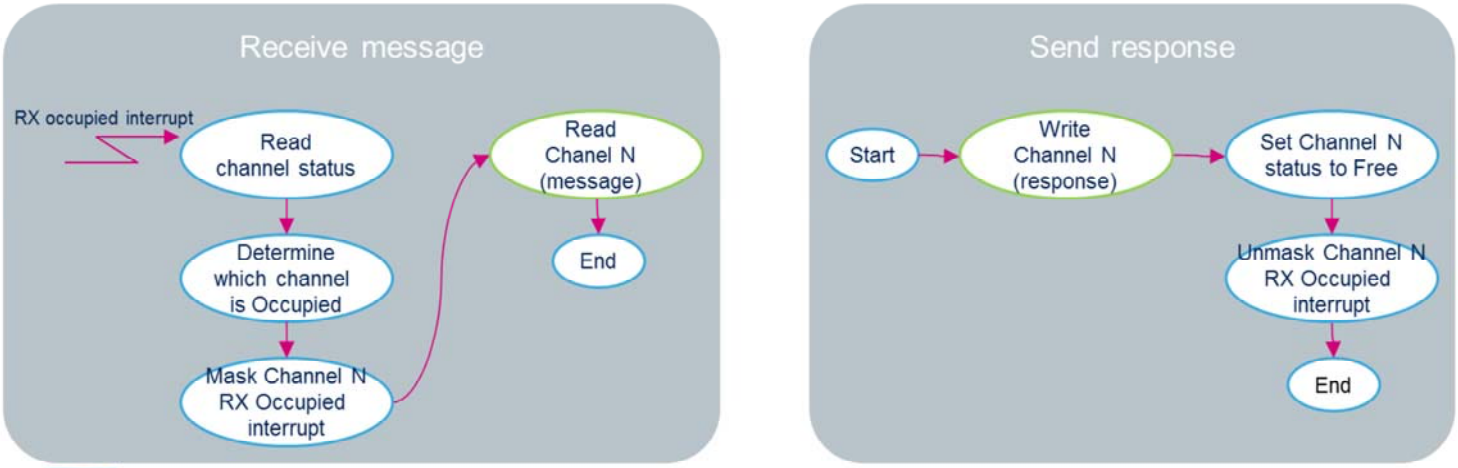
The half-duplex procedure allows the transfer of a message from a sending side to a receiving side, followed by a response sent from the receiving side back to the sending side, using a single shared buffer.
In the half-duplex procedure, the sending side will first check the Channel Status flag. If the Channel Status flag indicates the channel is occupied (flow off), i.e. due to the receiving side not having sent yet a response to a previous message, the sending side waits for the response (a software flag).
When the channel is free, the message can be written in the channel data buffer. Subsequently the channel status flag is set to Occupied, which triggers a Channel Occupied interrupt for the receiving side and the Channel Free interrupt is masked. The Channel Free interrupt indicates the availability of the response sent by the receiving side.

When a Channel Free interrupt (response ready) is generated, the sending side determines which channel is freed and masks the corresponding Channel Free interrupt. Subsequently, the response can be read from the channel data buffer.

IPCC Procedure

Receiving side: Half Duplex receive message send response

- Use occupied status for receiving message and sending response.



When a Channel Occupied interrupt (message available) is generated, the receiving side determines which channel is occupied and masks the corresponding Channel Occupied interrupt. Subsequently, the message can be read from the channel data buffer. The channel will only be freed once the receiving side has sent the response to the channel data buffer. Once the response is received in the channel data buffer, the channel status flag is cleared to Free (response ready) and the Channel Occupied interrupt is then unmasked.

Interrupts

Interrupt event	Description
CPU1 TX free	CPU1 sending Channel Status cleared to Free.
CPU1 RX occupied	CPU1 receiving Channel status set to Occupied.
CPU2 TX free	CPU2 sending Channel Status cleared to Free.
CPU2 RX occupied	CPU2 receiving Channel Status set to Occupied.

life.augmented

Here is an overview of Inter-Processor communication controller interrupt.
Each CPU has a TX Free interrupt associated to its own sending channel.
Each CPU has an RX Occupied interrupt associated to the sending CPU channel.

Low-power modes

System Mode	Description
Run	Active. Peripheral interrupts cause a CPU to exit CSleep, CStop and CStandby modes.
Sleep	Frozen. Peripheral registers content is kept.
Stop/ LP-Stop	Frozen. Peripheral registers content is kept.
LPLV-Stop	Frozen. Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting system Standby mode.

life.augmented

Here is an overview of the peripheral status at specific low-power configuration modes. The Inter-Processor communication controller is not able to change states in Sleep and Stop modes. In Standby mode, the Inter-Processor communication controller content is lost. The Inter-Processor communication controller will be in Run mode whenever a CPU is in Run mode. The Inter-Processor communication controller is able to interrupt and wake up a CPU in CRun, CSleep, CStop and CStandby modes.

Related peripherals

- This is a list of peripherals related to the IPCC controller. Please refer to these peripheral trainings for more information if needed.
  - Reset and clock control (RCC)
  - Extended Interrupt and event controller (EXTI)
  - Interrupts (NVIC)

Here is a list of peripherals related to the Inter-Processor communication controller module. Users should be familiar with all the relationships between these peripherals to correctly configure and use the Inter-Processor communication controller module.