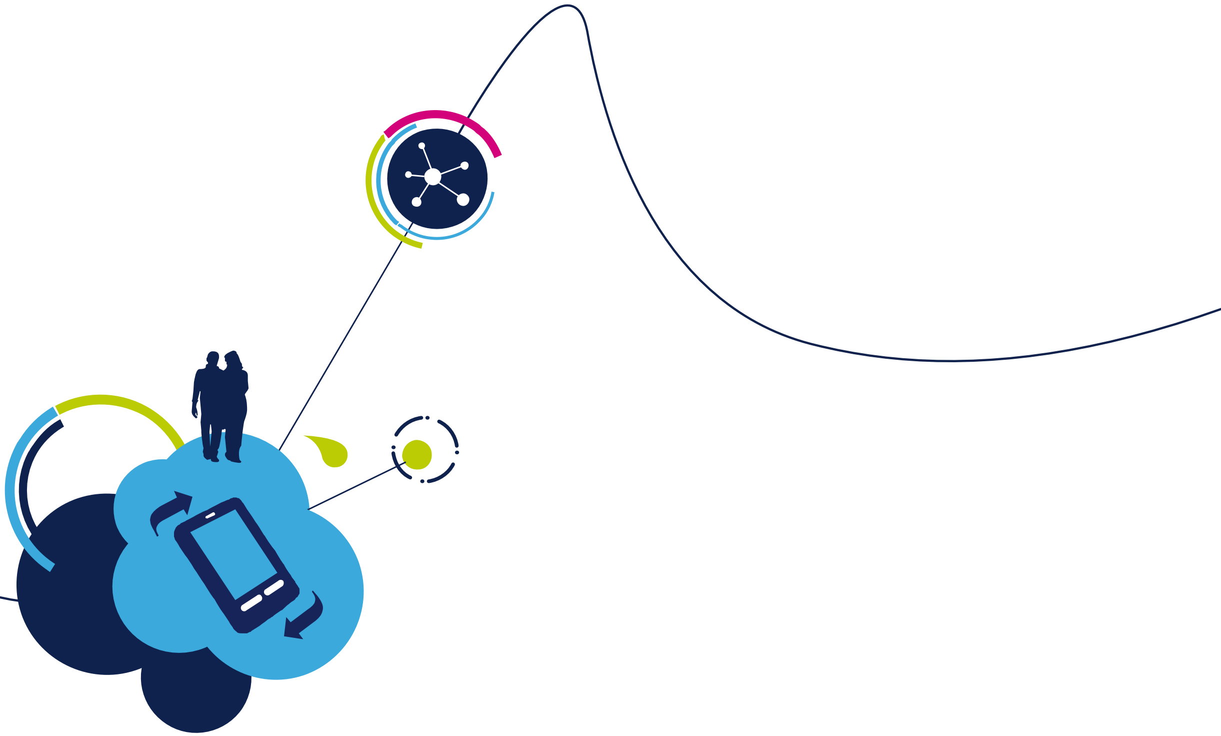
STM32

Training Hands on and Tools overview



2.1.1 UART Poll lab

2.1.1 Simple UART communication

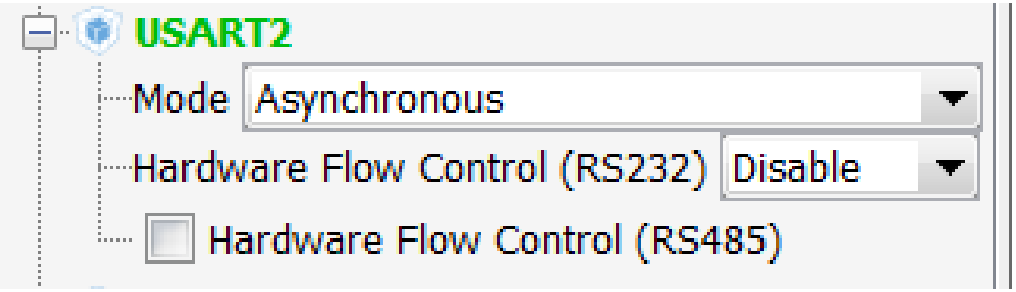
- Objective
  - Learn how to setup UART in CubeMX
  - How to Generate Code in CubeMX and use HAL functions
  - Work in pairs, one will create transmitter and second receiver
- Goal
  - Configure UART in CubeMX and Generate Code
  - Learn how to send and receive data over UART without interrupts
  - Verify the correct functionality

2.1.1 Simple UART communication

- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F0 > STM32F030 > LQFP64 > STM32F030R8
- Pin selection
  - We are looking for free pins where is possible to create wire loopback connection

2.1.1 Simple UART communication

- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F0 > STM32F030 > LQFP64 > STM32F030R8
- CubeMX UART selection
  - Select USART2 in asynchronous mode
  - Select PA2 and PA3 for USART2 if weren't selected

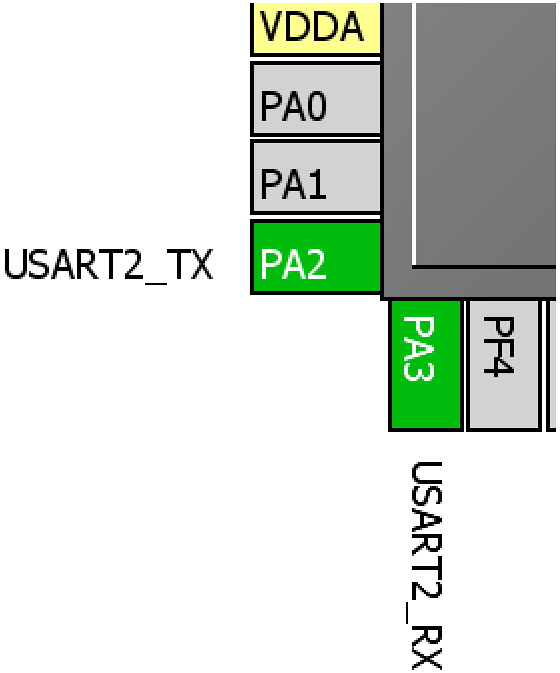


USART2

Mode: Asynchronous

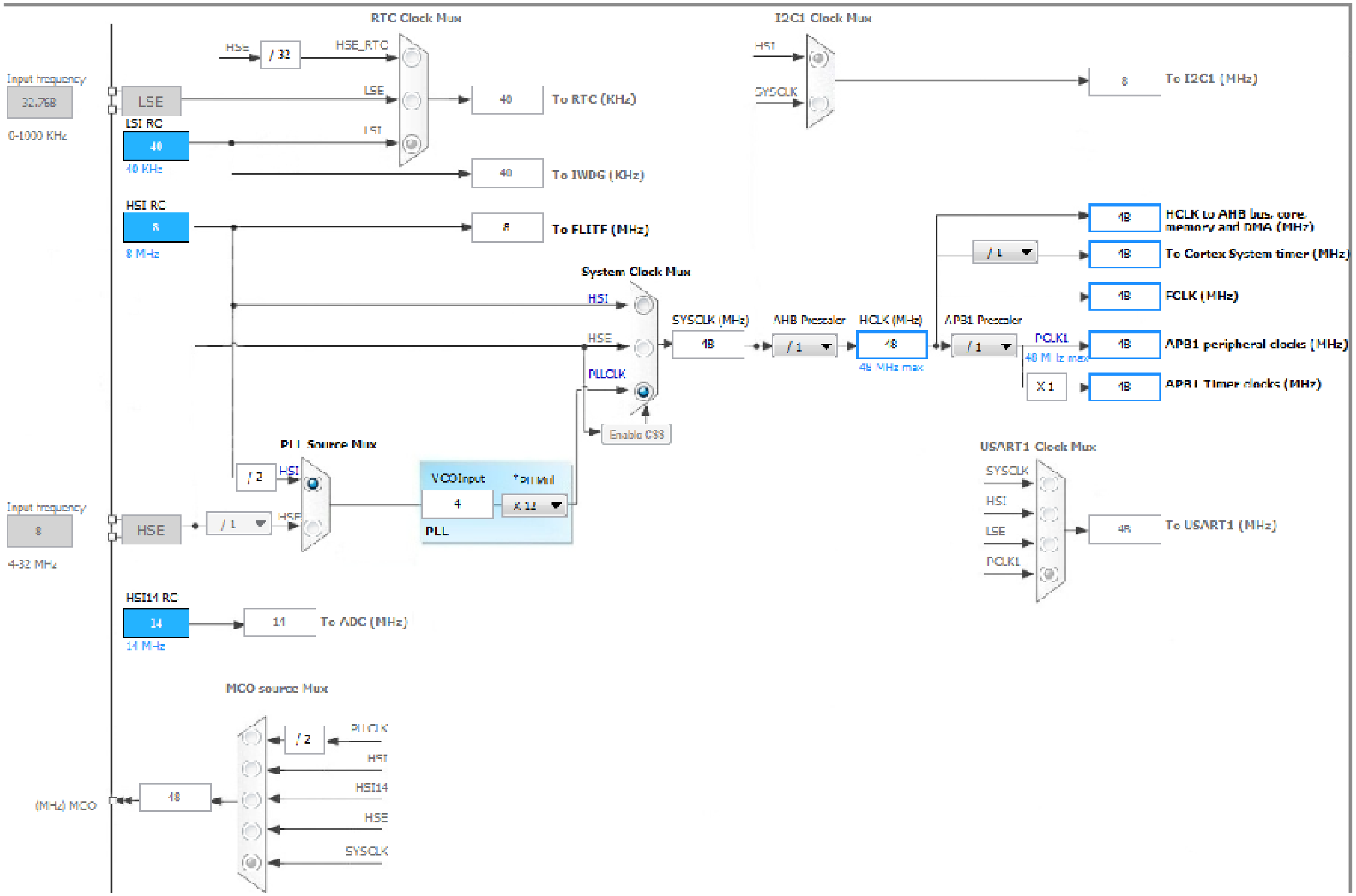
Hardware Flow Control (RS232): Disable

□ Hardware Flow Control (RS485)



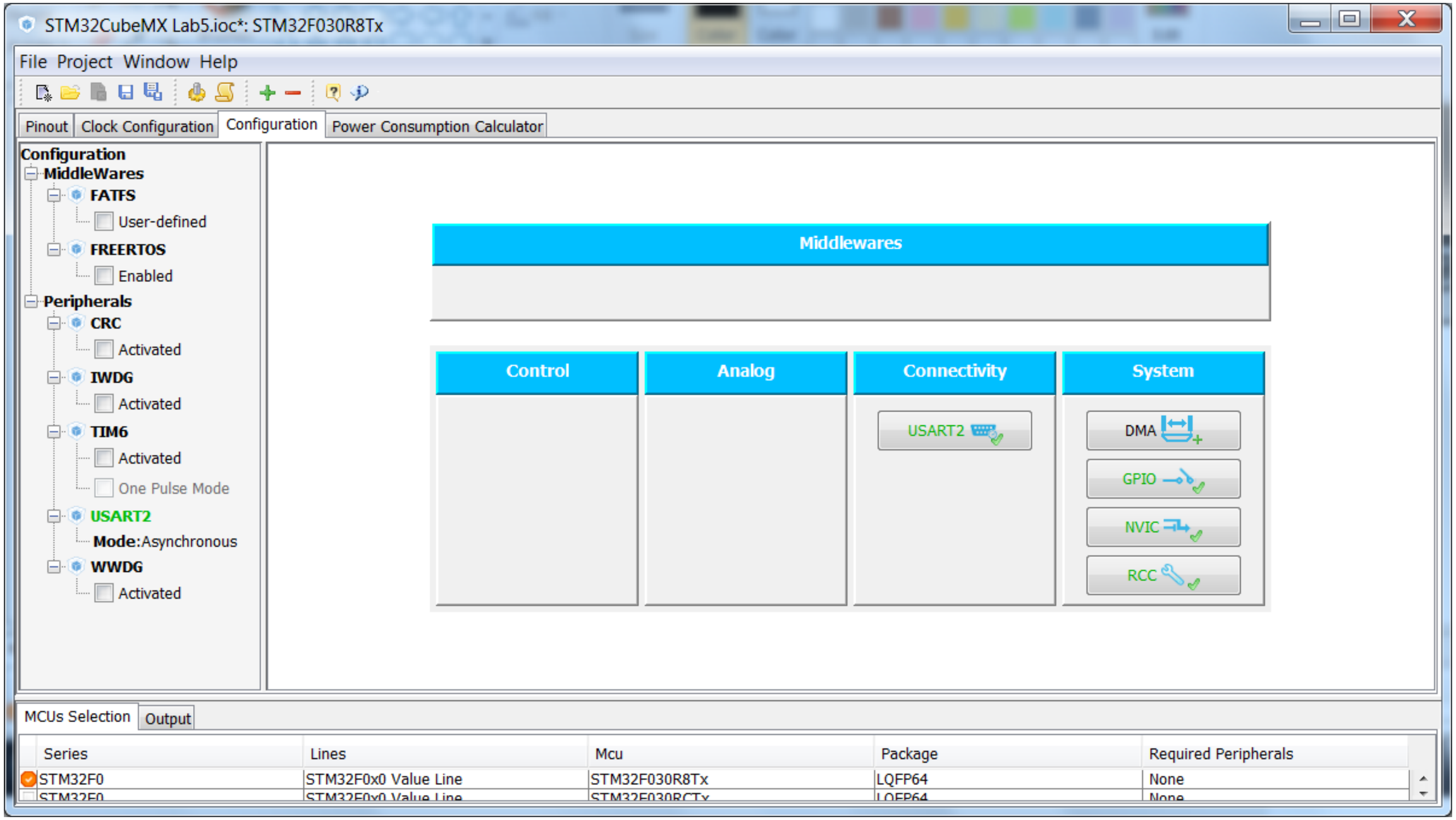
2.1.1 Simple UART communication

- In order to run on maximum frequency, setup clock system
- Details in lab 0



2.1.1 Simple UART communication

- CubeMX UART configuration
  - Tab>Configuration>Connectivity>USART2

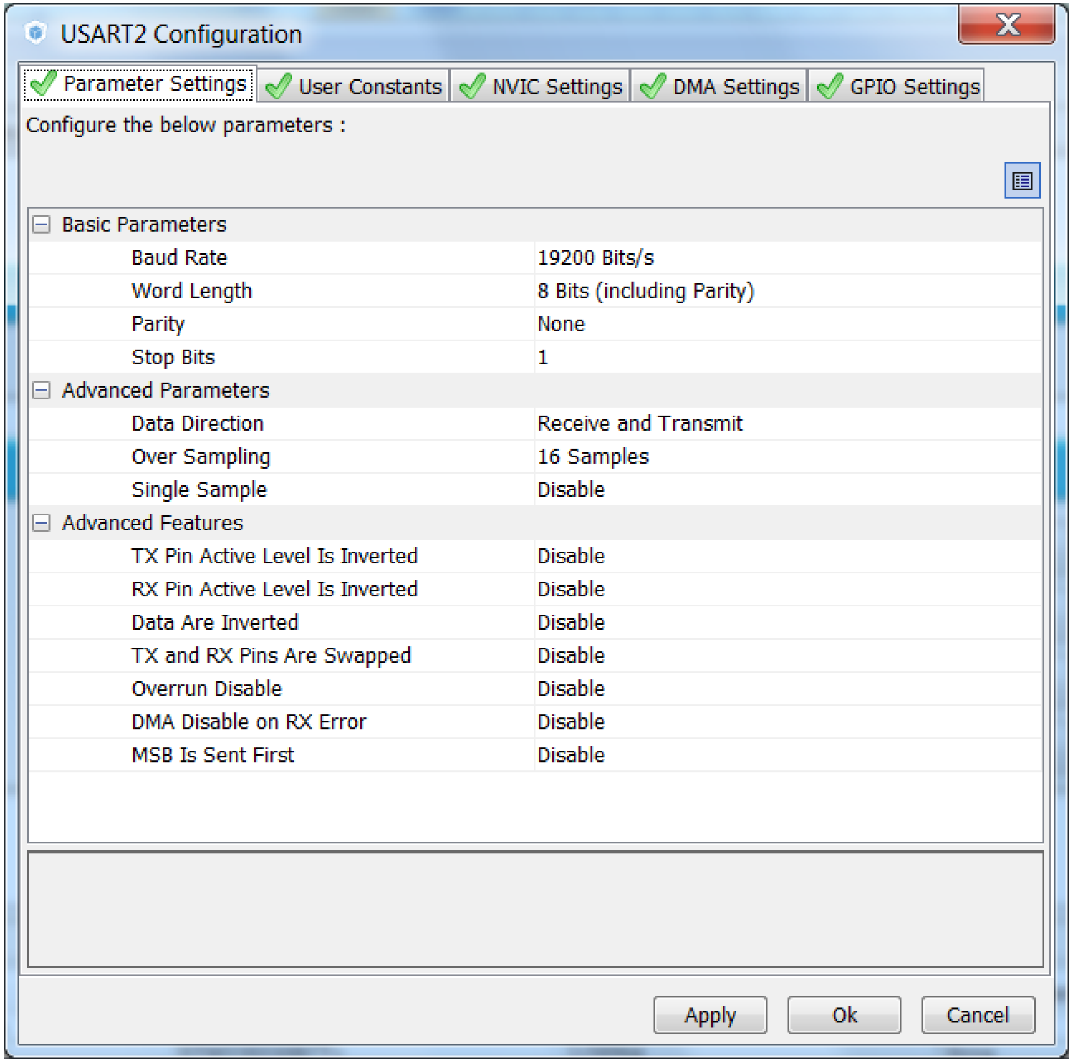


Configuration
- MiddleWares
  - FATFS
    - □ User-defined
  - FREERTOS
    - □ Enabled
- Peripherals
  - CRC
    - □ Activated
  - IWDG
    - □ Activated
  - TIM6
    - □ Activated
    - □ One Pulse Mode
  - USART2
    - **Mode:**Asynchronous
  - WWDG
    - □ Activated

Series	Lines	Mcu	Package	Required Peripherals
STM32F0	STM32F0x0 Value Line	STM32F030R8Tx	LQFP64	None
STM32F0	STM32F0x0 Value Line	STM32F030RCTx	LQFP64	None

2.1.1 Simple UART communication

- CubeMX USART configuration check:
  - BaudRate
  - World length
  - Parity
  - Stop bits
  - Data direction
  - Oversampling

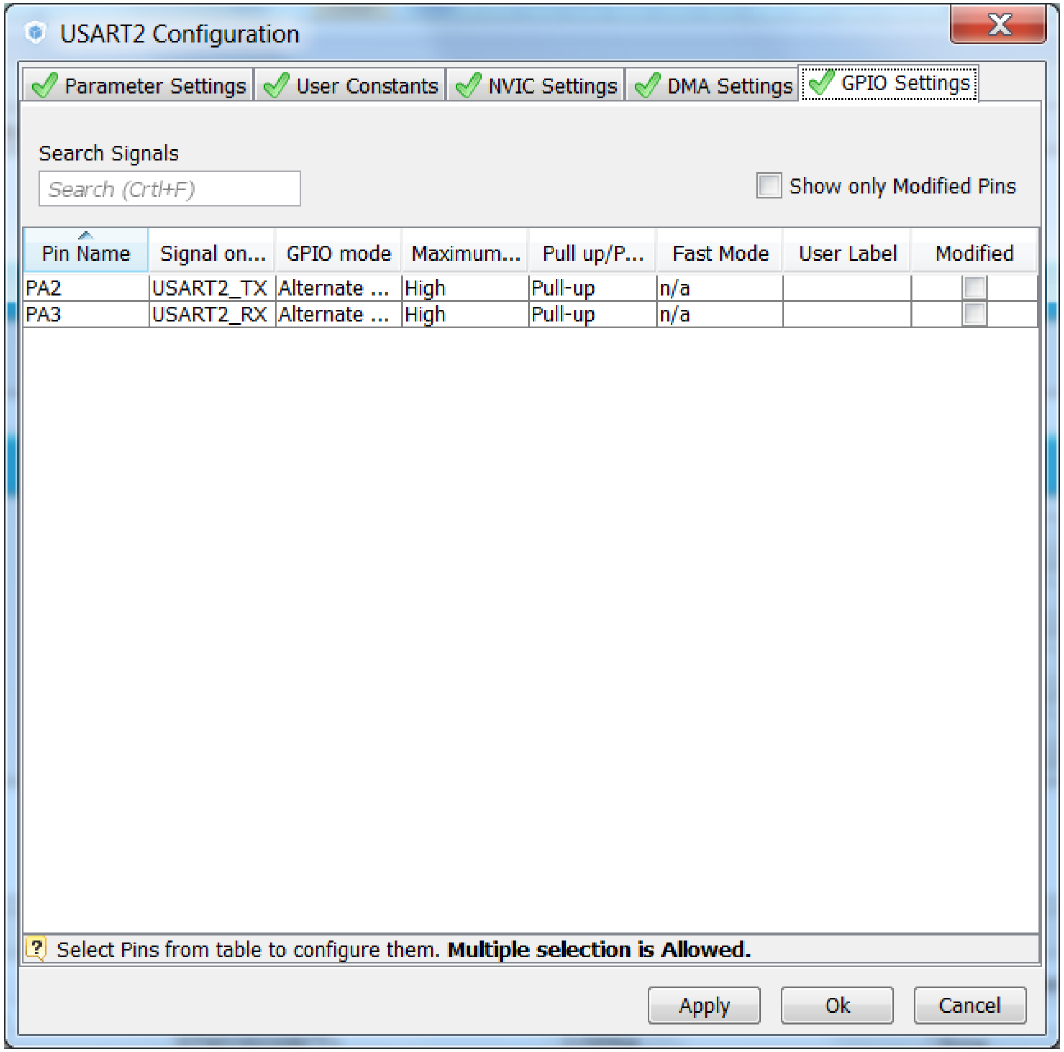


Configure the below parameters :

⊟ Basic Parameters	
Baud Rate	19200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
⊟ Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable
⊟ Advanced Features	
TX Pin Active Level Is Inverted	Disable
RX Pin Active Level Is Inverted	Disable
Data Are Inverted	Disable
TX and RX Pins Are Swapped	Disable
Overrun Disable	Disable
DMA Disable on RX Error	Disable
MSB Is Sent First	Disable

2.1.1 Simple UART communication

- CubeMX USART GPIO configuration check:
  - On high baud rate set the GPIO speed to HIGH
  - Set the HIGH output speed Button OK



USART2 Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

Search Signals

Search (Crtl+F)

