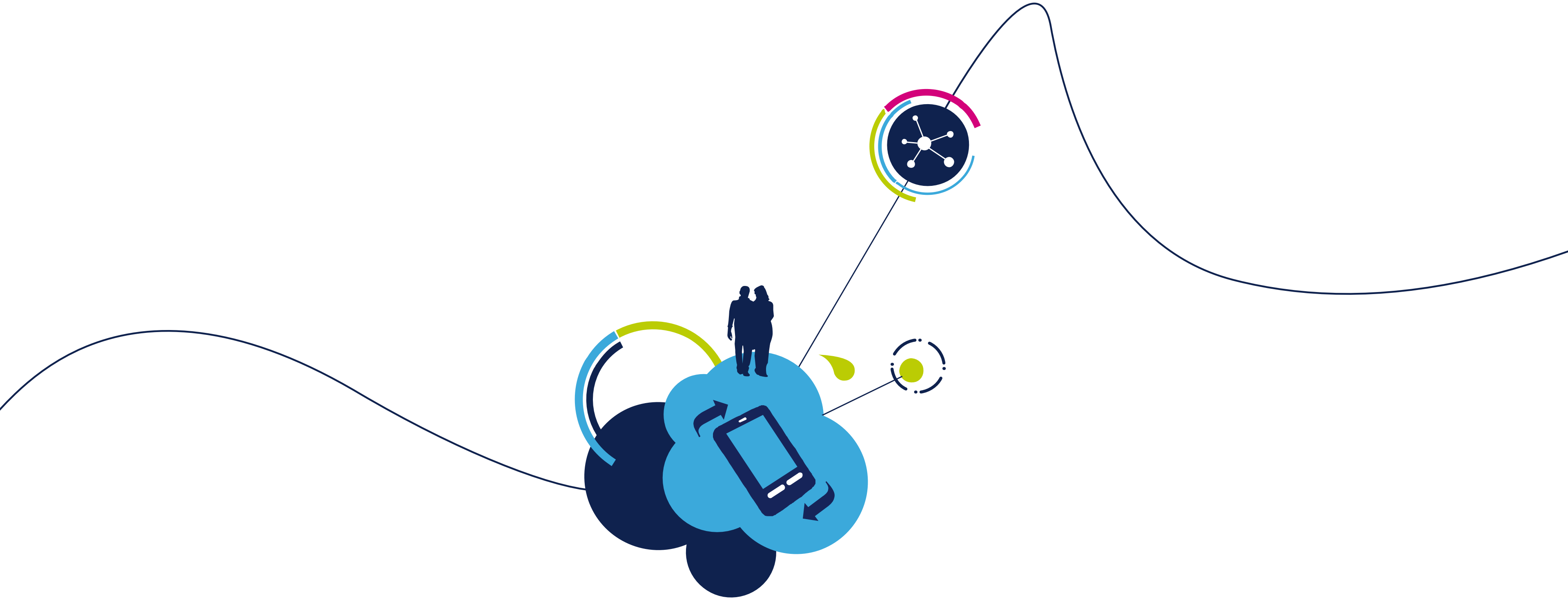


UART DMA lab **11**

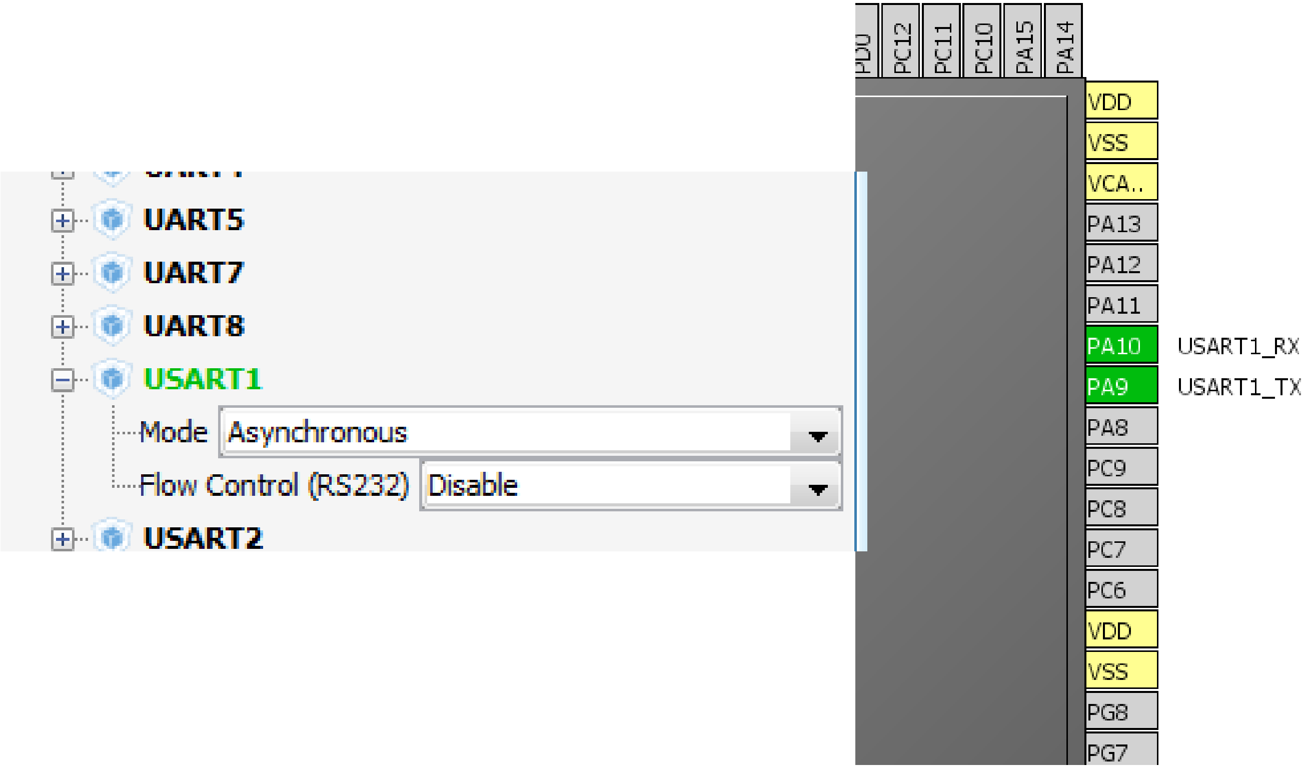
life.augmented

11 Use UART with DMA transfer

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

11 Use UART with DMA transfer

- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F4 > STM32F429/439 > LQFP144 > STM32F439ZITx
- Pin selection
  - It will be same as previous lab we use again PA9 and PA10