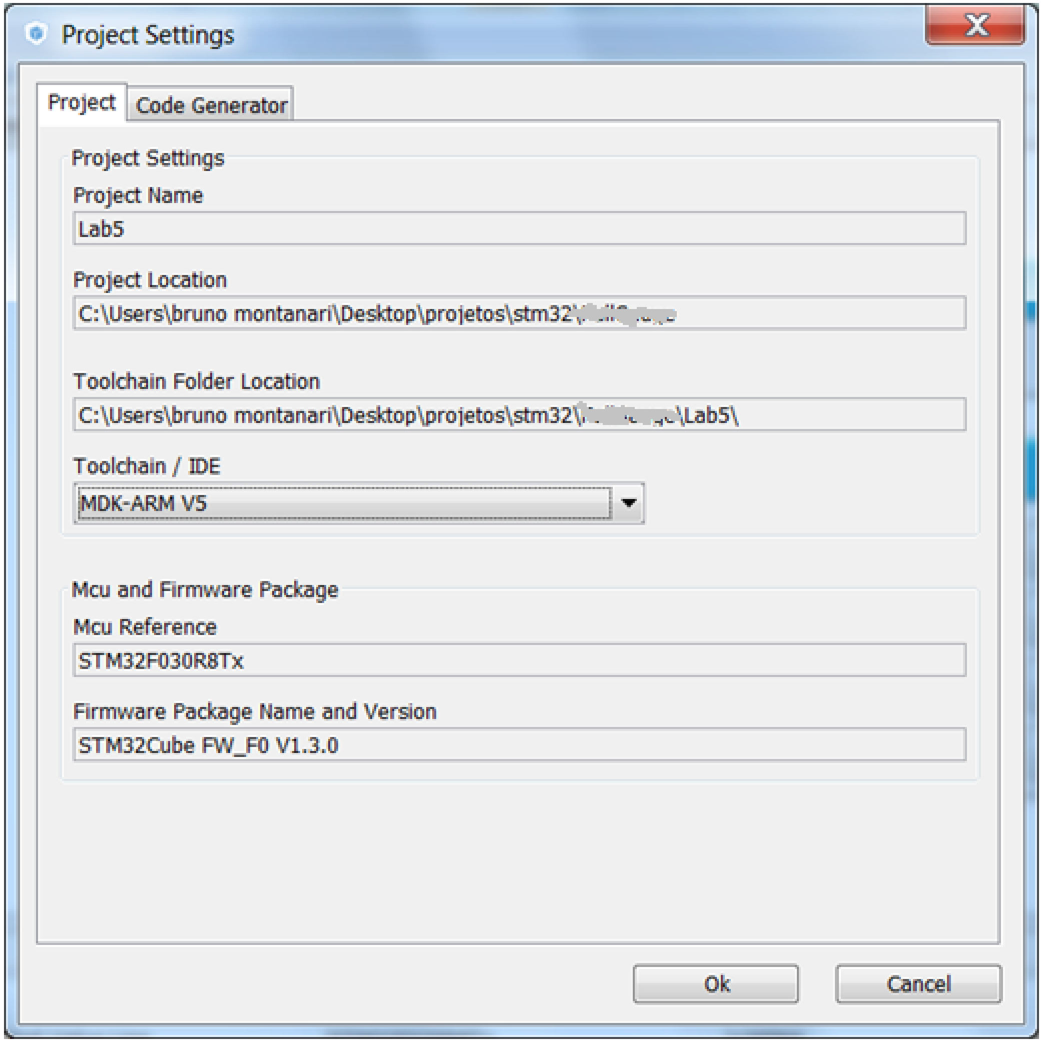
□ Show only Modified Pins

Pin Name	Signal on...	GPIO mode	Maximum...	Pull up/P...	Fast Mode	User Label	Modified
PA2	USART2_TX	Alternate ...	High	Pull-up	n/a		□
PA3	USART2_RX	Alternate ...	High	Pull-up	n/a		□

Select Pins from table to configure them. **Multiple selection is Allowed.**

2.1.1 Simple UART communication

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Now we can Generate Code
  - Menu > Project > Generate Code



Project Settings

Project Name: Lab5

Project Location: C:\Users\bruno montanari\Desktop\projetos\stm32\[redacted]

Toolchain Folder Location: C:\Users\bruno montanari\Desktop\projetos\stm32\[redacted]\Lab5\

Toolchain / IDE: MDK-ARM V5

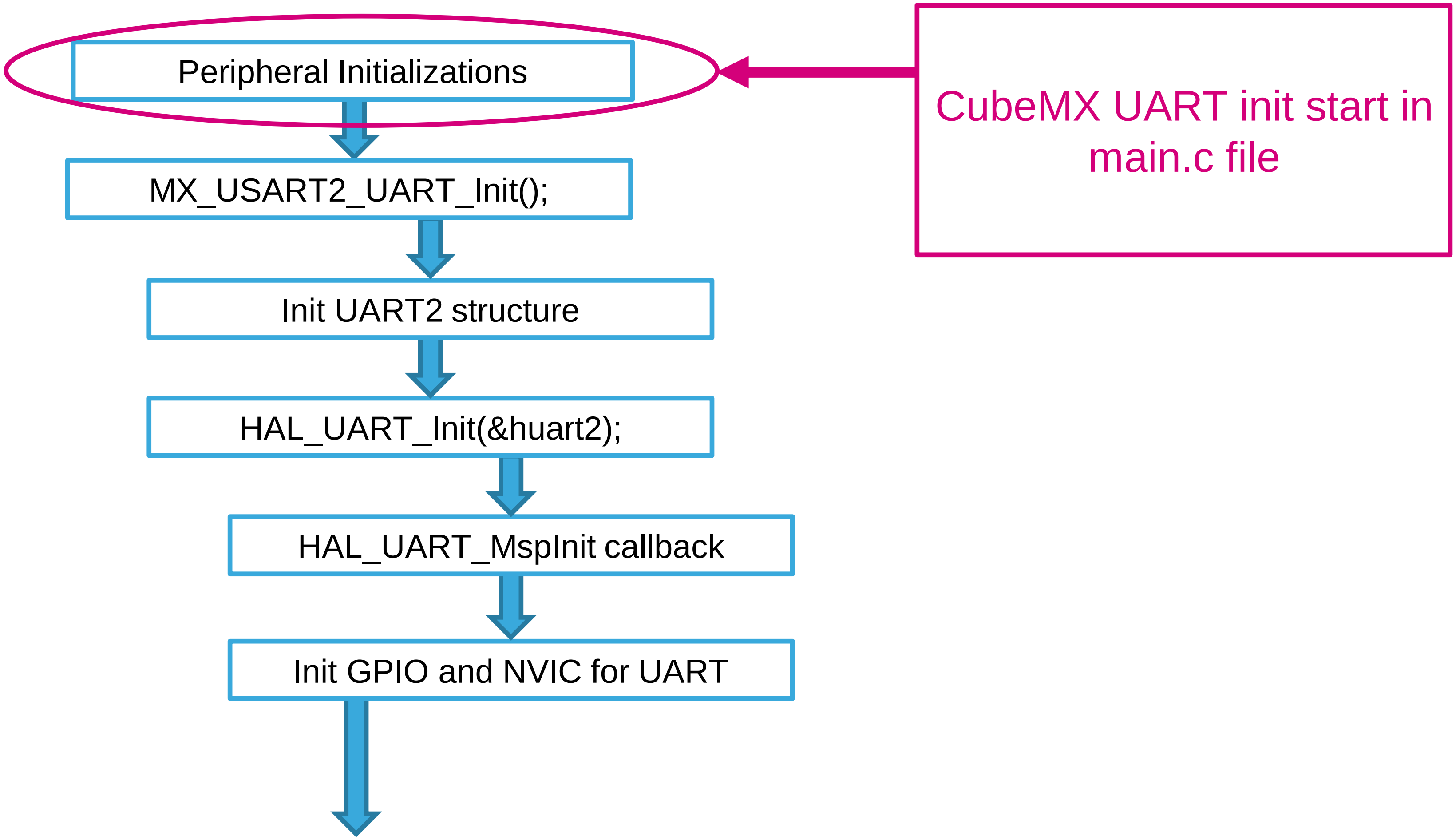
Mcu and Firmware Package

Mcu Reference: STM32F030R8Tx

Firmware Package Name and Version: STM32Cube FW_F0 V1.3.0

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

MX_USART2_UART_Init();

Init UART2 structure

HAL_UART_Init(&huart2);

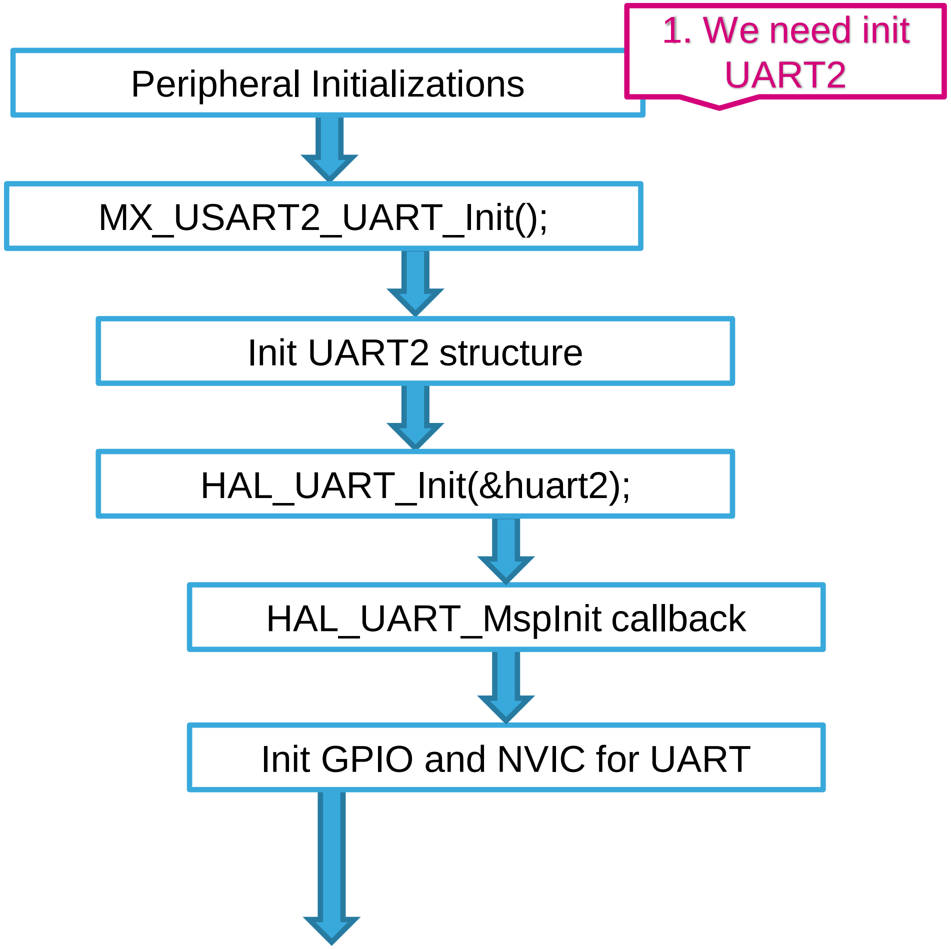
HAL_UART_MspInit callback

Init GPIO and NVIC for UART

CubeMX UART init start in main.c file

2.1.1 Simple UART communication

HAL Library init flow



1. We need init UART2

Peripheral Initializations

MX_USART2_UART_Init();

Init UART2 structure

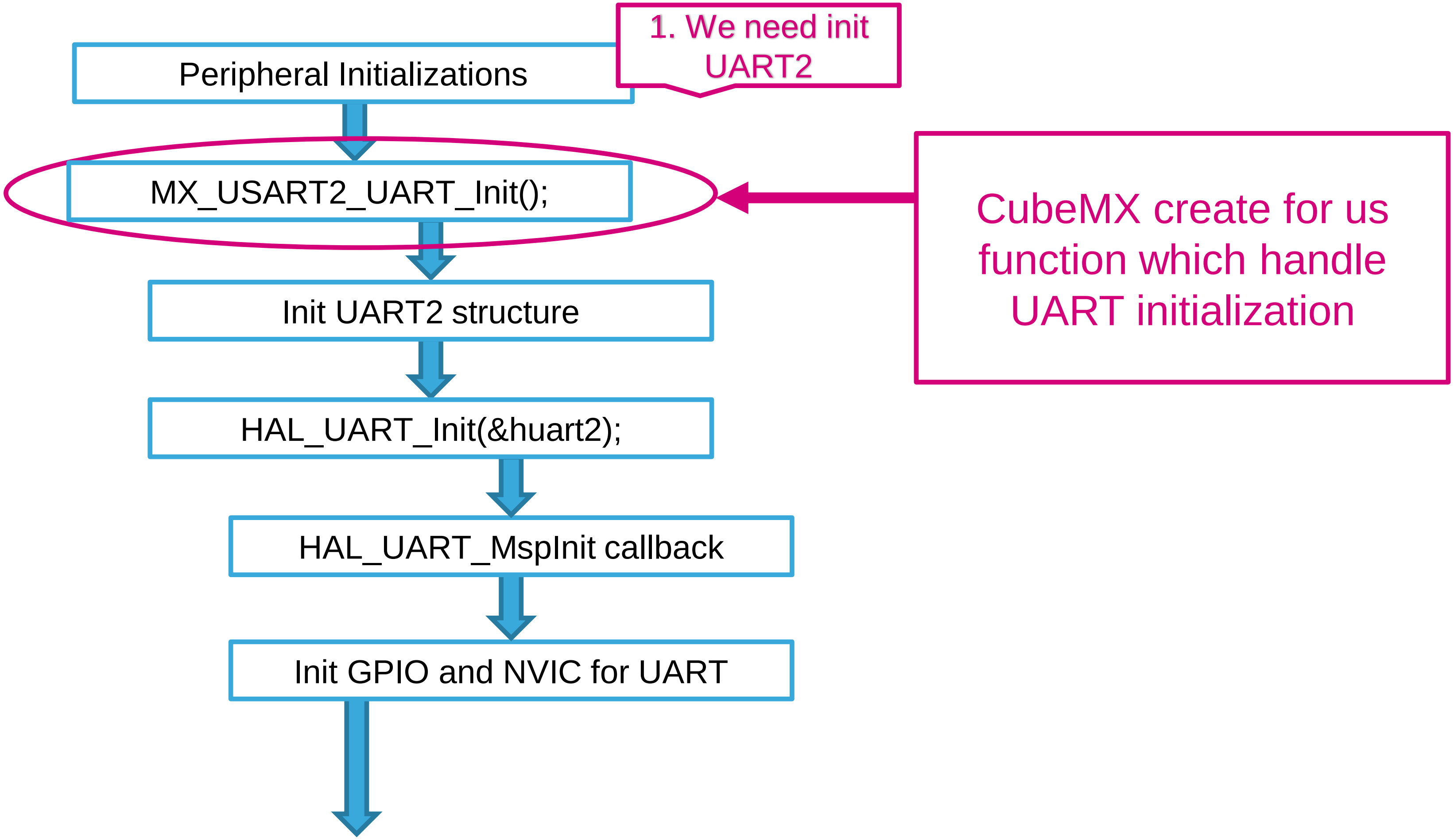
HAL_UART_Init(&huart2);

HAL_UART_MspInit callback

Init GPIO and NVIC for UART

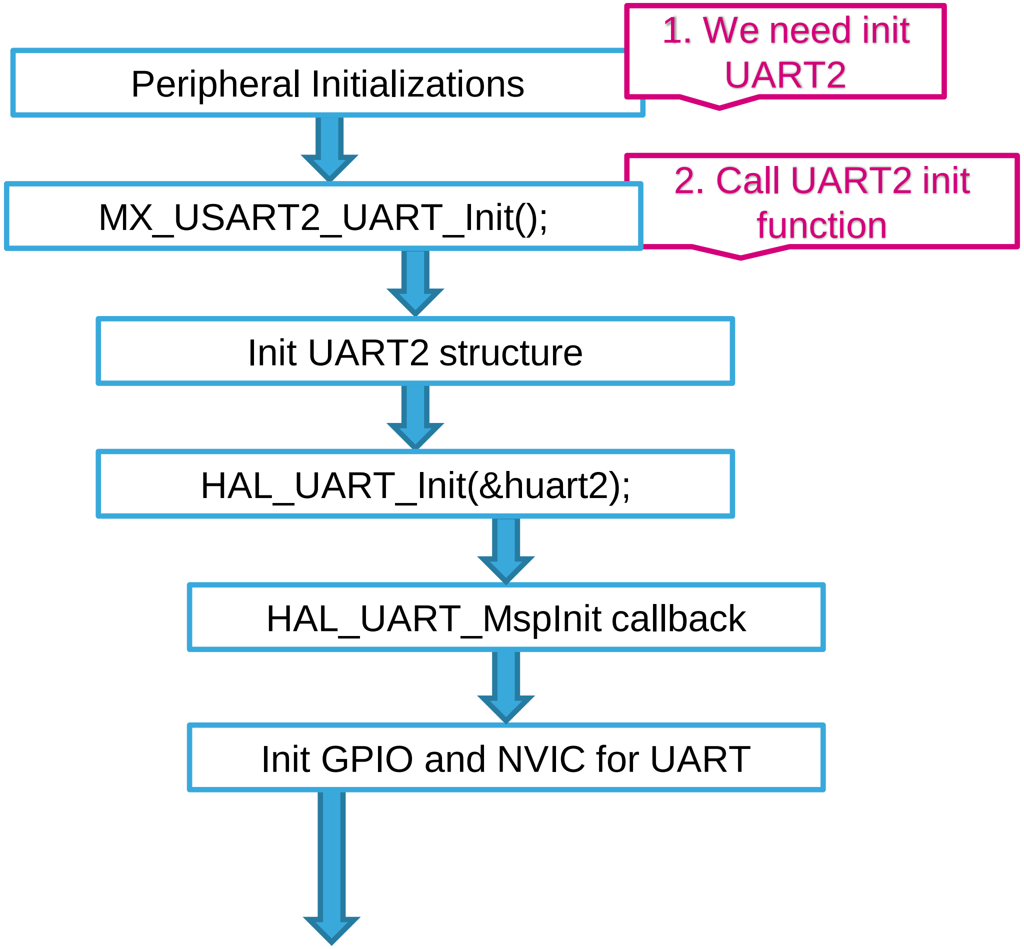
2.1.1 Simple UART communication

HAL Library init flow


Peripheral Initializations
1. We need init UART2
MX_USART2_UART_Init();
CubeMX create for us function which handle UART initialization
Init UART2 structure
HAL_UART_Init(&huart2);
HAL_UART_MspInit callback
Init GPIO and NVIC for UART

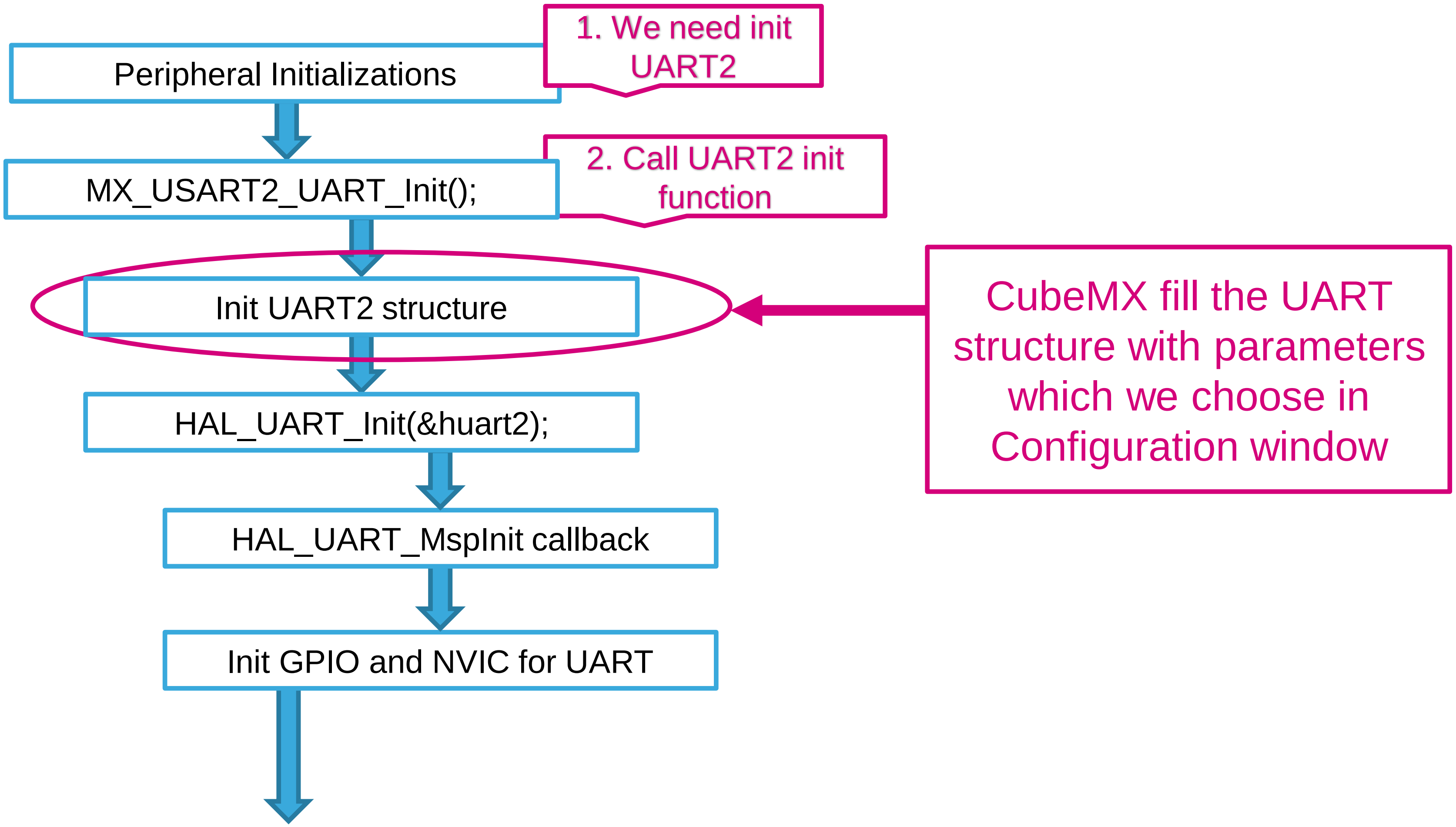
2.1.1 Simple UART communication

HAL Library init flow


Peripheral Initializations
1. We need init UART2
MX_USART2_UART_Init();
2. Call UART2 init function
Init UART2 structure
HAL_UART_Init(&huart2);
HAL_UART_MspInit callback
Init GPIO and NVIC for UART

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

1. We need init UART2

MX_USART2_UART_Init();

2. Call UART2 init function

Init UART2 structure

CubeMX fill the UART structure with parameters which we choose in Configuration window

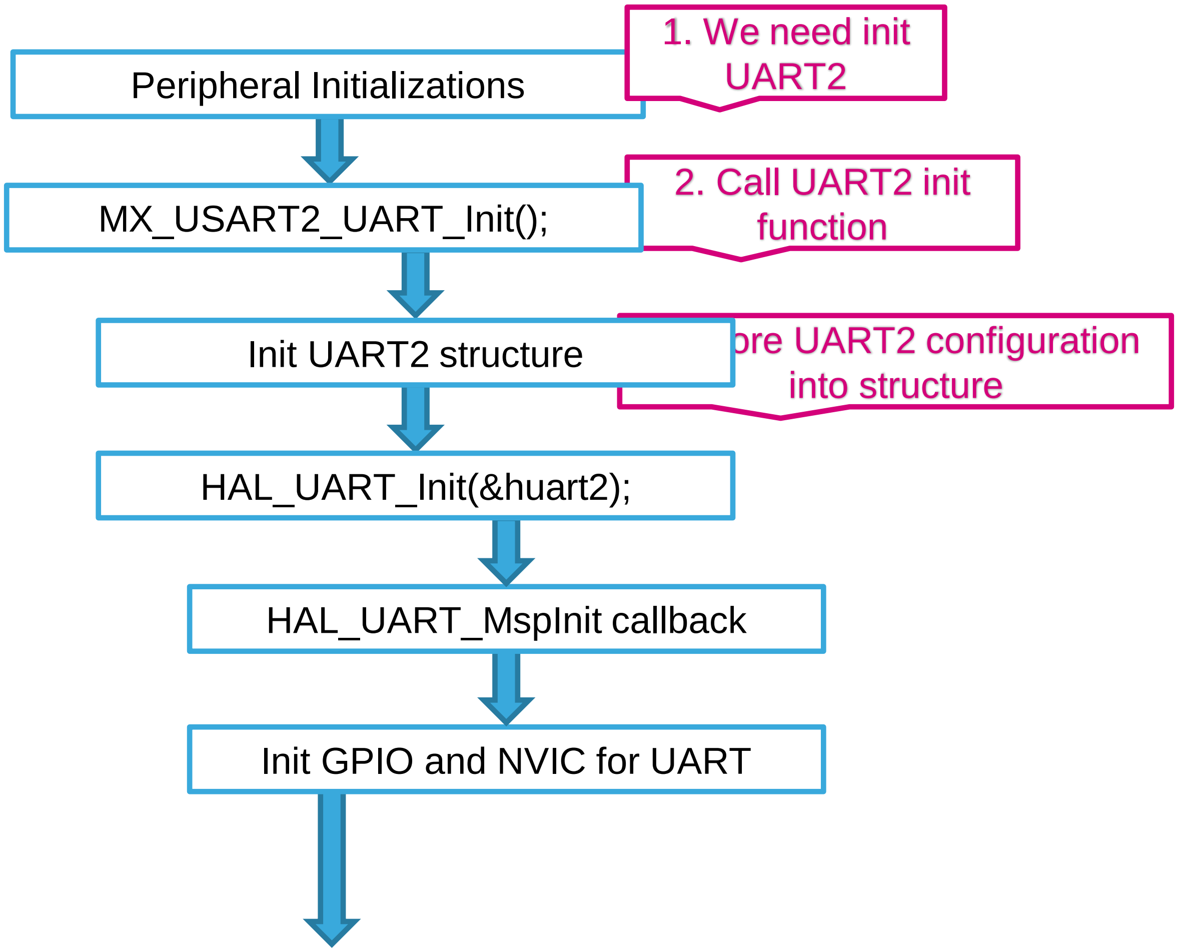
HAL_UART_Init(&huart2);

HAL_UART_MspInit callback

Init GPIO and NVIC for UART

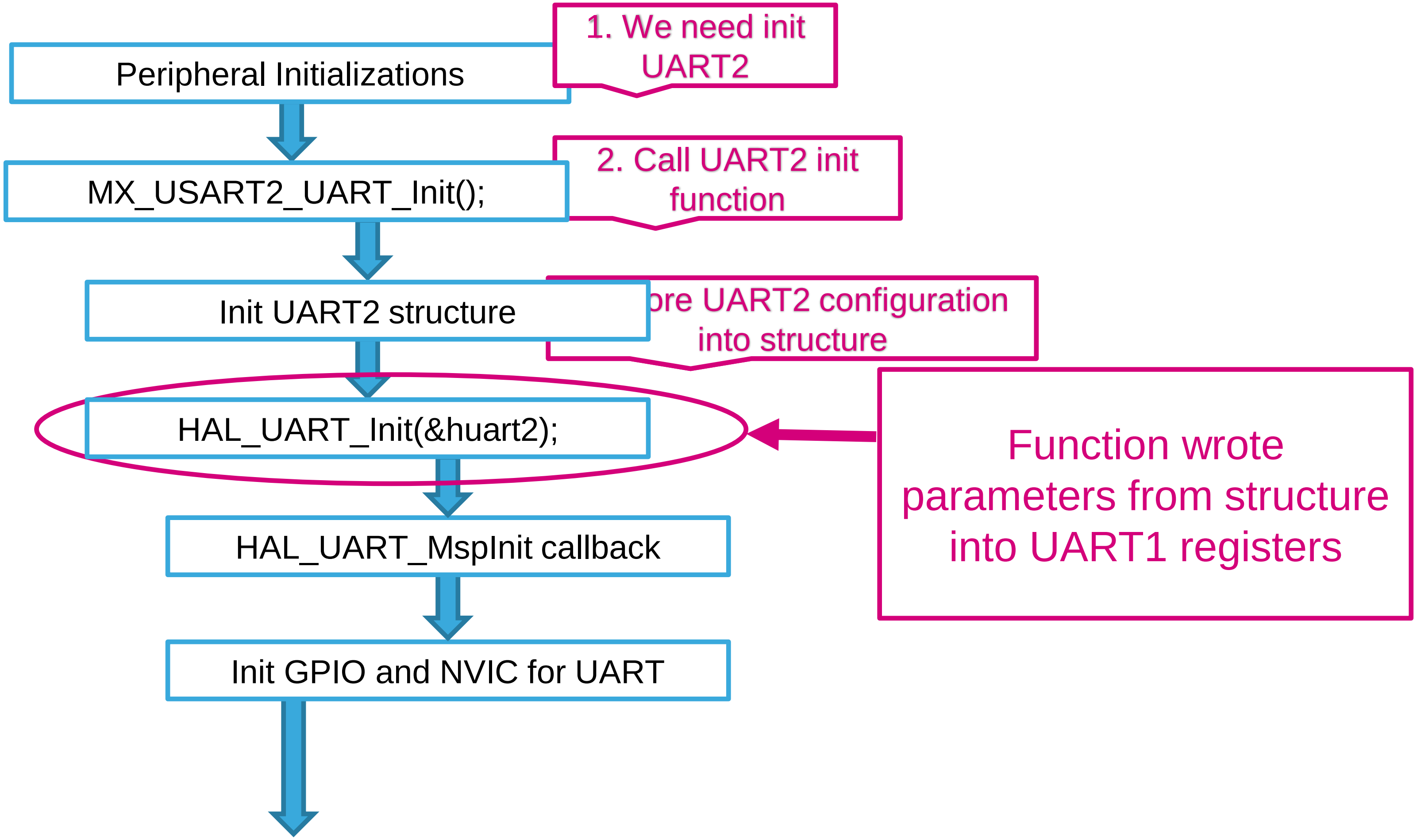
2.1.1 Simple UART communication

HAL Library init flow


Peripheral Initializations
MX_USART2_UART_Init();
Init UART2 structure
HAL_UART_Init(&huart2);
HAL_UART_MspInit callback
Init GPIO and NVIC for UART
1. We need init UART2
2. Call UART2 init function
ore UART2 configuration into structure

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

1. We need init UART2

MX_USART2_UART_Init();

2. Call UART2 init function

Init UART2 structure

ore UART2 configuration into structure

HAL_UART_Init(&huart2);

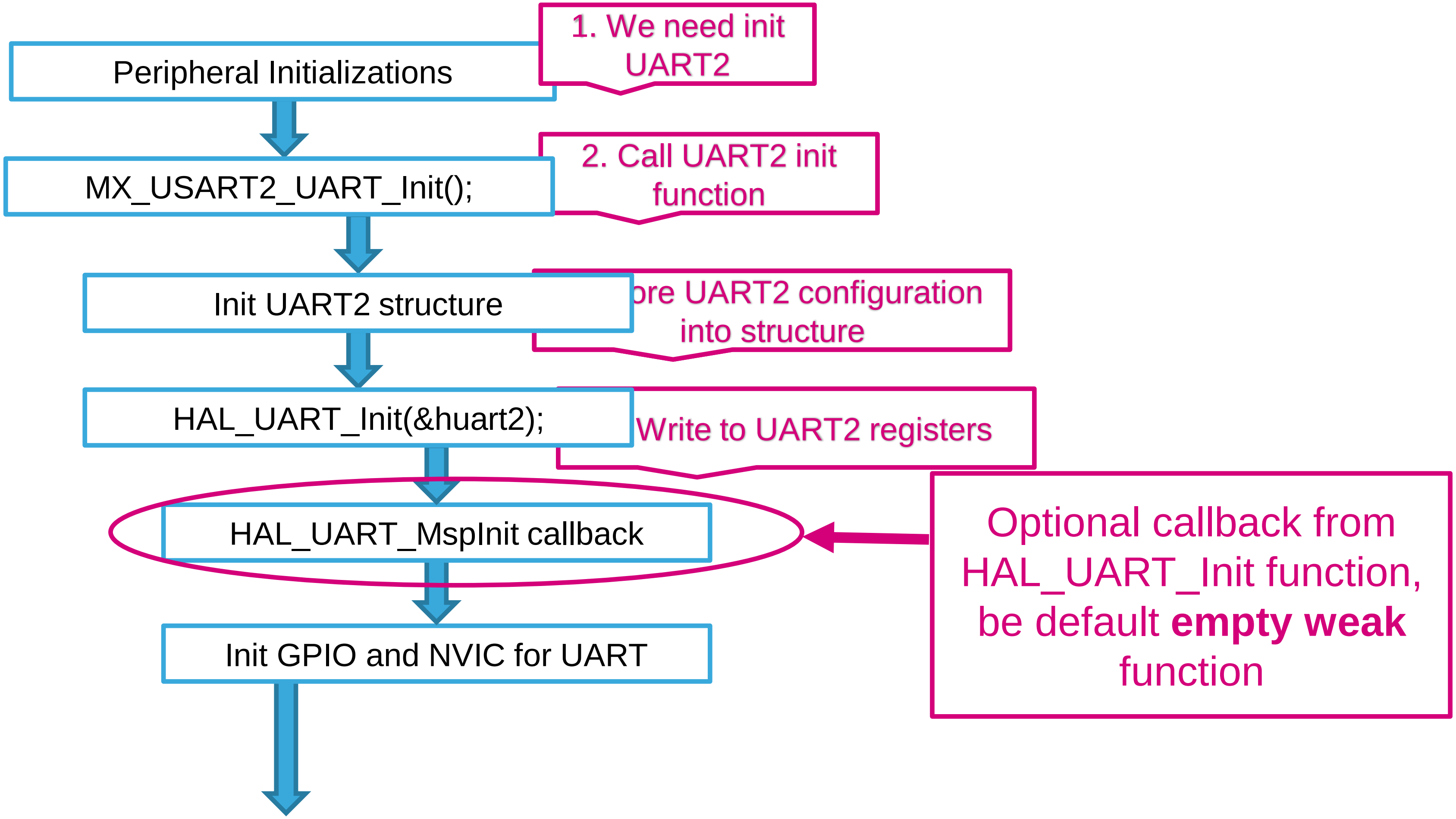
Function wrote parameters from structure into UART1 registers

HAL_UART_MspInit callback

Init GPIO and NVIC for UART

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

1. We need init UART2

MX_USART2_UART_Init();

2. Call UART2 init function

Init UART2 structure

ore UART2 configuration into structure

HAL_UART_Init(&huart2);

Write to UART2 registers

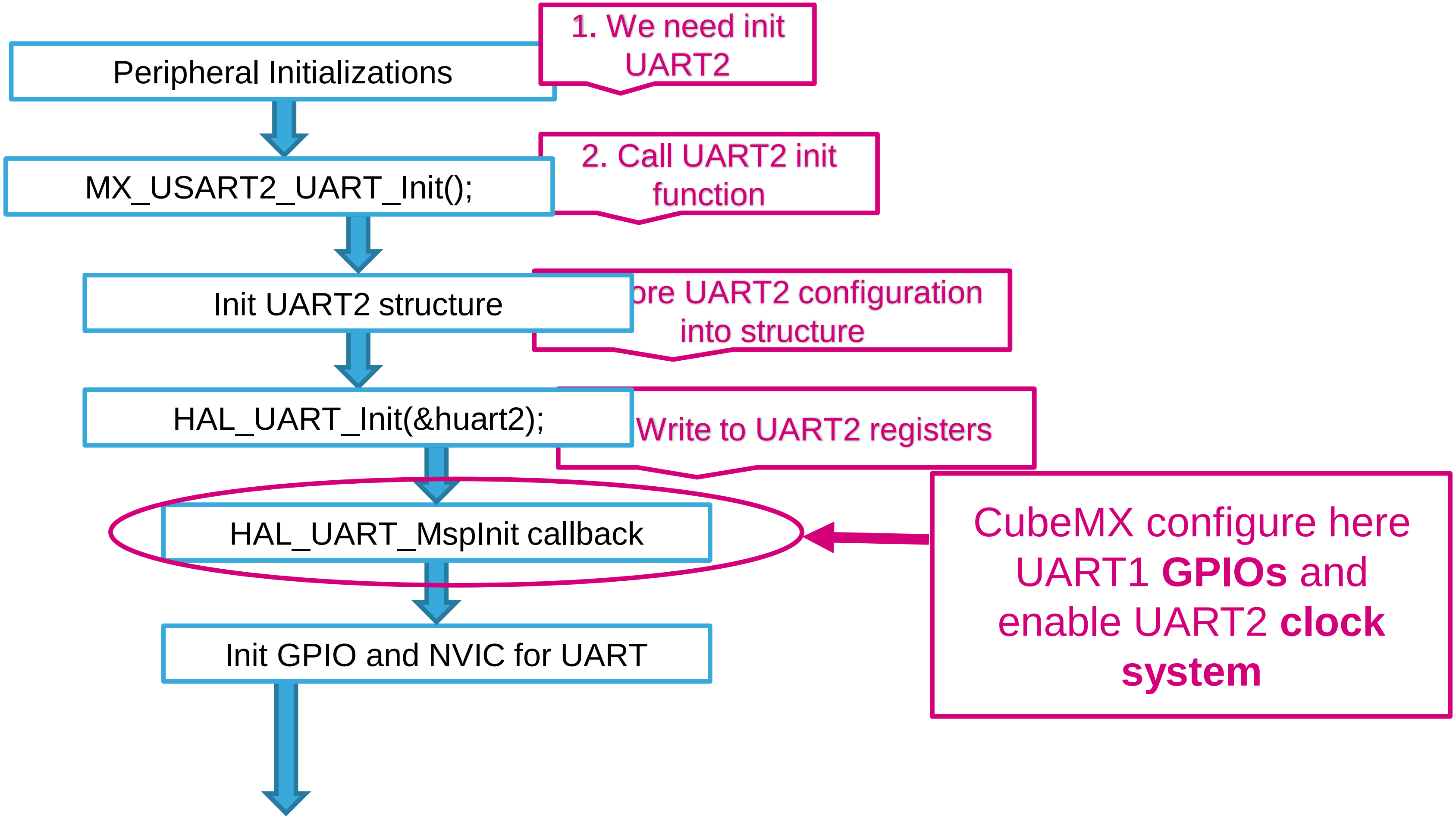
HAL_UART_MspInit callback

Optional callback from HAL_UART_Init function, be default **empty weak** function

Init GPIO and NVIC for UART

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

MX_USART2_UART_Init();

Init UART2 structure

HAL_UART_Init(&huart2);

HAL_UART_MspInit callback

Init GPIO and NVIC for UART

1. We need init UART2

2. Call UART2 init function

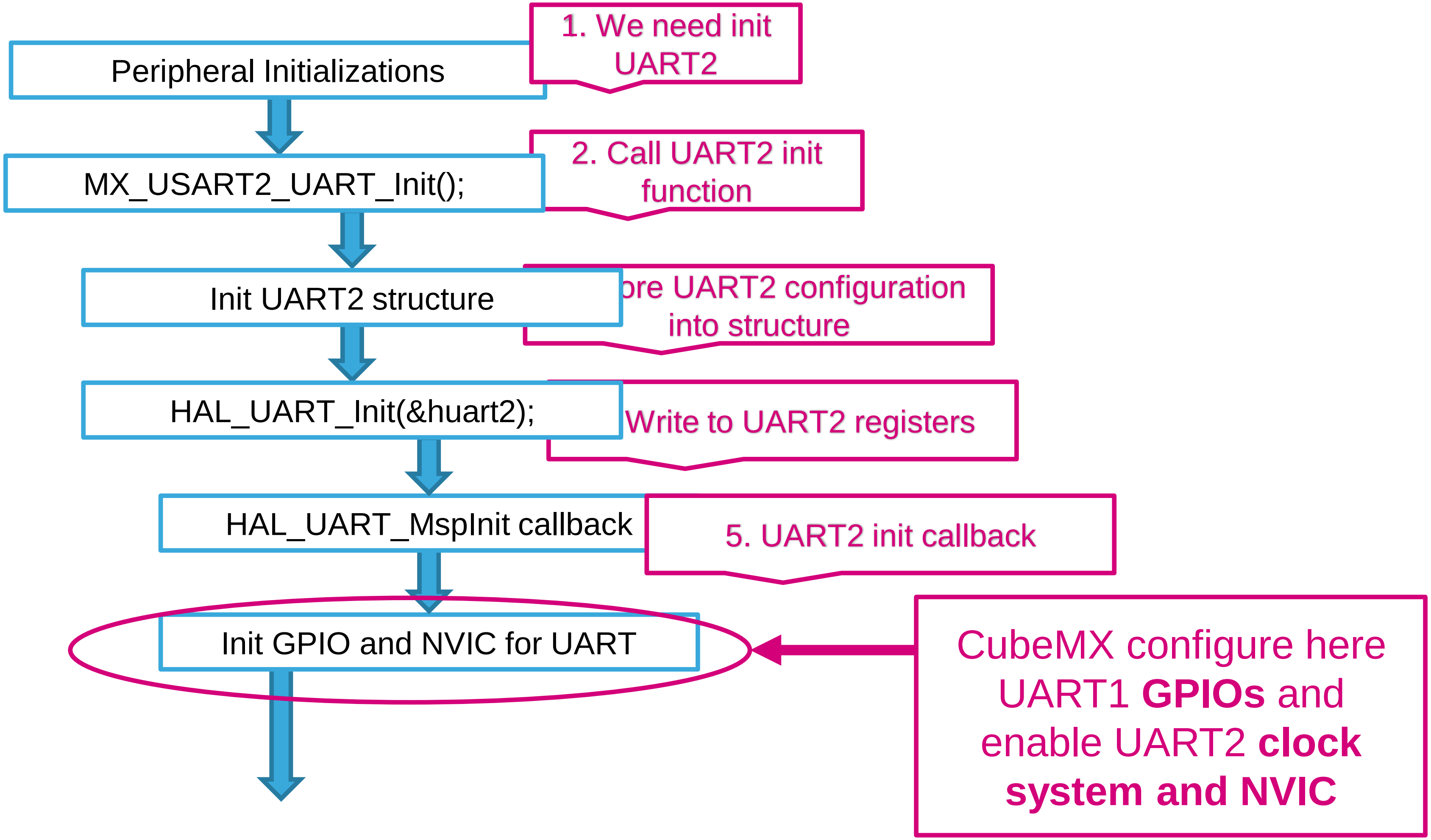
ore UART2 configuration into structure

Write to UART2 registers

CubeMX configure here UART1 **GPIOs** and enable UART2 **clock system**

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

1. We need init UART2

MX_USART2_UART_Init();

2. Call UART2 init function

Init UART2 structure

ore UART2 configuration into structure

HAL_UART_Init(&huart2);

Write to UART2 registers

HAL_UART_MspInit callback

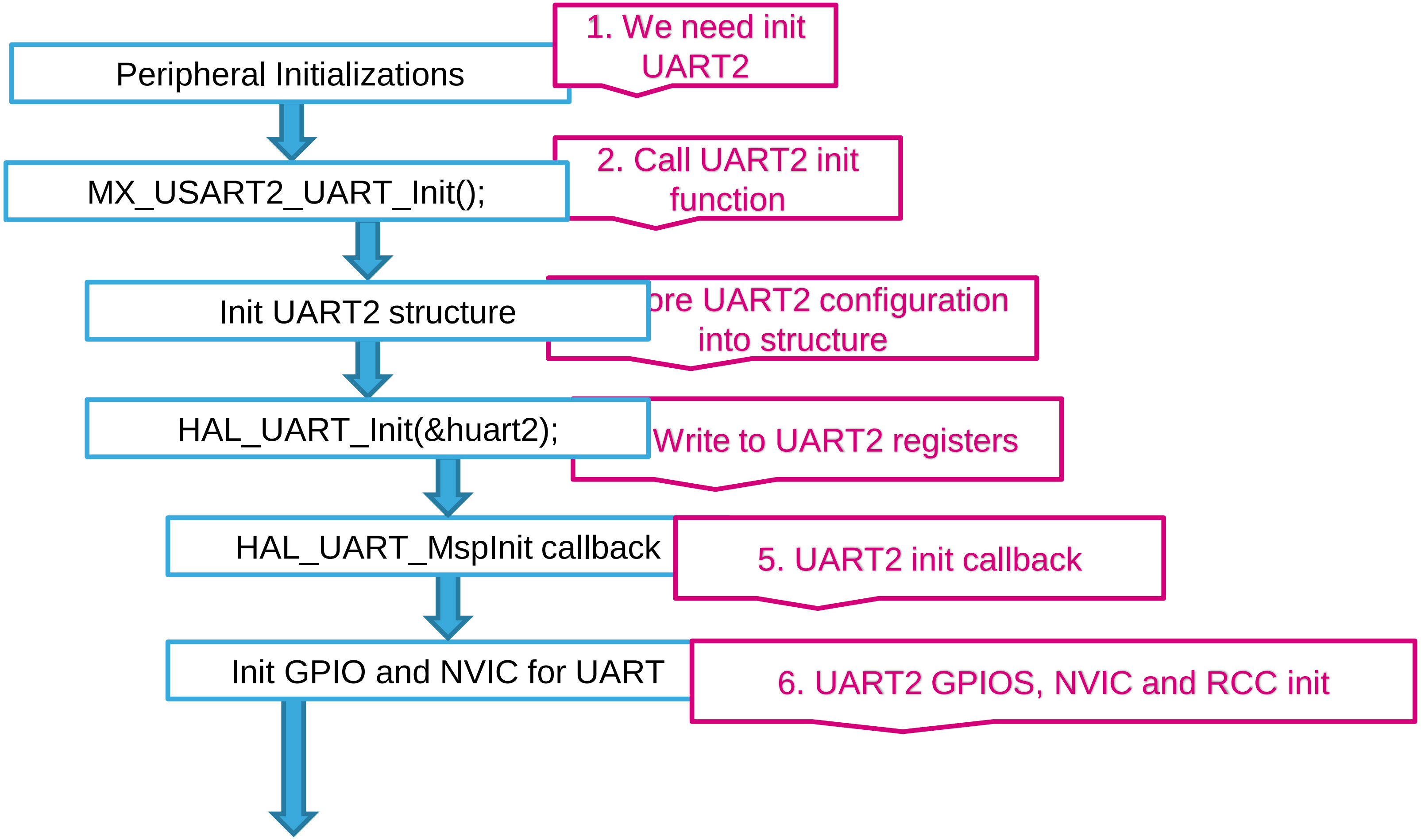
5. UART2 init callback

Init GPIO and NVIC for UART

CubeMX configure here UART1 **GPIOs** and enable UART2 **clock system and NVIC**

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

1. We need init UART2

MX_USART2_UART_Init();

2. Call UART2 init function

Init UART2 structure

ore UART2 configuration into structure

HAL_UART_Init(&huart2);