USART1

Mode: Asynchronous

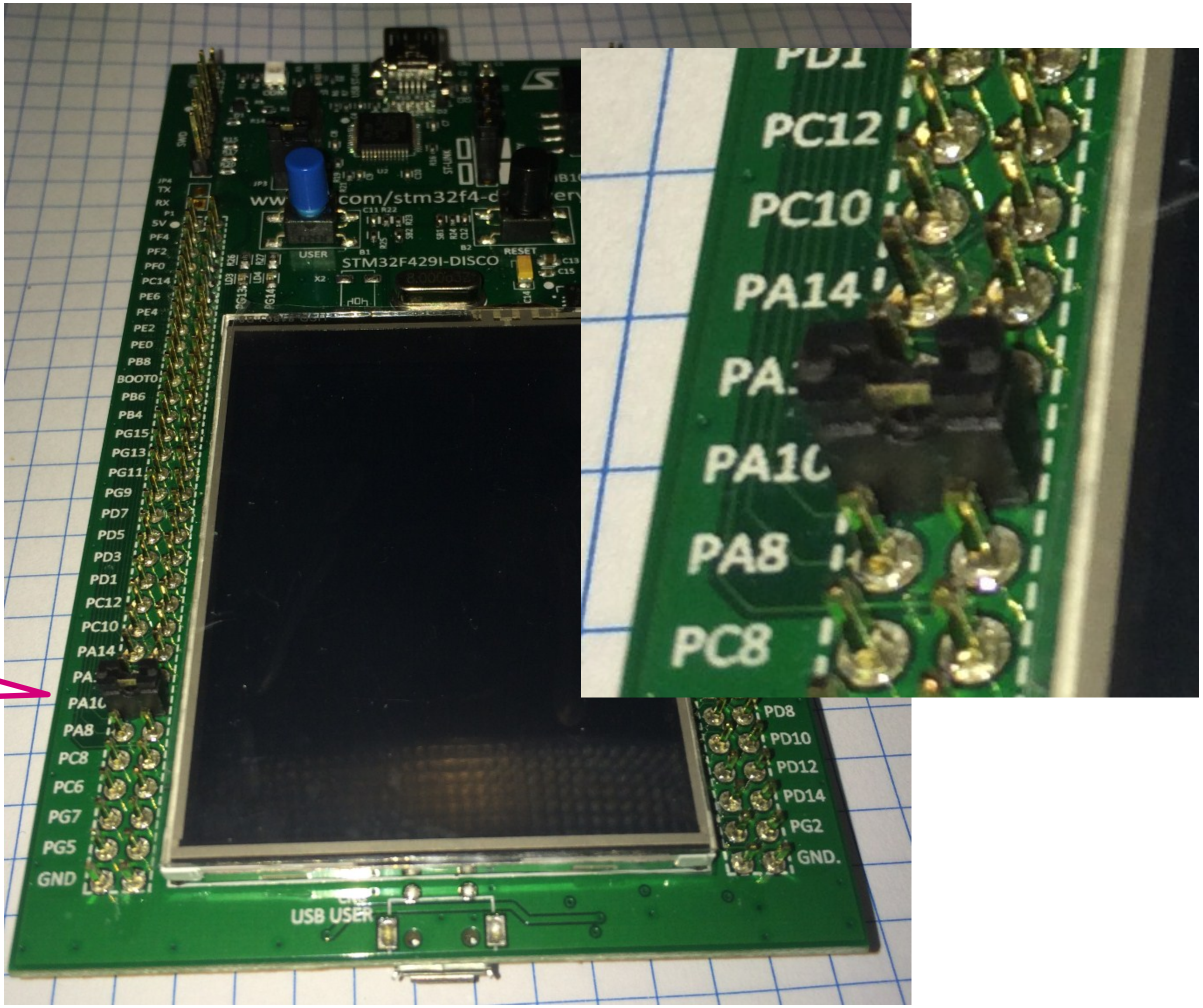
Flow Control (RS232): Disable

PA10 USART1_RX

PA9 USART1_TX

Use UART with DMA transfer

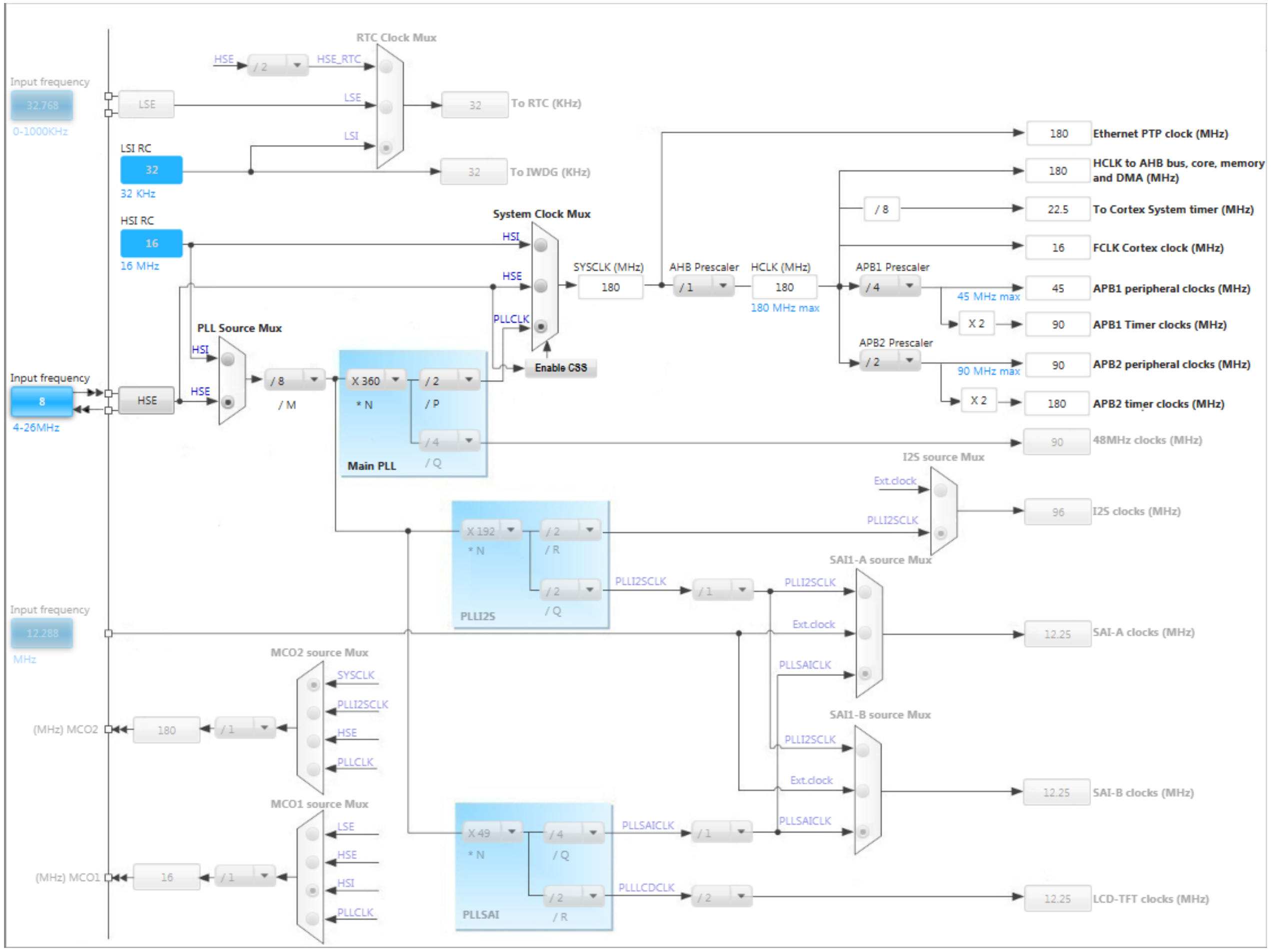
- Hardware preparation
  - We connect selected pins together by jumper, this help us to create loopback on UART



Hardware loopback

Use UART with DMA transfer

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



Input frequency 32.768 0-1000KHz

LSI RC 32 32 KHz

HSI RC 16 16 MHz

Input frequency 8 4-26MHz

Input frequency 12.288 MHz

32 To RTC (KHz)

32 To IWDG (KHz)

PLL Source Mux: HSE

/ 8 / M

Main PLL: X 360 * N, / 2 / P, / 4 / Q

PLLI2S: X 192 * N, / 2 / R, / 2 / Q

PLLSAI: X 49 * N, / 4 / Q, / 2 / R

System Clock Mux: PLLCLK

SYSCLK (MHz) 180

AHB Prescaler / 1

HCLK (MHz) 180 (180 MHz max)

APB1 Prescaler / 4 (45 MHz max)

APB2 Prescaler / 2 (90 MHz max)

180 Ethernet PTP clock (MHz)

180 HCLK to AHB bus, core, memory and DMA (MHz)

/ 8 22.5 To Cortex System timer (MHz)

16 FCLK Cortex clock (MHz)

45 APB1 peripheral clocks (MHz)

X 2 90 APB1 Timer clocks (MHz)

90 APB2 peripheral clocks (MHz)

X 2 180 APB2 timer clocks (MHz)

90 48MHz clocks (MHz)

96 I2S clocks (MHz)

/ 1 12.25 SAI-A clocks (MHz)

/ 1 12.25 SAI-B clocks (MHz)