Write to UART2 registers

HAL_UART_MspInit callback

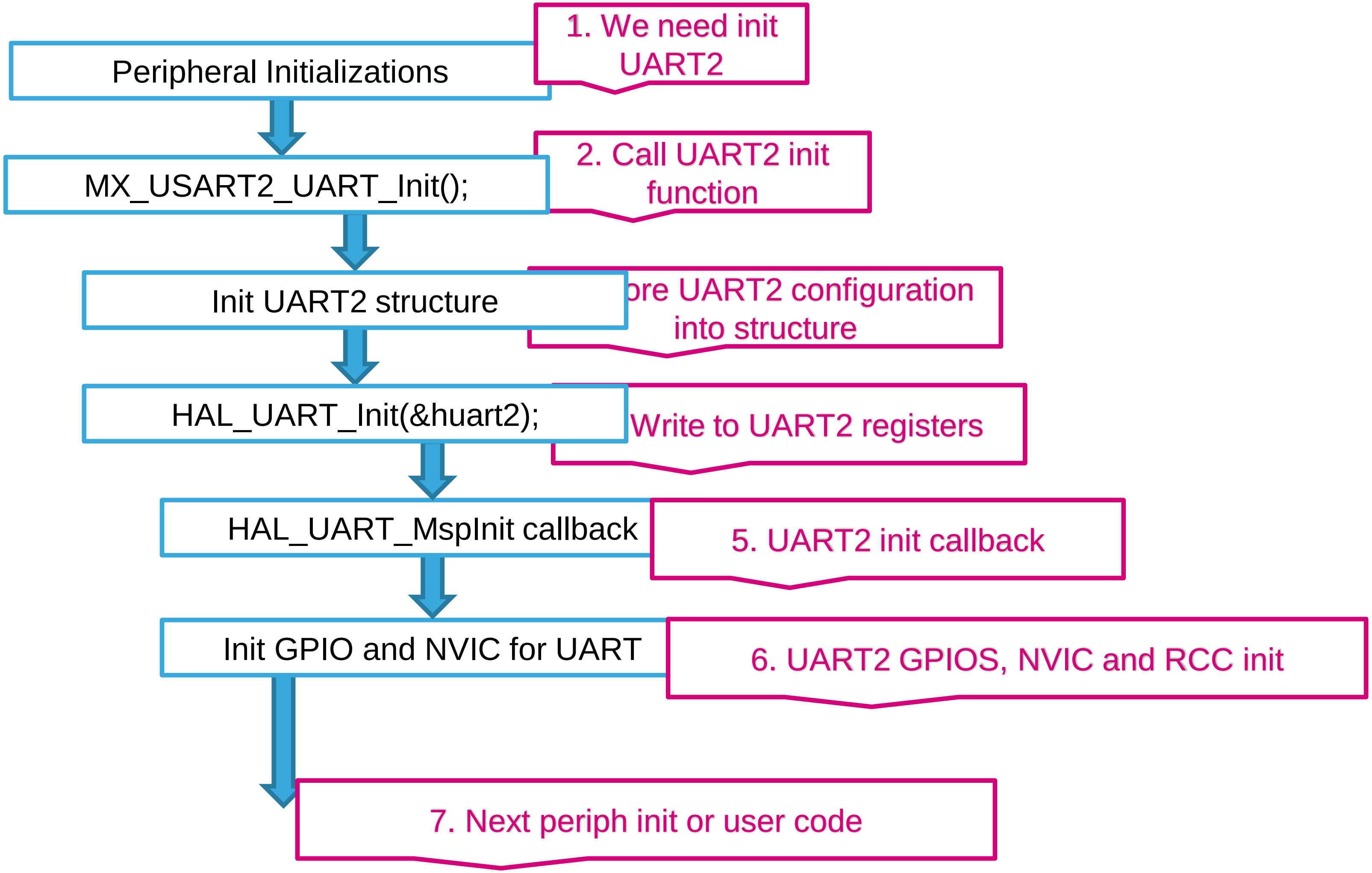
5. UART2 init callback

Init GPIO and NVIC for UART

6. UART2 GPIOS, NVIC and RCC init

2.1.1 Simple UART communication

HAL Library init flow



Peripheral Initializations

1. We need init UART2

MX_USART2_UART_Init();

2. Call UART2 init function

Init UART2 structure

ore UART2 configuration into structure

HAL_UART_Init(&huart2);

Write to UART2 registers

HAL_UART_MspInit callback

5. UART2 init callback

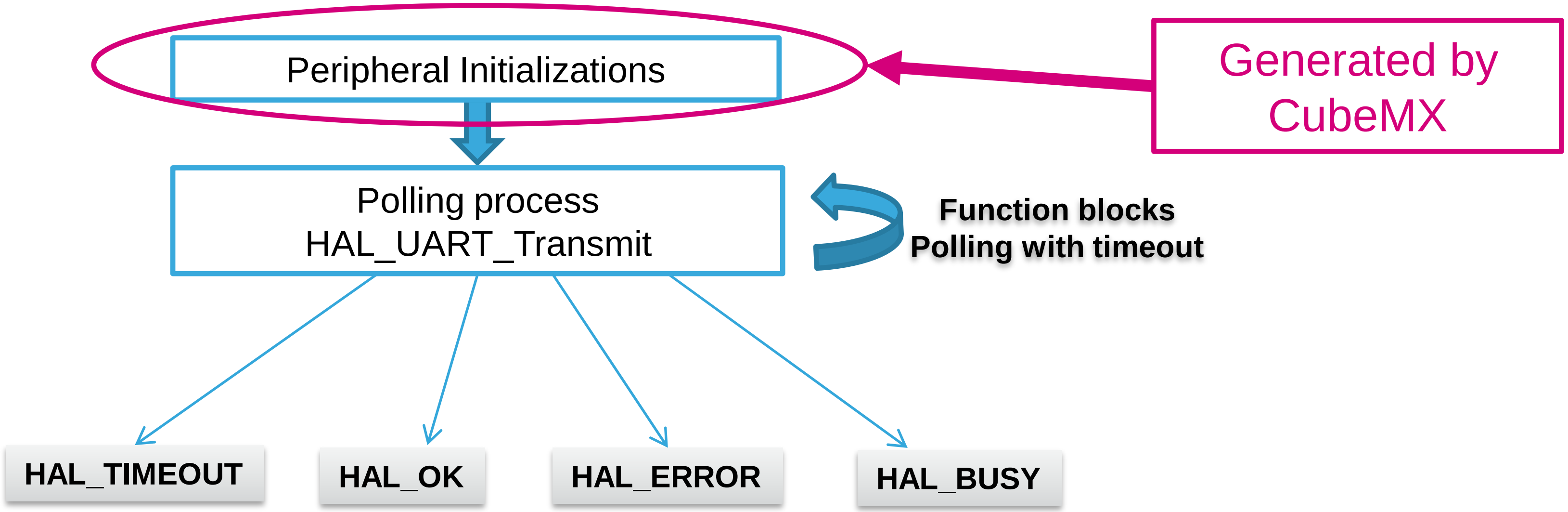
Init GPIO and NVIC for UART

6. UART2 GPIOS, NVIC and RCC init

7. Next periph init or user code

2.1.1 Simple UART communication

HAL Library transmit flow



Peripheral Initializations

Generated by CubeMX

Polling process
HAL_UART_Transmit

Function blocks
Polling with timeout

HAL_TIMEOUT

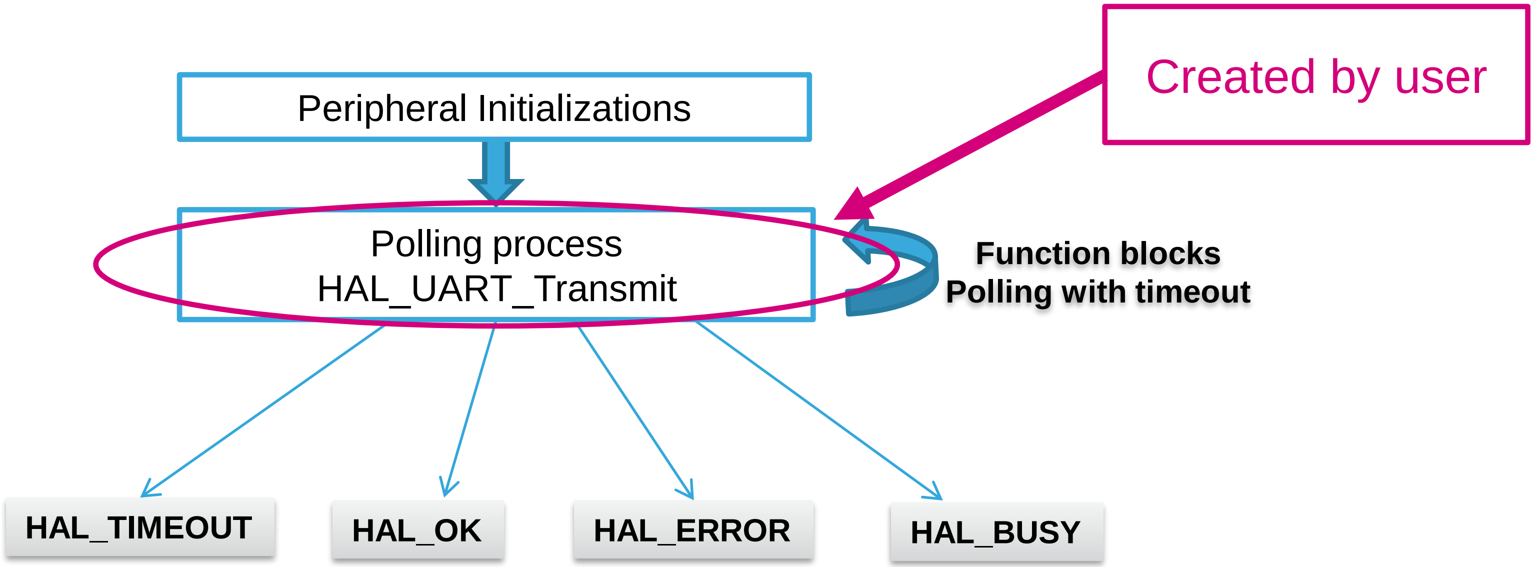
HAL_OK

HAL_ERROR

HAL_BUSY

2.1.1 Simple UART communication

HAL Library transmit flow



Created by user

Peripheral Initializations

Polling process
HAL_UART_Transmit

Function blocks
Polling with timeout

HAL_TIMEOUT

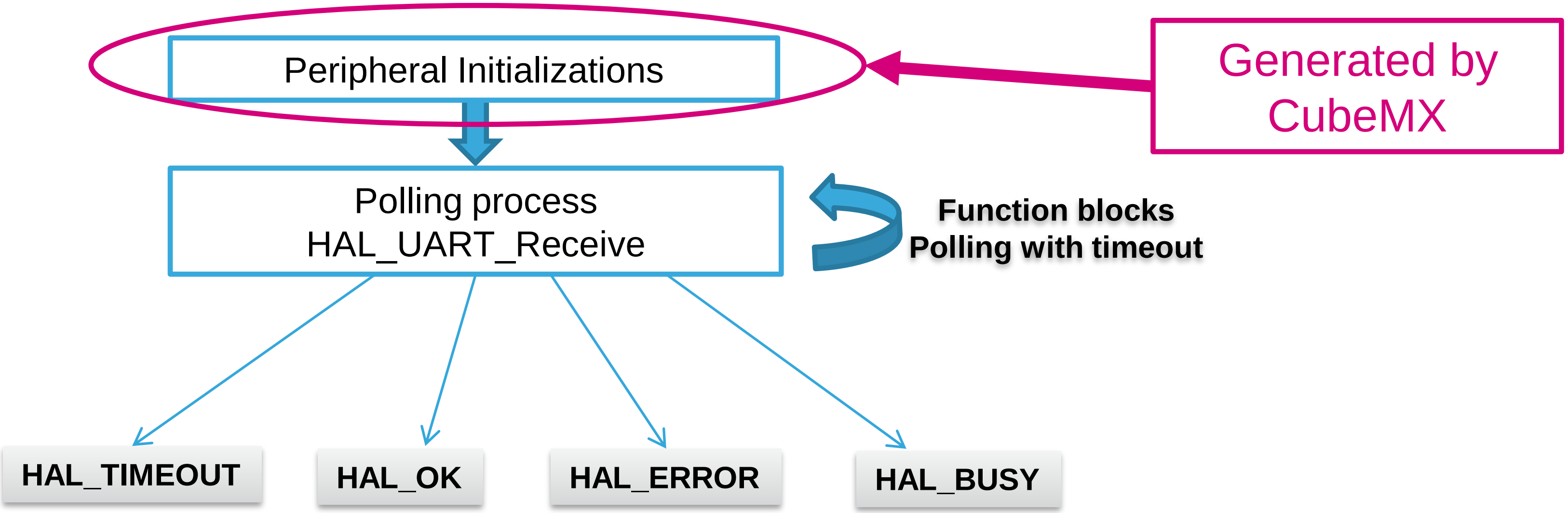
HAL_OK

HAL_ERROR

HAL_BUSY

2.1.1 Simple UART communication

HAL Library receive flow



Peripheral Initializations

Generated by CubeMX

Polling process
HAL_UART_Receive

Function blocks
Polling with timeout

HAL_TIMEOUT

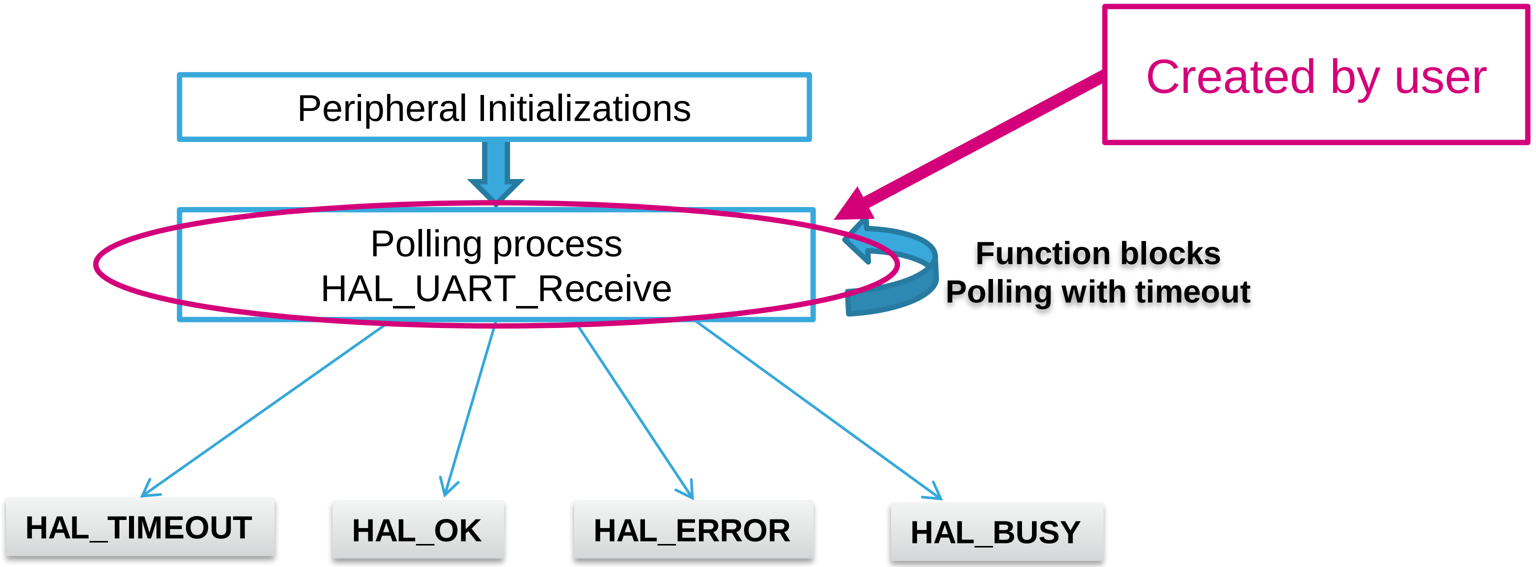
HAL_OK

HAL_ERROR

HAL_BUSY

2.1.1 Simple UART communication

HAL Library receive flow



Peripheral Initializations

Polling process
HAL_UART_Receive

Created by user

Function blocks
Polling with timeout

HAL_TIMEOUT

HAL_OK

HAL_ERROR

HAL_BUSY

2.1.1 Simple UART communication

- Open the project in our IDE
  - The functions we want to put into main.c
  - Between */* USER CODE BEGIN 3 */* and */* USER CODE END 3 */* tags
  - Into infinite while function
- For transmit use function
  - HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
- For receive use function
  - HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout);

2.1.1 Simple UART communication

- Transmit solution
  - Create data structure for data

```
/* USER CODE BEGIN 0 */
uint8_t
data[]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
/* USER CODE END 0 */
```

  - Call transmit function from while loop

```
/* USER CODE BEGIN 3 */
 /* Infinite loop */
 while (1)
 {
   HAL_UART_Transmit(&huart2,data,10,1000);
 }
 /* USER CODE END 3 */
```

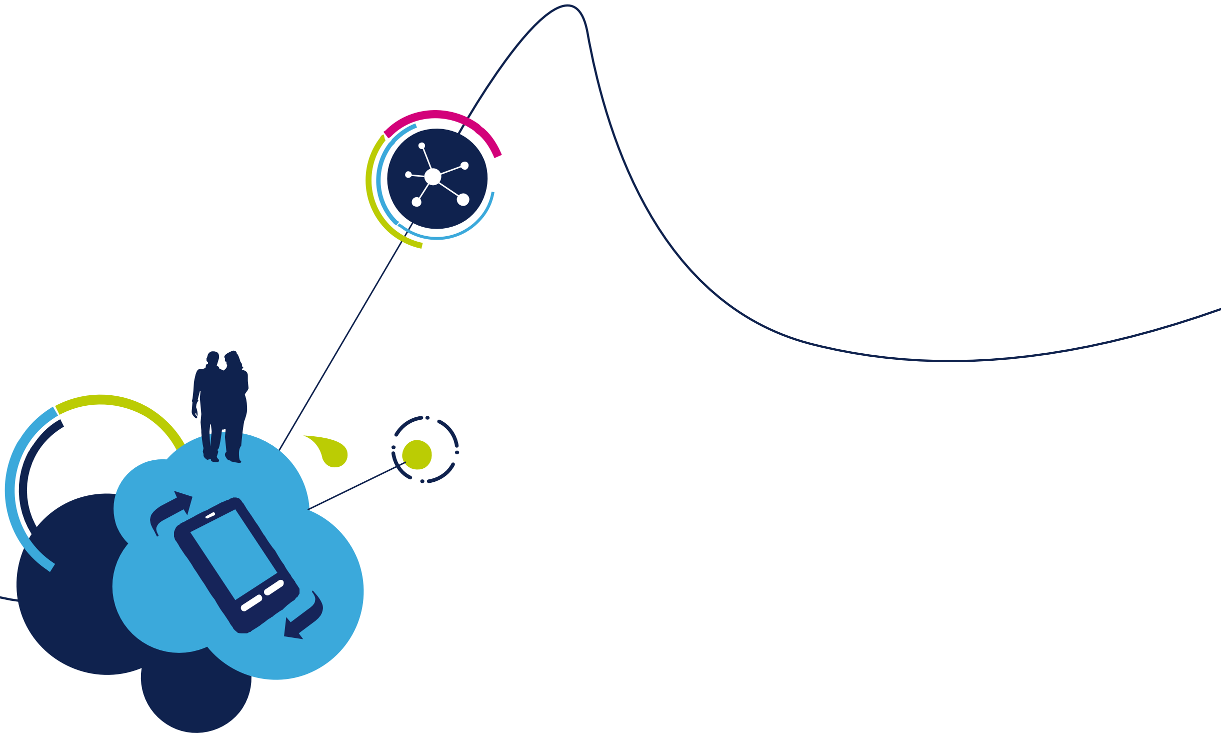
2.1.1 Simple UART communication

- Receive solution
  - Create data structure for data

```
/* USER CODE BEGIN 0 */
uint8_t data[10];
/* USER CODE END 0 */
```

  - Call transmit function from while loop

```
/* USER CODE BEGIN 3 */
 /* Infinite loop */
 while (1)
 {
   HAL_UART_Receive(&huart2,data,10,1000);
 }
 /* USER CODE END 3 */
```



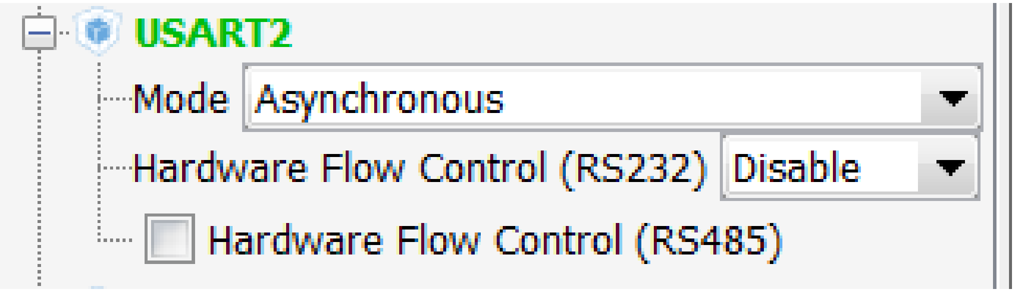
2.1.2 UART Interrupt lab

2.1.2 Use UART with interrupt

- Objective
  - Learn how to setup UART with interrupts in CubeMX
  - How to Generate Code in CubeMX and use HAL functions
  - Create simple loopback example with interrupts
- Goal
  - Configure UART in CubeMX and Generate Code
  - Learn how to send and receive data over UART with interrupts
  - Verify the correct functionality

2.1.2 Use UART with interrupt

- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F0 > STM32F030 > LQFP64 > STM32F030R8
- CubeMX UART selection
  - Select USART2 in asynchronous mode
  - Select PA2 and PA3 for USART2 if weren't selected

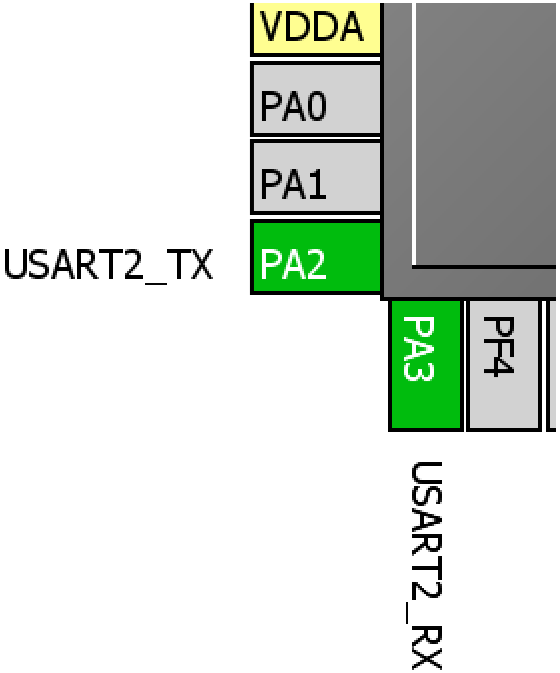


USART2

Mode: Asynchronous

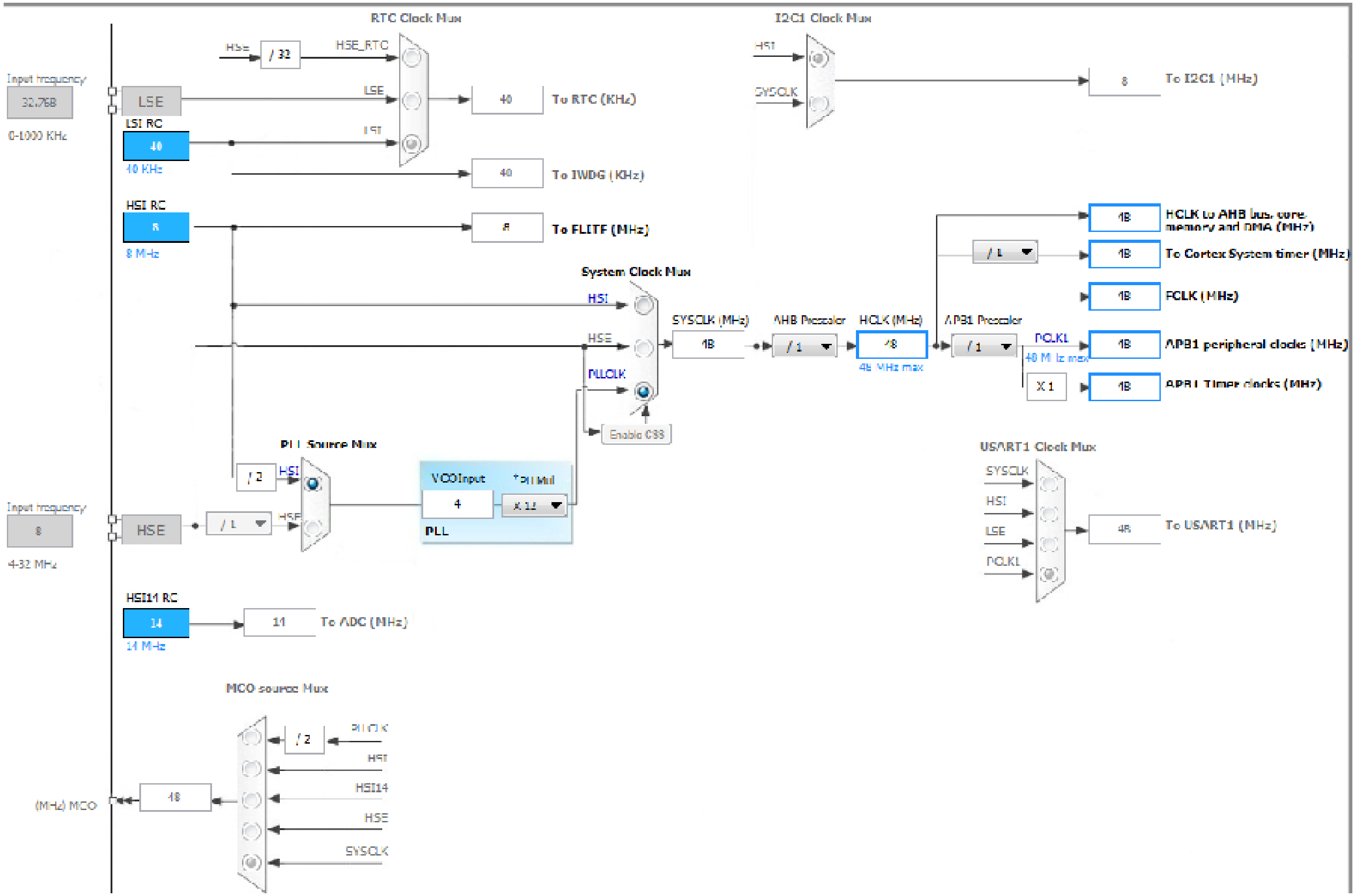
Hardware Flow Control (RS232): Disable

□ Hardware Flow Control (RS485)



2.1.2 Use UART with interrupt

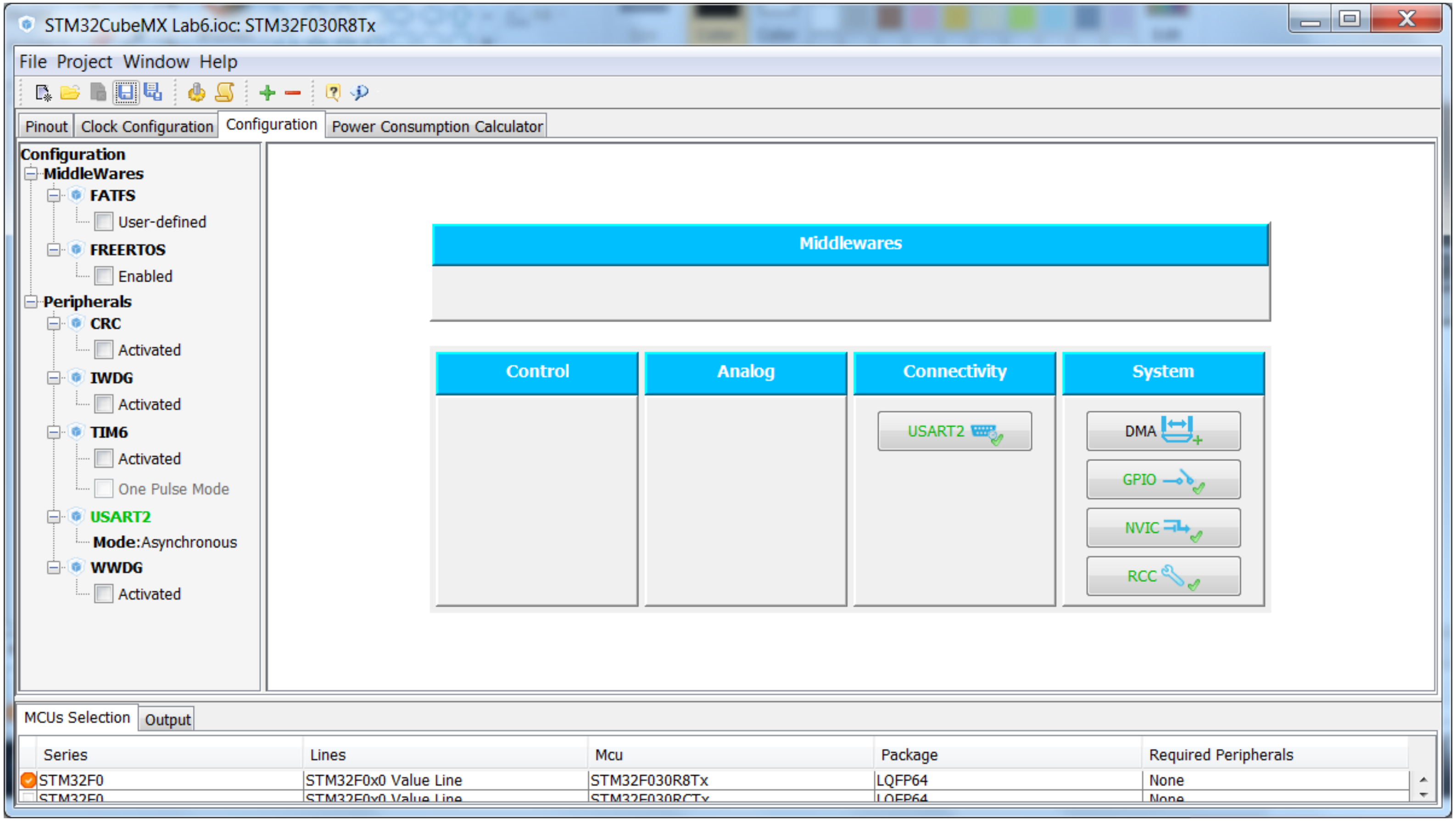
- In order to run on maximum frequency, setup clock system
- Details in lab 0



life.augmented

2.1.2 Use UART with interrupt

- CubeMX UART configuration
  - Tab>Configuration>Connectivity>USART2

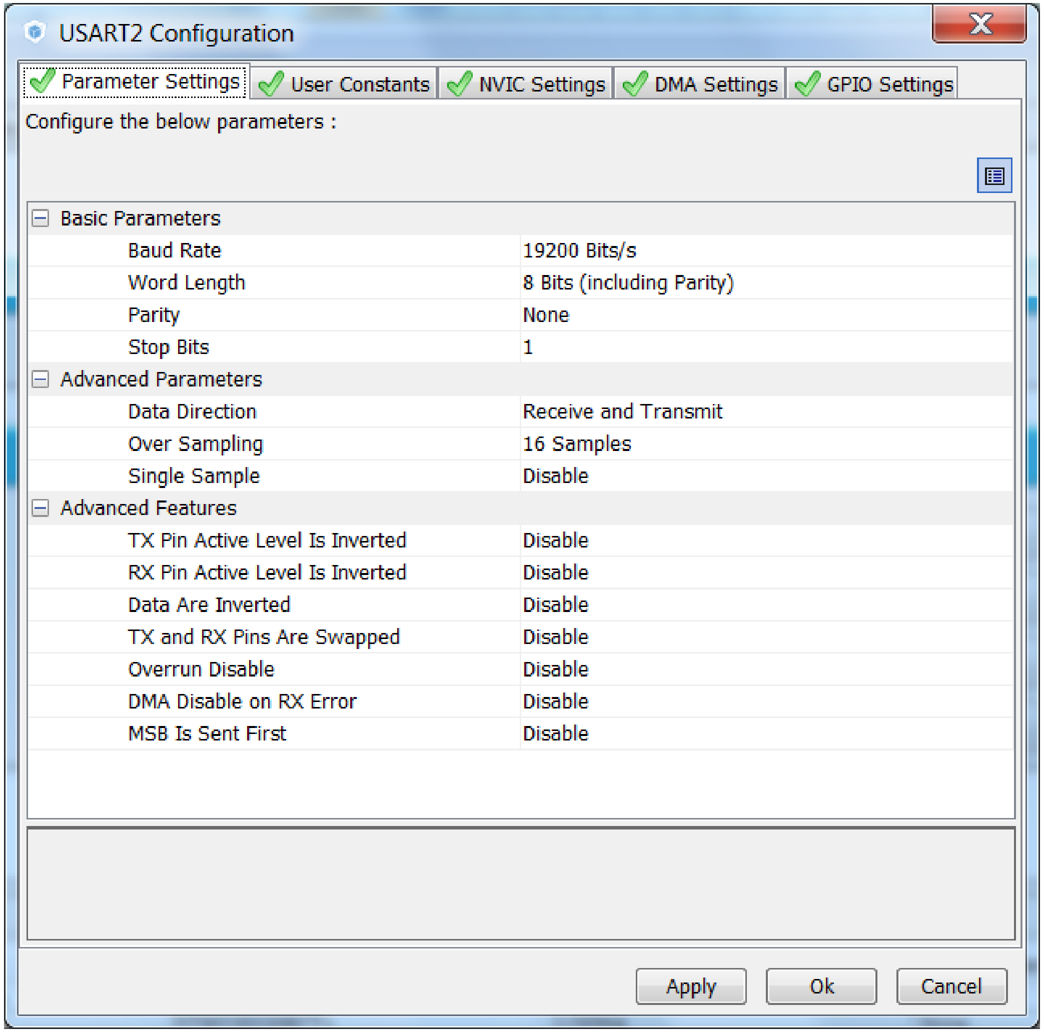


Configuration
- MiddleWares
  - FATFS
    - ☐ User-defined
  - FREERTOS
    - ☐ Enabled
- Peripherals
  - CRC
    - ☐ Activated
  - IWDG
    - ☐ Activated
  - TIM6
    - ☐ Activated
    - ☐ One Pulse Mode
  - USART2
    - **Mode:**Asynchronous
  - WWDG
    - ☐ Activated

Series	Lines	Mcu	Package	Required Peripherals
STM32F0	STM32F0x0 Value Line	STM32F030R8Tx	LQFP64	None
STM32F0	STM32F0x0 Value Line	STM32F030RCTx	LQFP64	None

Use UART with interrupt

- CubeMX UART configuration check:
  - BaudRate
  - World length
  - Parity
  - Stop bits
  - Data direction
  - Oversampling

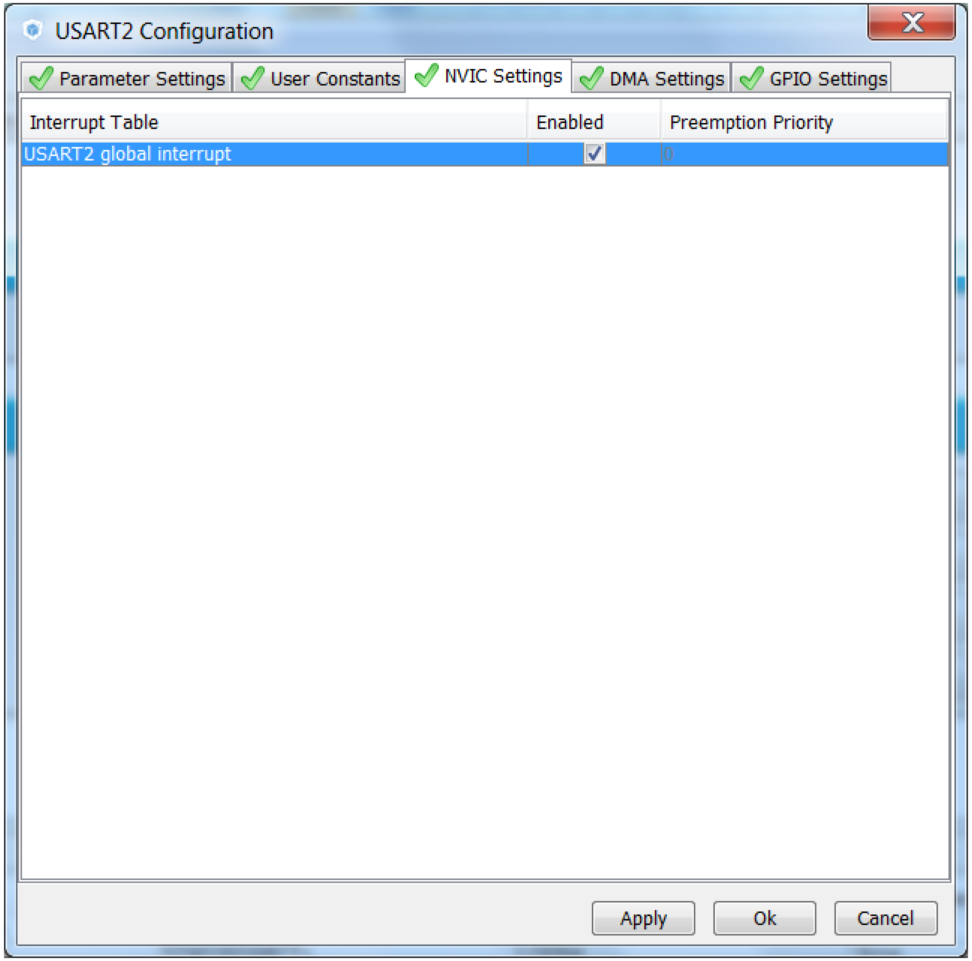


Configure the below parameters :

Basic Parameters	
Baud Rate	19200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable
Advanced Features	
TX Pin Active Level Is Inverted	Disable
RX Pin Active Level Is Inverted	Disable
Data Are Inverted	Disable
TX and RX Pins Are Swapped	Disable
Overrun Disable	Disable
DMA Disable on RX Error	Disable
MSB Is Sent First	Disable

2.1.2 Use UART with interrupt

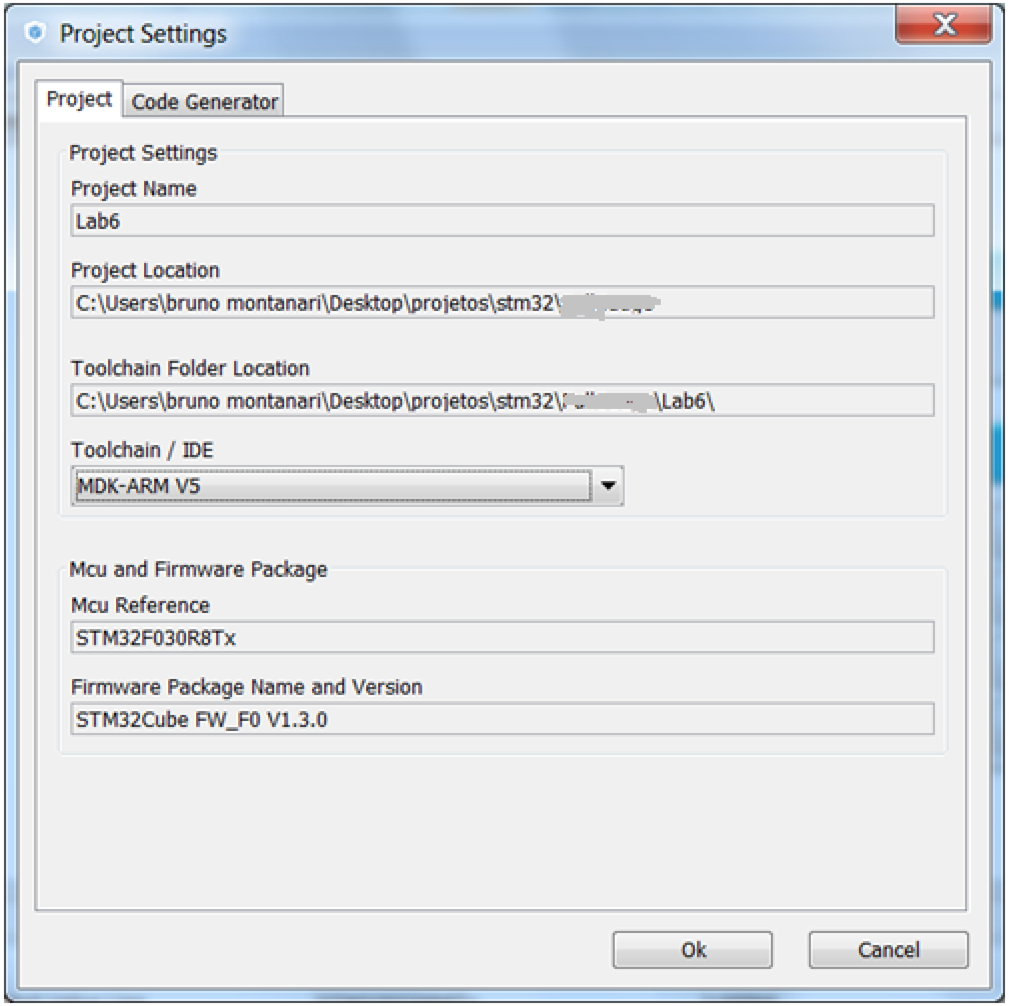
- CubeMX USART configuration NVIC settings
  - TAB>NVIC Settings
  - Enable interrupts
  - OK



Interrupt Table	Enabled	Preemption Priority
USART2 global interrupt	☑	0

2.1.2 Use UART with interrupt

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Now we can Generate Code
  - Menu > Project > Generate Code



Project Settings

Project Settings

Project Name: Lab6

Project Location: C:\Users\bruno montanari\Desktop\projetos\stm32\[redacted]

Toolchain Folder Location: C:\Users\bruno montanari\Desktop\projetos\stm32\[redacted]\Lab6\

Toolchain / IDE: MDK-ARM V5

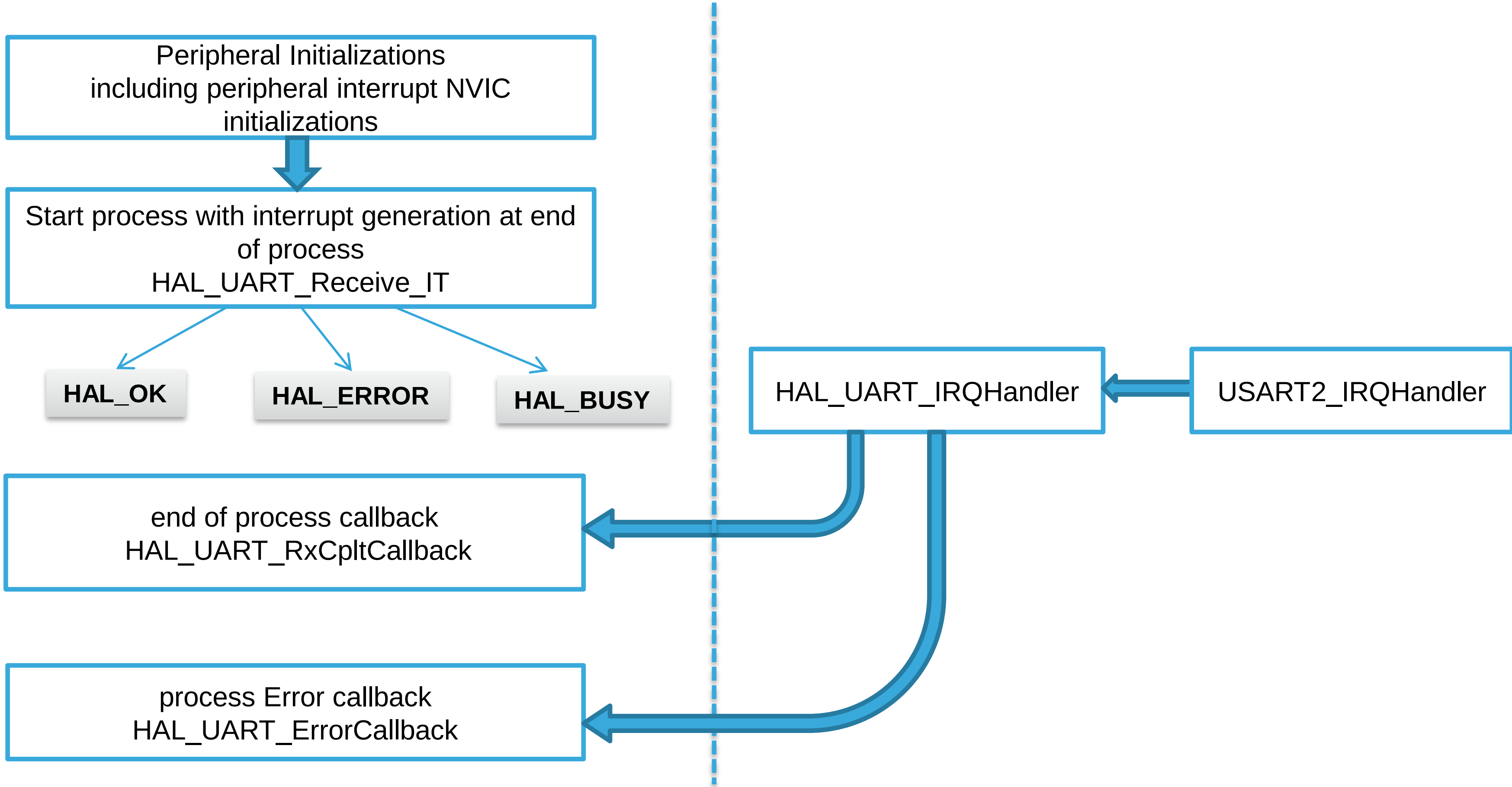
Mcu and Firmware Package

Mcu Reference: STM32F030R8Tx

Firmware Package Name and Version: STM32Cube FW_F0 V1.3.0

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Receive_IT

HAL_UART_IRQHandler

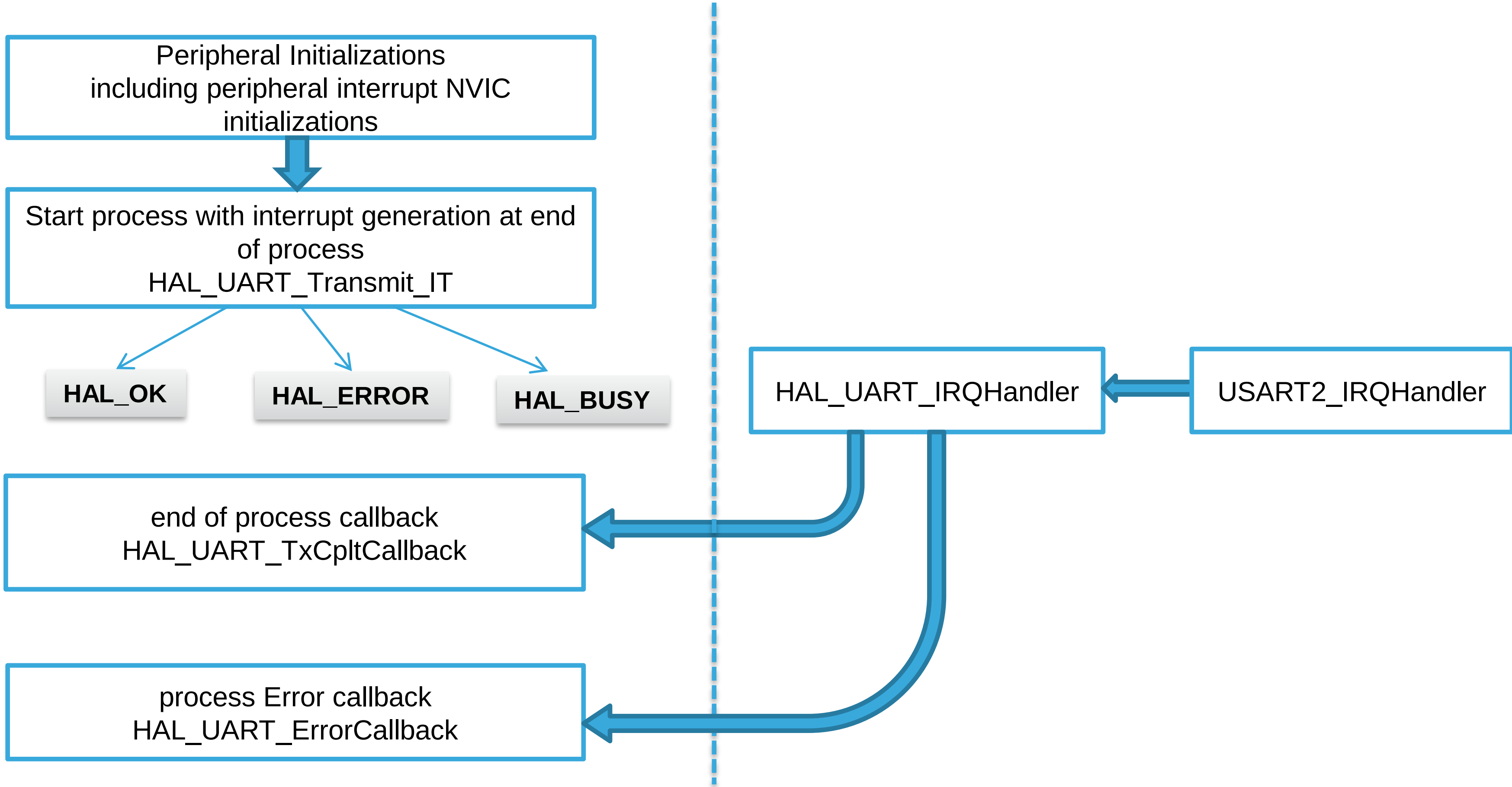
USART2_IRQHandler

end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

2.1.2 Use UART with interrupt

HAL Library UART with IT transmit flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

HAL_UART_IRQHandler

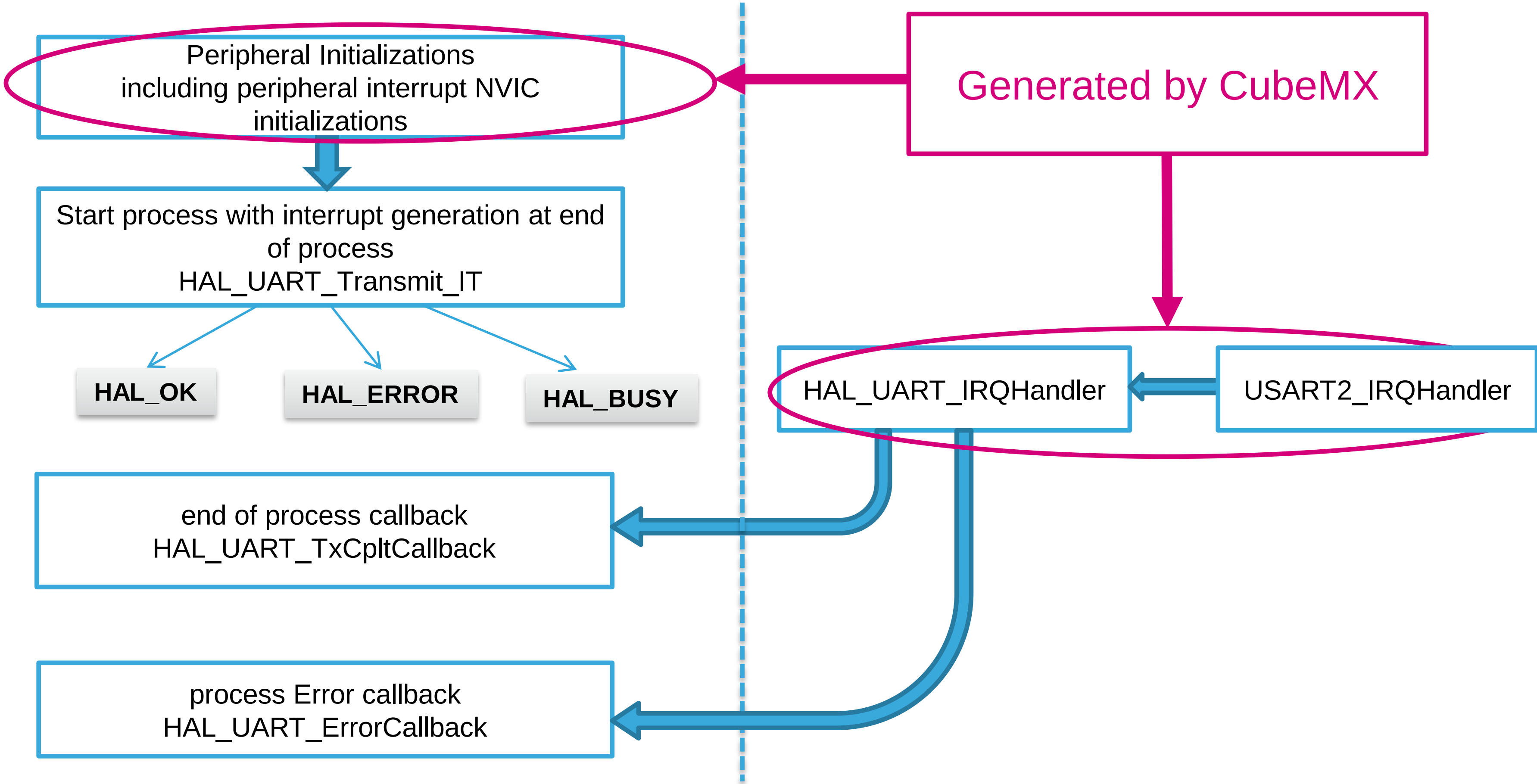
USART2_IRQHandler

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

2.1.2 Use UART with interrupt

HAL Library UART with IT transmit flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

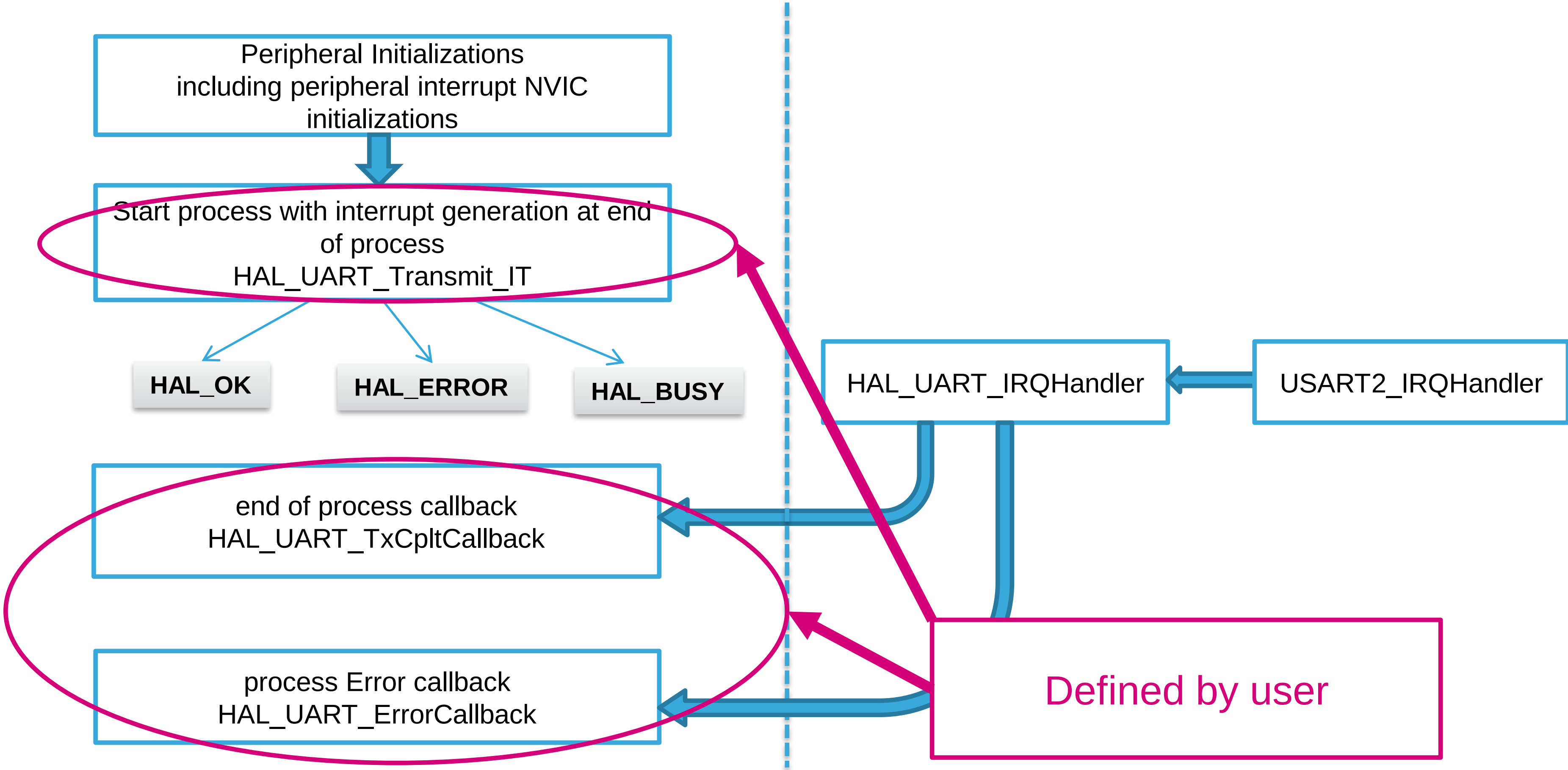
Generated by CubeMX

HAL_UART_IRQHandler

USART2_IRQHandler

Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

HAL_UART_IRQHandler

USART2_IRQHandler

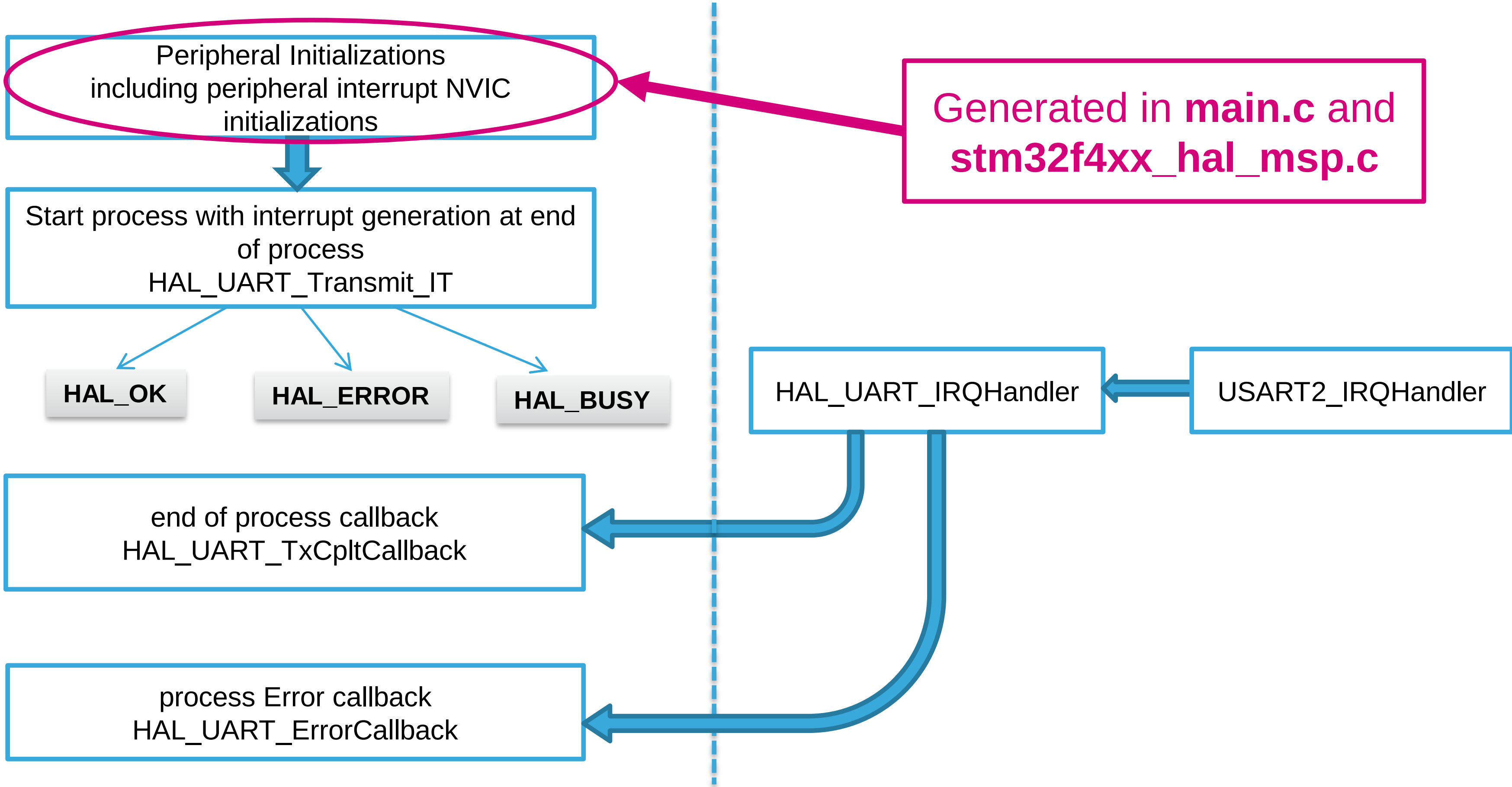
end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

Defined by user

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

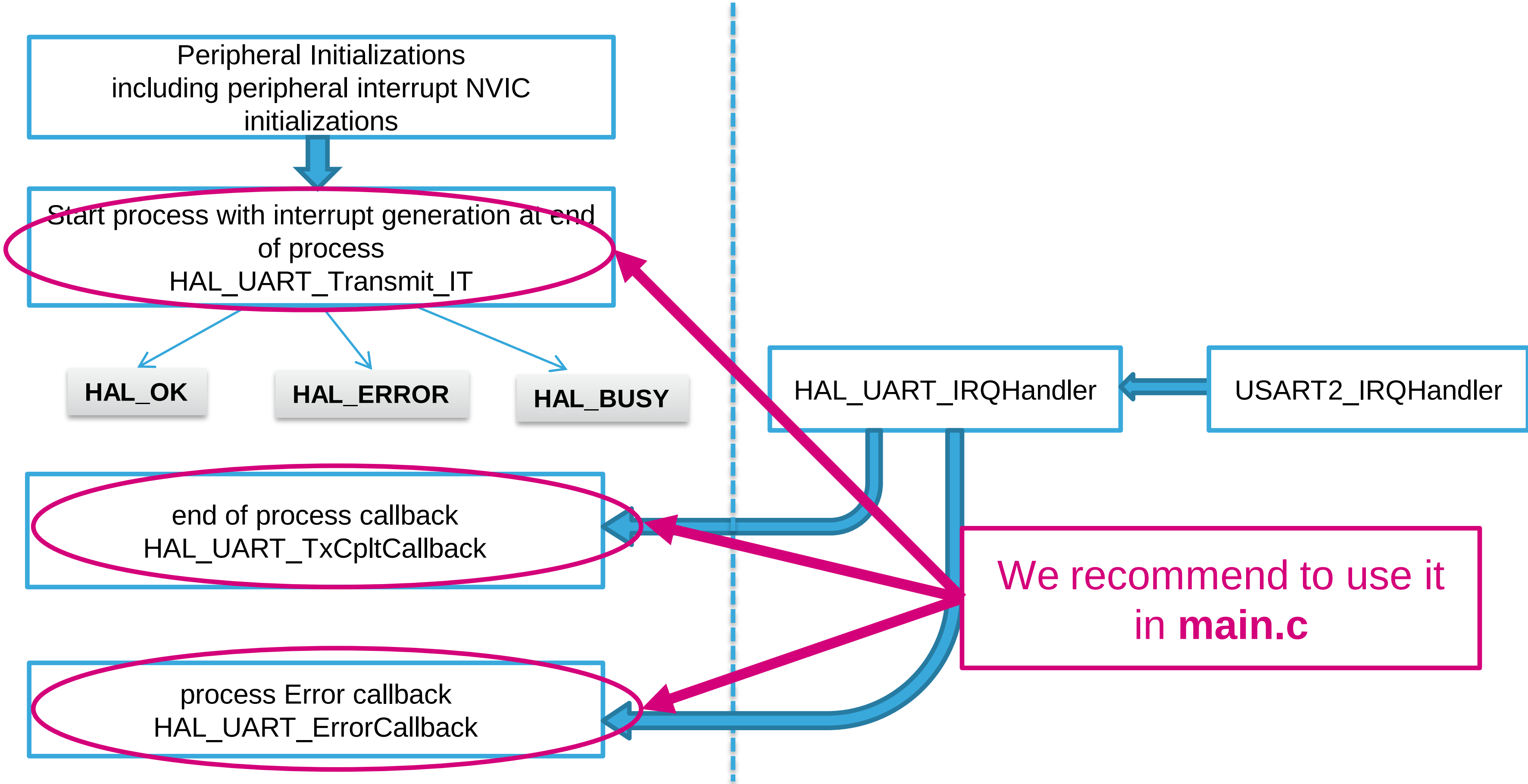
HAL_UART_IRQHandler

USART2_IRQHandler

Generated in **main.c** and **stm32f4xx_hal_msp.c**

2.1.2 Use UART with interrupt

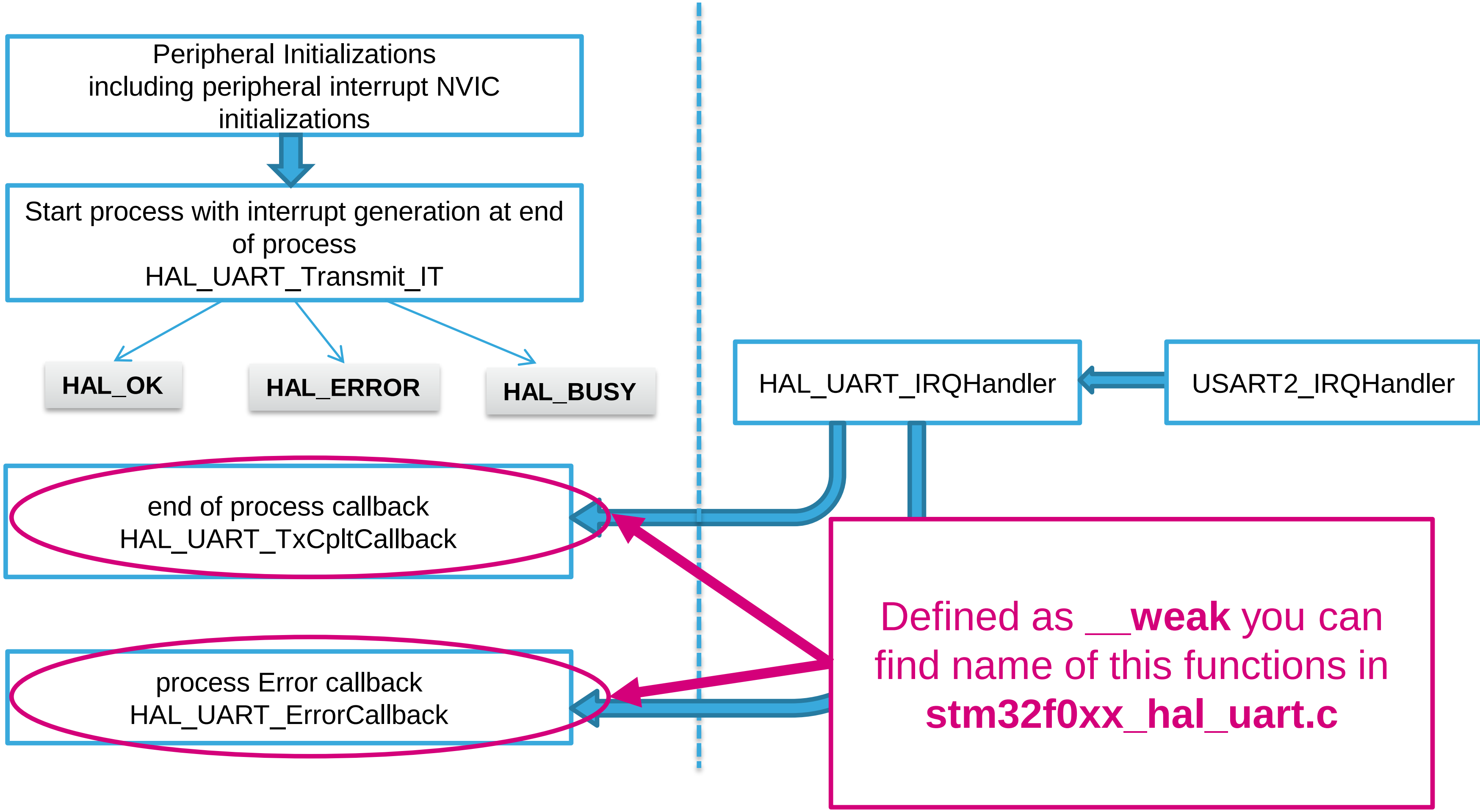
HAL Library UART with IT receive flow



We recommend to use it in **main.c**

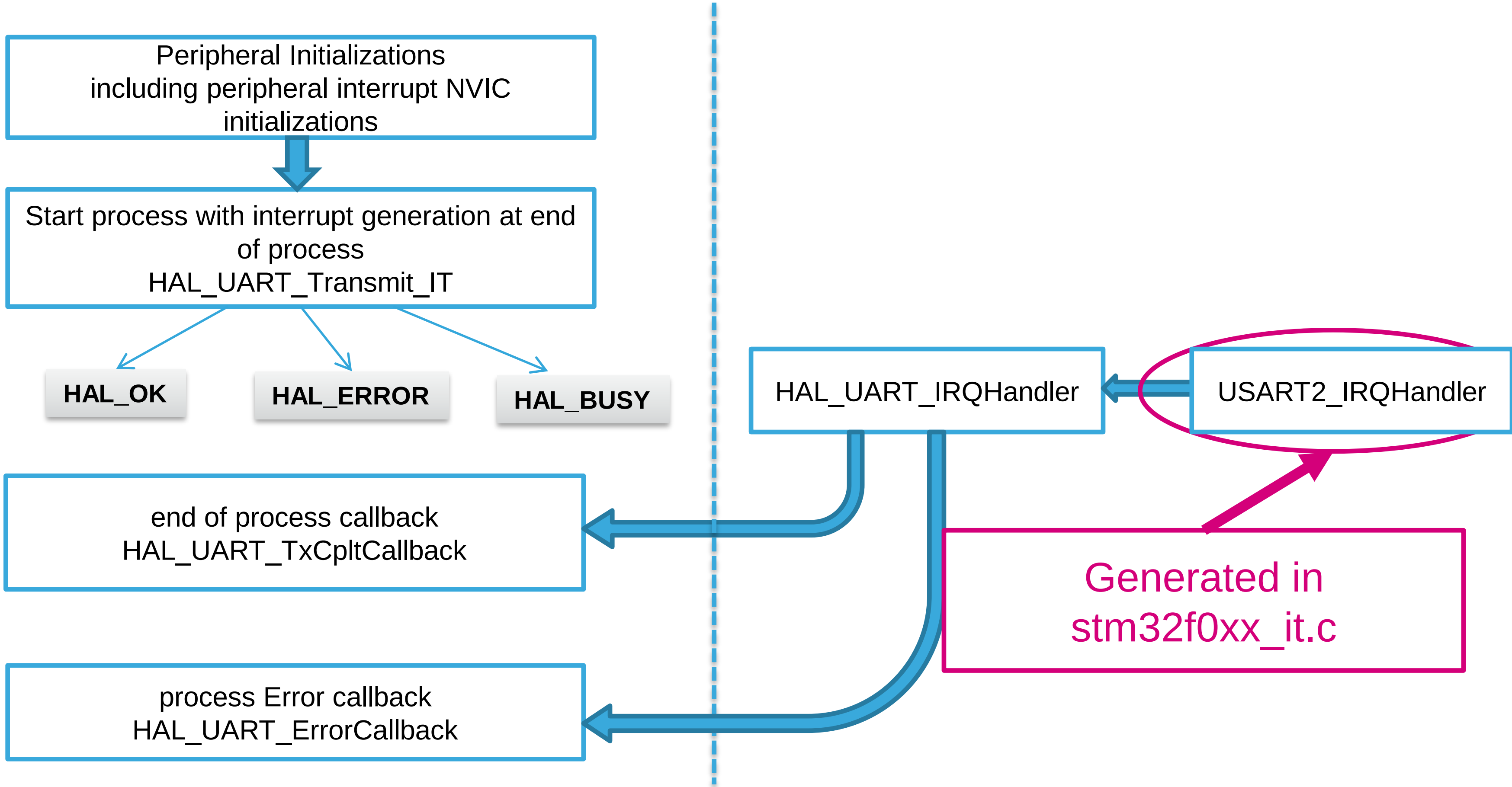
2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow


Peripheral Initializations including peripheral interrupt NVIC initializations
Start process with interrupt generation at end of process HAL_UART_Transmit_IT
HAL_OK
HAL_ERROR
HAL_BUSY
HAL_UART_IRQHandler
USART2_IRQHandler
end of process callback HAL_UART_TxCpltCallback
process Error callback HAL_UART_ErrorCallback
Defined as __**weak** you can find name of this functions in **stm32f0xx_hal_uart.c**

2.1.2 Use UART with interrupt

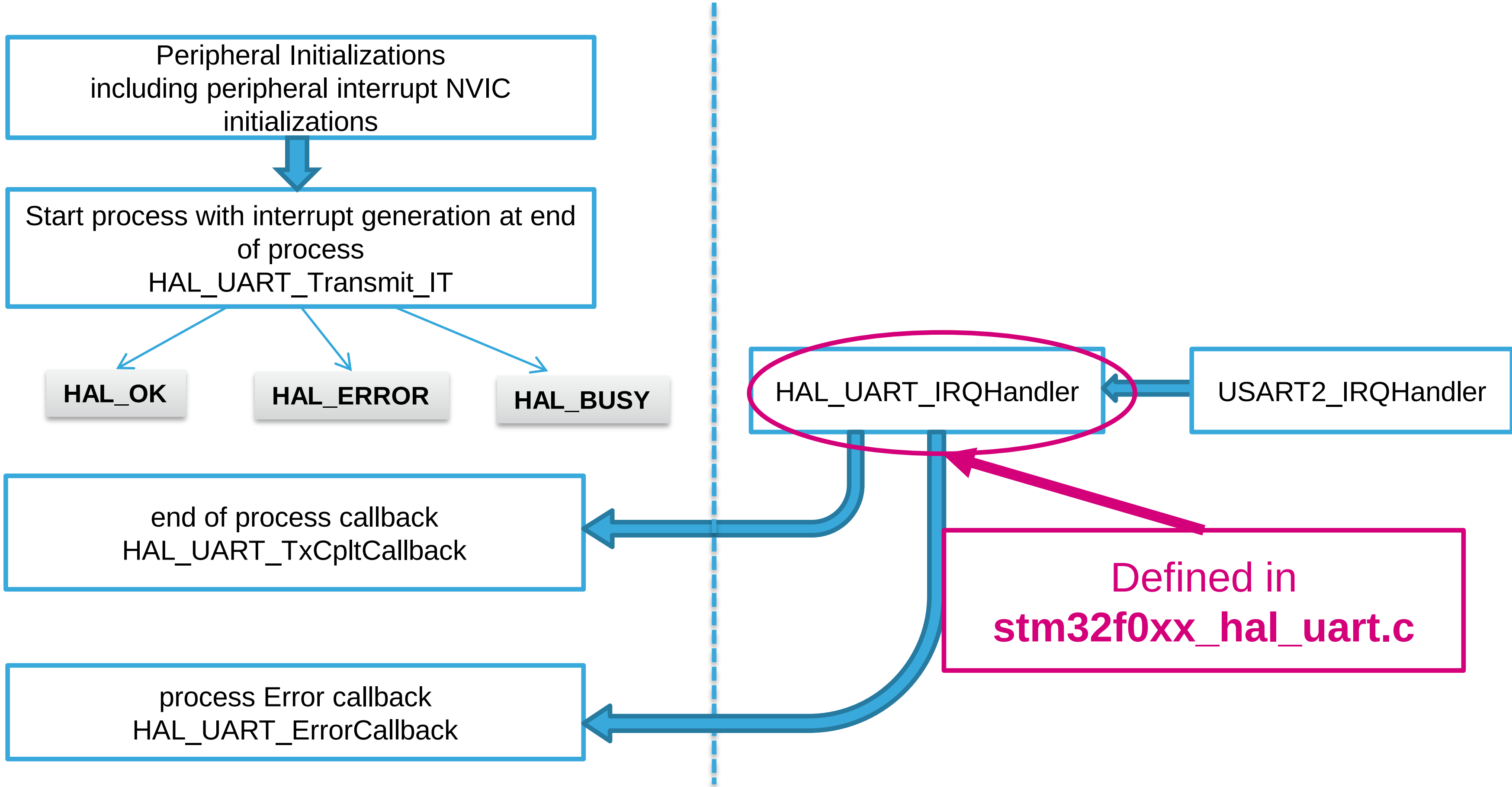
HAL Library UART with IT receive flow


Peripheral Initializations including peripheral interrupt NVIC initializations
Start process with interrupt generation at end of process HAL_UART_Transmit_IT
HAL_OK
HAL_ERROR
HAL_BUSY
HAL_UART_IRQHandler
USART2_IRQHandler
end of process callback HAL_UART_TxCpltCallback
process Error callback HAL_UART_ErrorCallback

Generated in stm32f0xx_it.c

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

HAL_UART_IRQHandler

USART2_IRQHandler

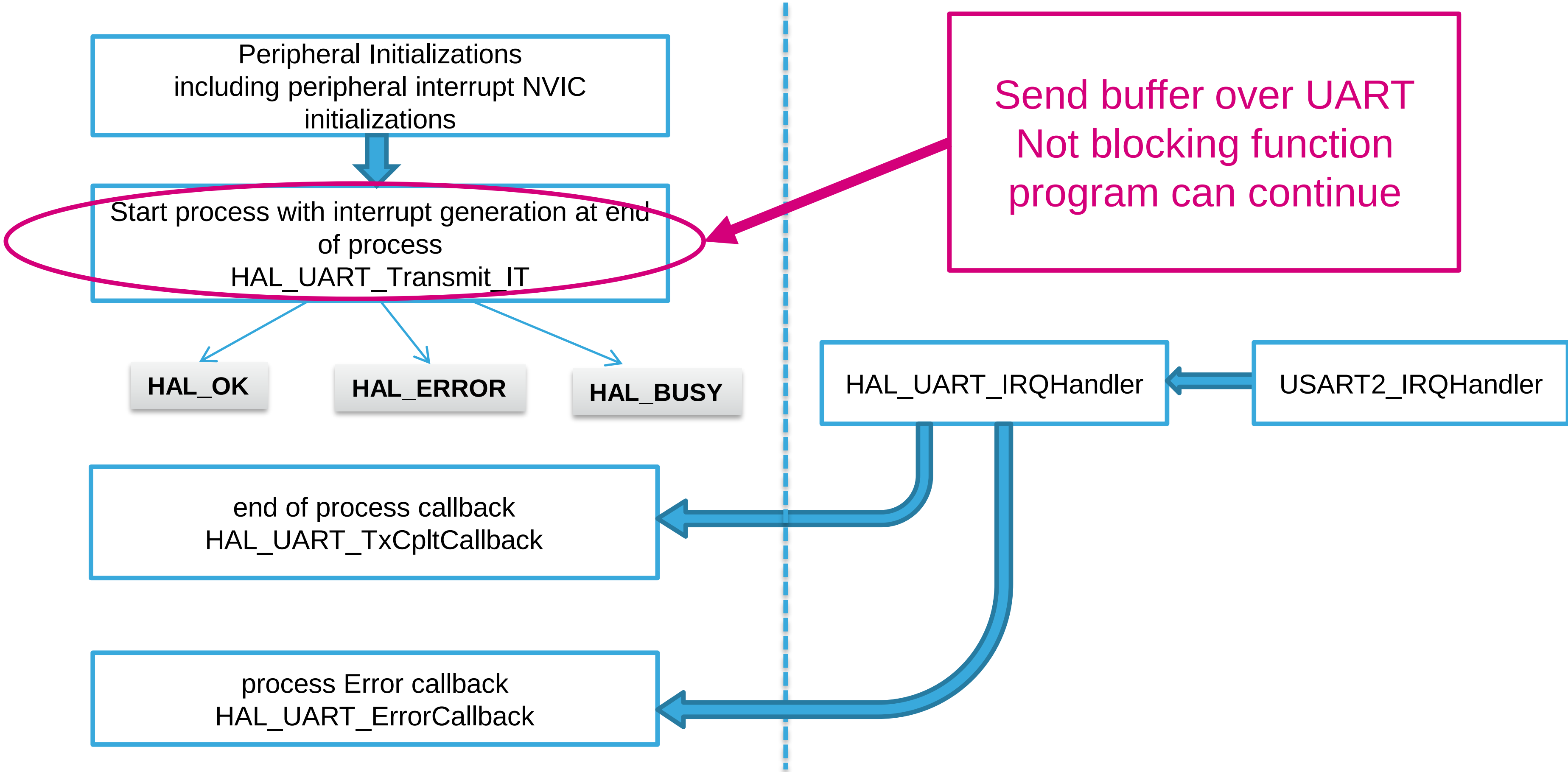
Defined in **stm32f0xx_hal_uart.c**

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

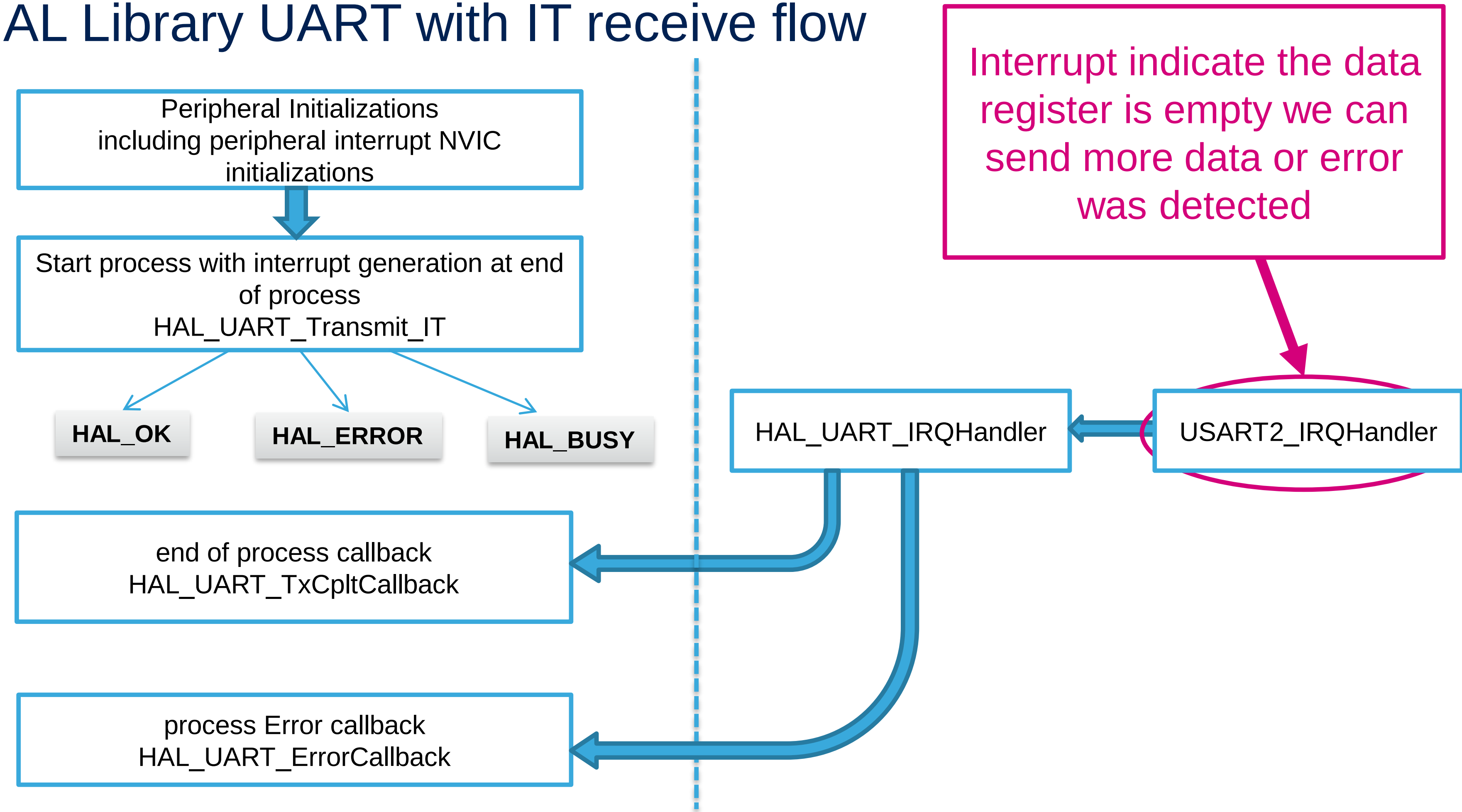
HAL_UART_IRQHandler

USART2_IRQHandler

Send buffer over UART
Not blocking function
program can continue

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process
HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

end of process callback
HAL_UART_TxCpltCallback

process Error callback
HAL_UART_ErrorCallback

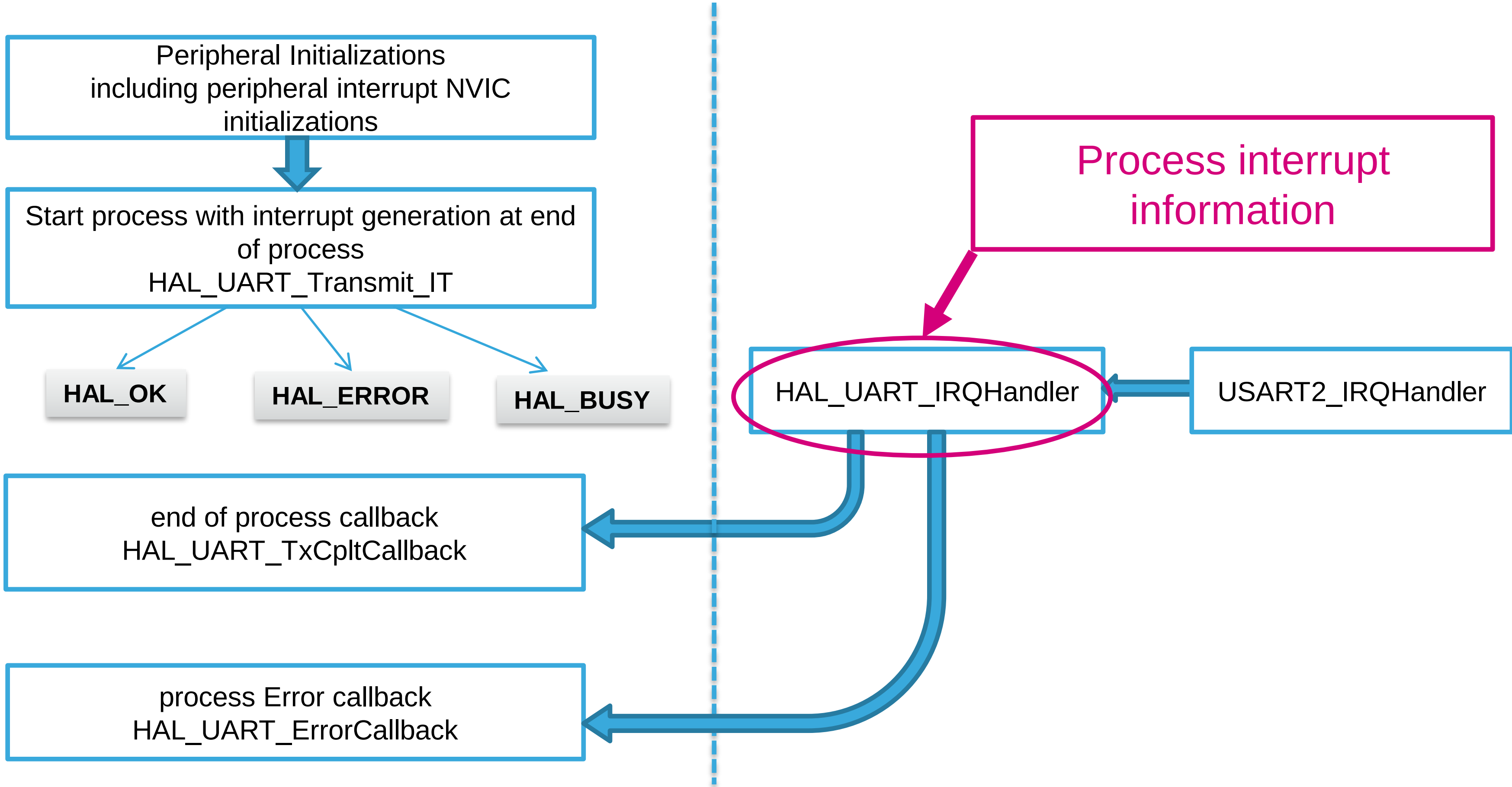
HAL_UART_IRQHandler

USART2_IRQHandler

Interrupt indicate the data register is empty we can send more data or error was detected

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

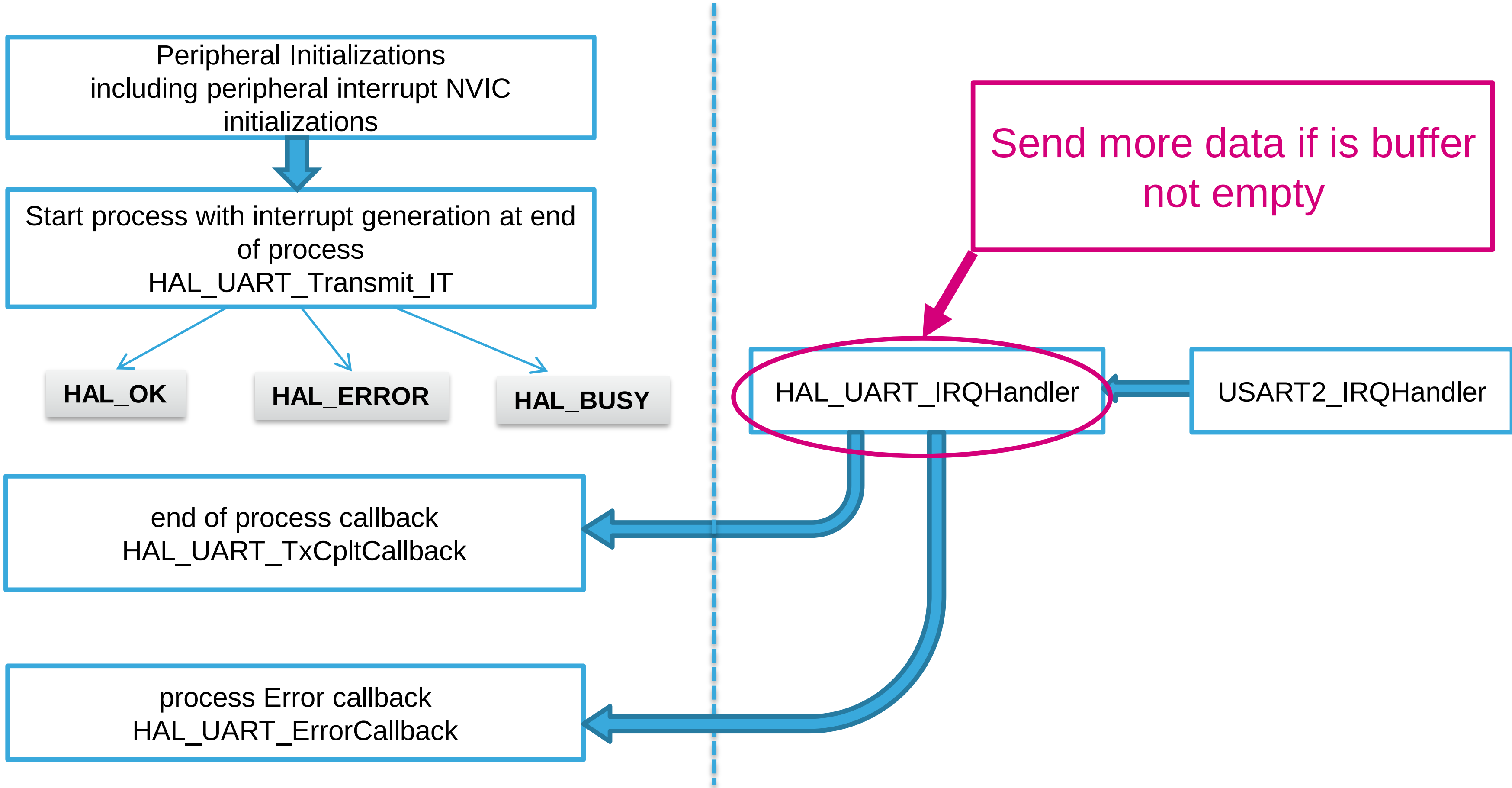
Process interrupt information

HAL_UART_IRQHandler

USART2_IRQHandler

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

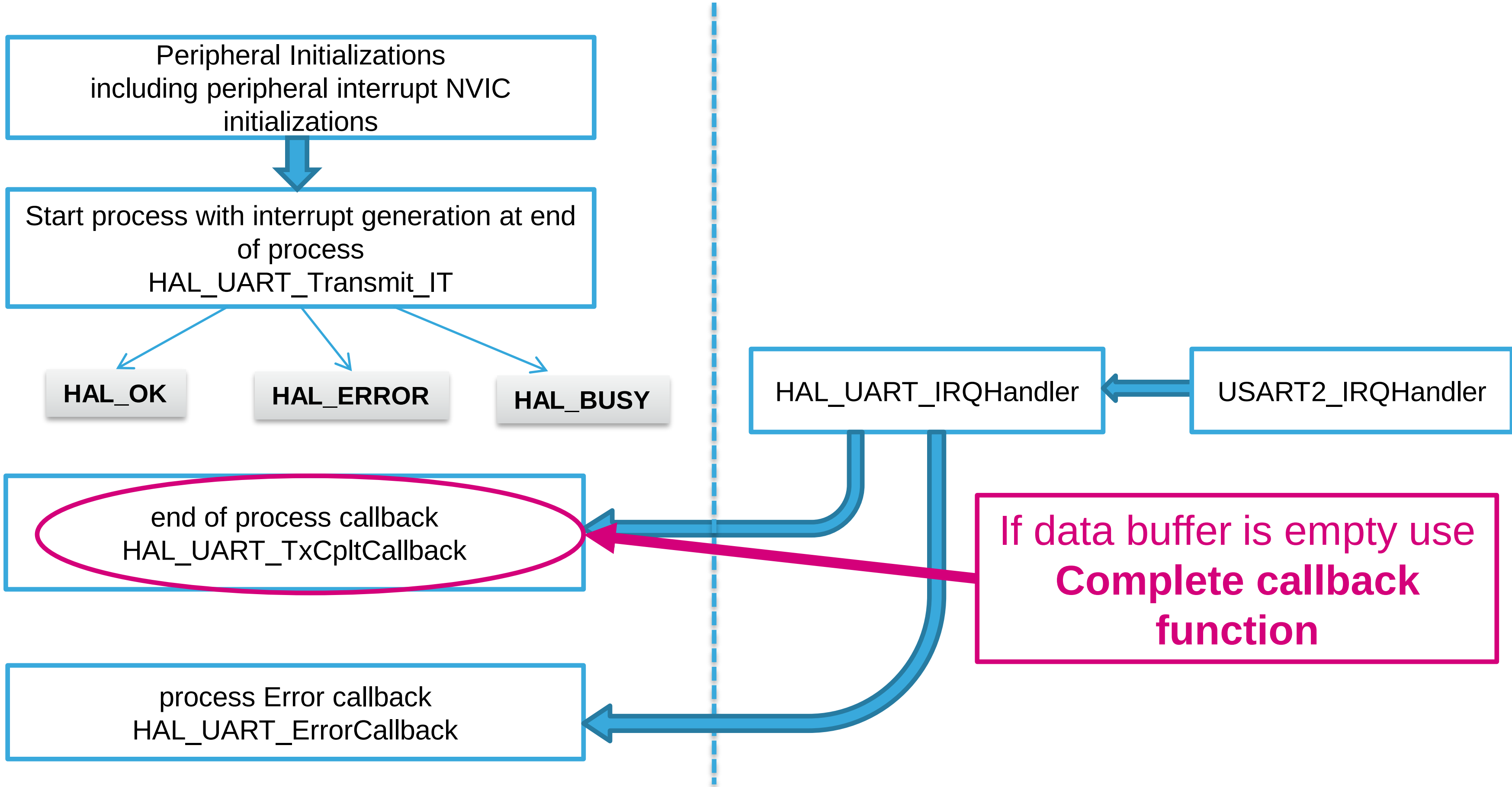
HAL_UART_IRQHandler

USART2_IRQHandler

Send more data if is buffer not empty

2.1.2 Use UART with interrupt

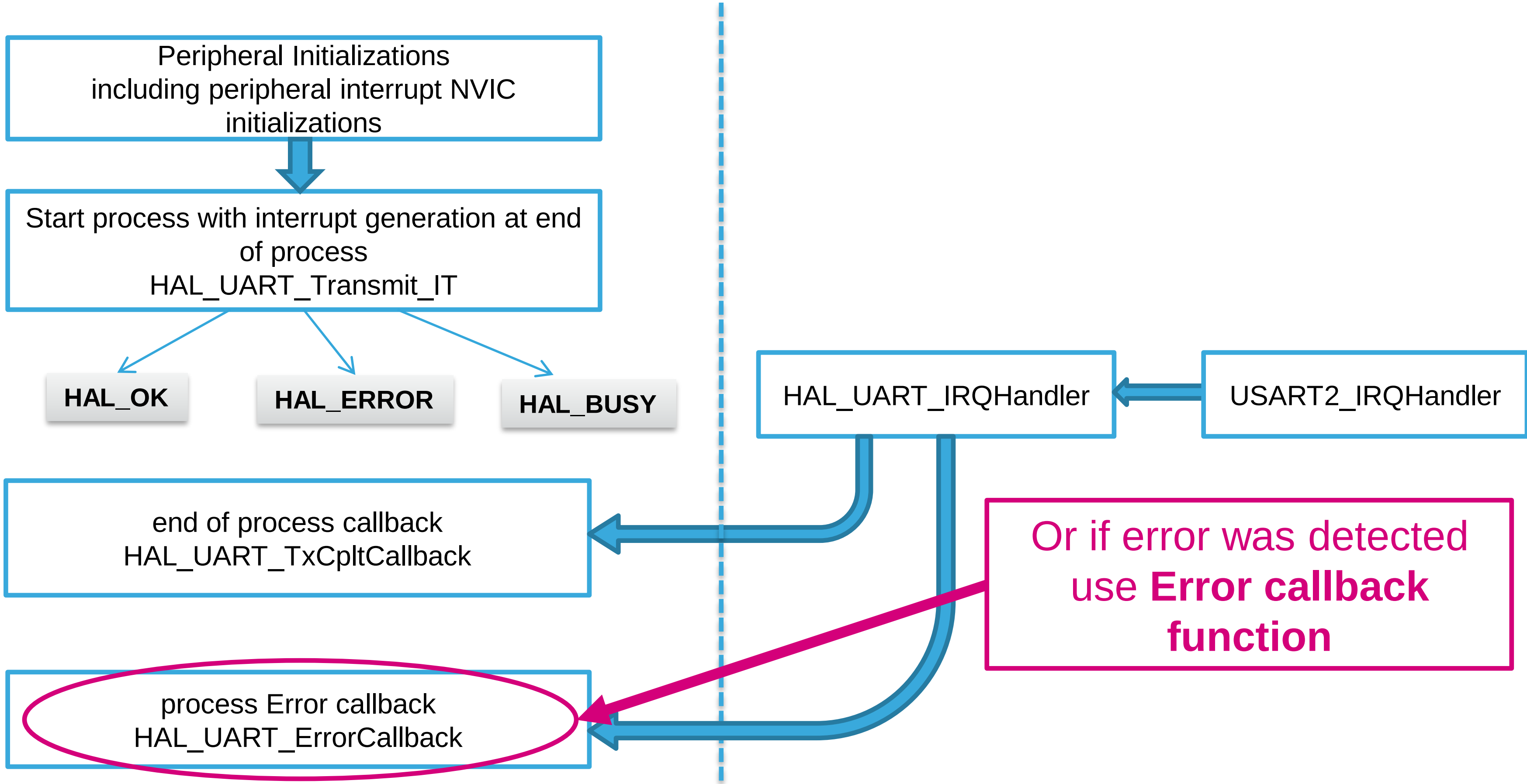
HAL Library UART with IT receive flow


Peripheral Initializations including peripheral interrupt NVIC initializations
Start process with interrupt generation at end of process HAL_UART_Transmit_IT
HAL_OK
HAL_ERROR
HAL_BUSY
HAL_UART_IRQHandler
USART2_IRQHandler
end of process callback HAL_UART_TxCpltCallback
process Error callback HAL_UART_ErrorCallback

If data buffer is empty use **Complete callback function**

2.1.2 Use UART with interrupt

HAL Library UART with IT receive flow



Peripheral Initializations including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_UART_Transmit_IT

HAL_OK

HAL_ERROR

HAL_BUSY

HAL_UART_IRQHandler

USART2_IRQHandler

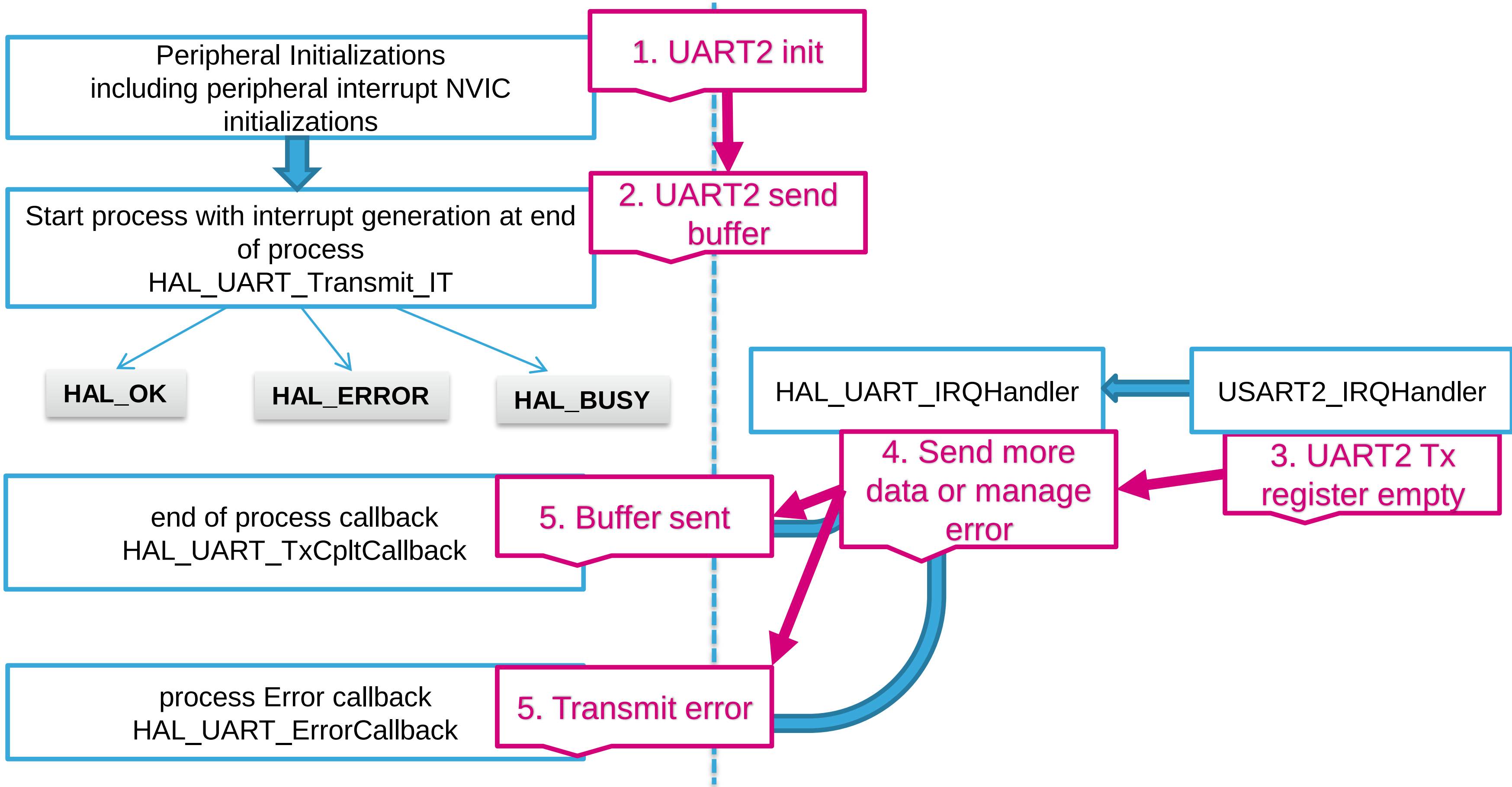
end of process callback HAL_UART_TxCpltCallback

process Error callback HAL_UART_ErrorCallback

Or if error was detected use **Error callback function**

Use UART with interrupt

HAL Library UART with IT receive flow



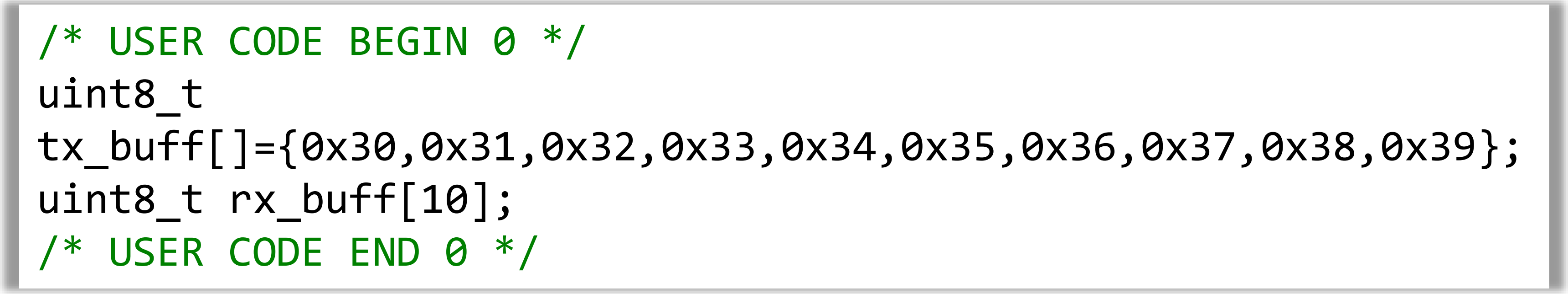
Peripheral Initializations including peripheral interrupt NVIC initializations
1. UART2 init
Start process with interrupt generation at end of process HAL_UART_Transmit_IT
2. UART2 send buffer
HAL_OK
HAL_ERROR
HAL_BUSY
HAL_UART_IRQHandler
USART2_IRQHandler
4. Send more data or manage error
3. UART2 Tx register empty
end of process callback HAL_UART_TxCpltCallback
5. Buffer sent
process Error callback HAL_UART_ErrorCallback
5. Transmit error

2.1.2 Use UART with interrupt

- Open the project in our IDE
  - The functions we want to put into main.c
  - Between */* USER CODE BEGIN 2 */* and */* USER CODE END 2 */* tags
- For transmit use function
  - HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
- For receive use function
  - HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);

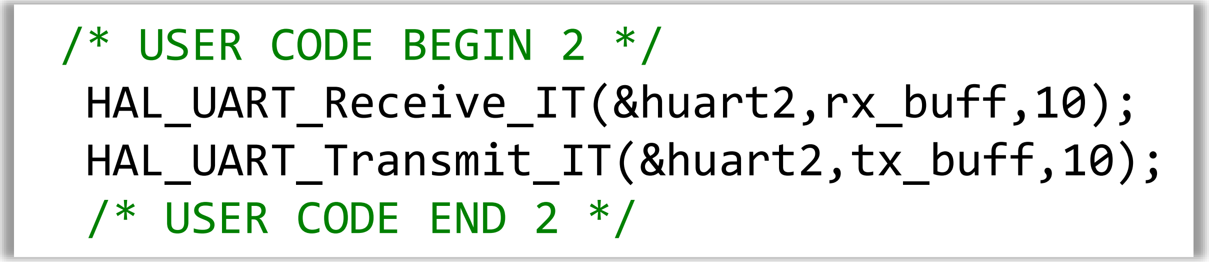
2.1.2 Use UART with interrupt

- Buffer definition



```
/* USER CODE BEGIN 0 */
uint8_t
tx_buff[]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
uint8_t rx_buff[10];
/* USER CODE END 0 */
```

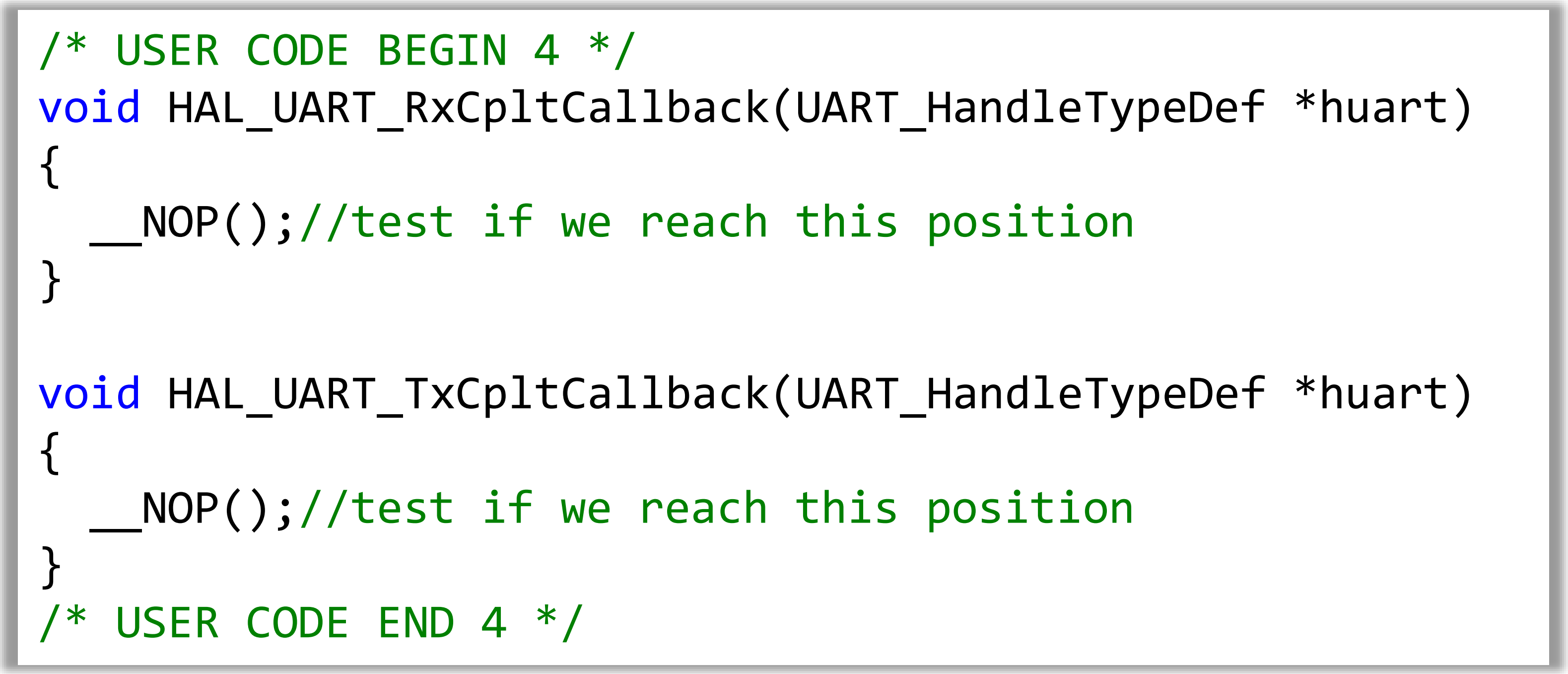
- Sending and receiving methods



```
 /* USER CODE BEGIN 2 */
  HAL_UART_Receive_IT(&huart2,rx_buff,10);
  HAL_UART_Transmit_IT(&huart2,tx_buff,10);
  /* USER CODE END 2 */
```

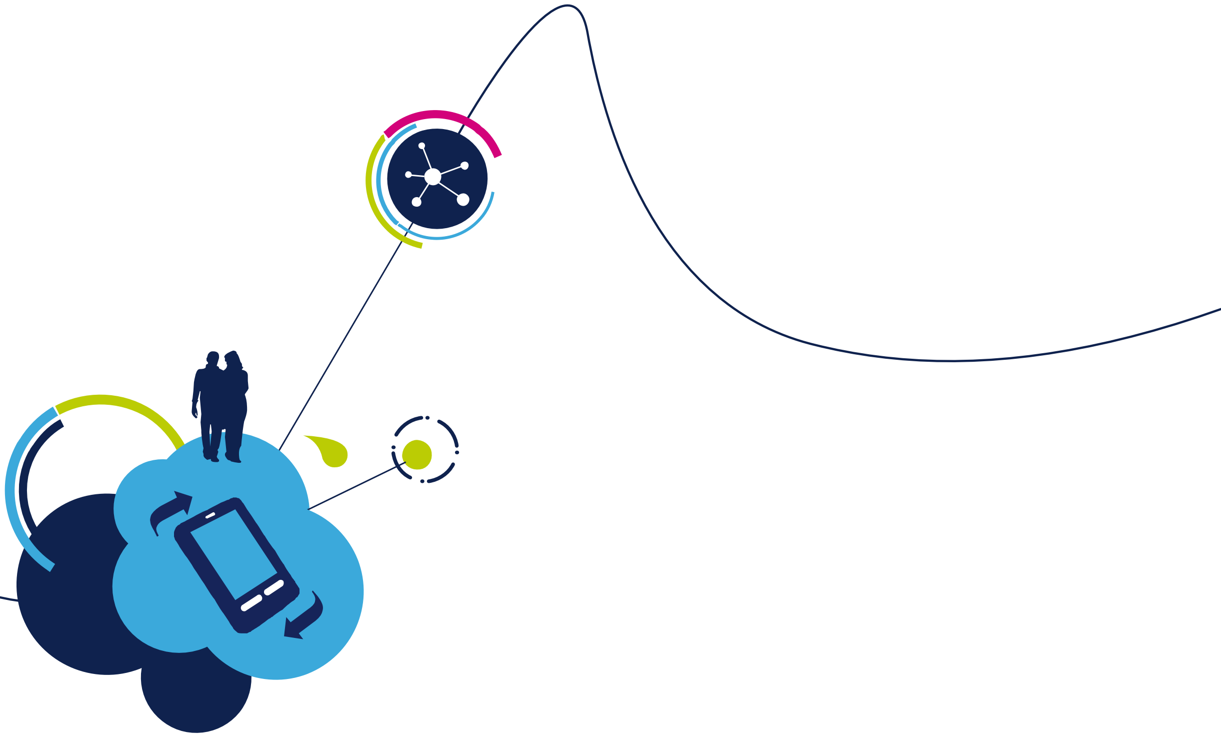
2.1.2 Use UART with interrupt

- Complete callback check
  - We can put brakepoints on NOPs to watch if we send or receive complete buffer



```
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  __NOP();//test if we reach this position
}

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
  __NOP();//test if we reach this position
}
/* USER CODE END 4 */
```



Appendix B Documents

CubeMX documentation

- CubeMX user manual UM1718
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00104712.pdf
- CubeMX release note RN0094
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00104712.pdf
- CubeMX technical note TN0072
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/technical_note/CD00214439.pdf

B

STM32F429i-Discovery documentation

- STM32F429i-Discovery page
  - http://www.st.com/web/en/catalog/tools/FM116/SC959/SS1532/LN1848/PF259090?s_searchtype=keyword
- STM32F429i-Discovery user manual with discovery schematics
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00093903.pdf