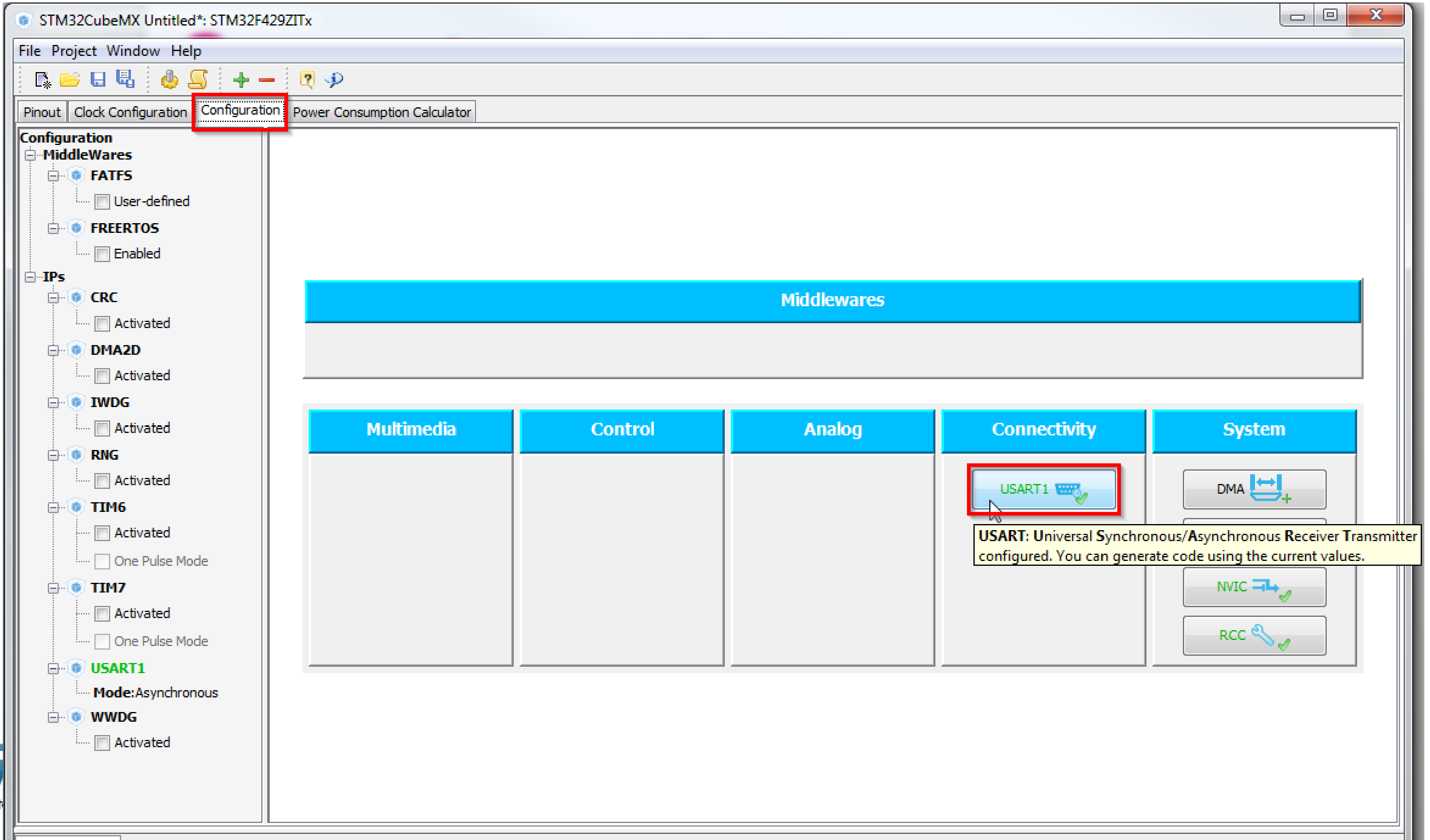
/ 2 12.25 LCD-TFT clocks (MHz)

(MHz) MCO2 180 / 1

(MHz) MCO1 16 / 1

Use UART with DMA transfer

- CubeMX UART configuration
  - Tab>Configuration>Connectivity>USART1

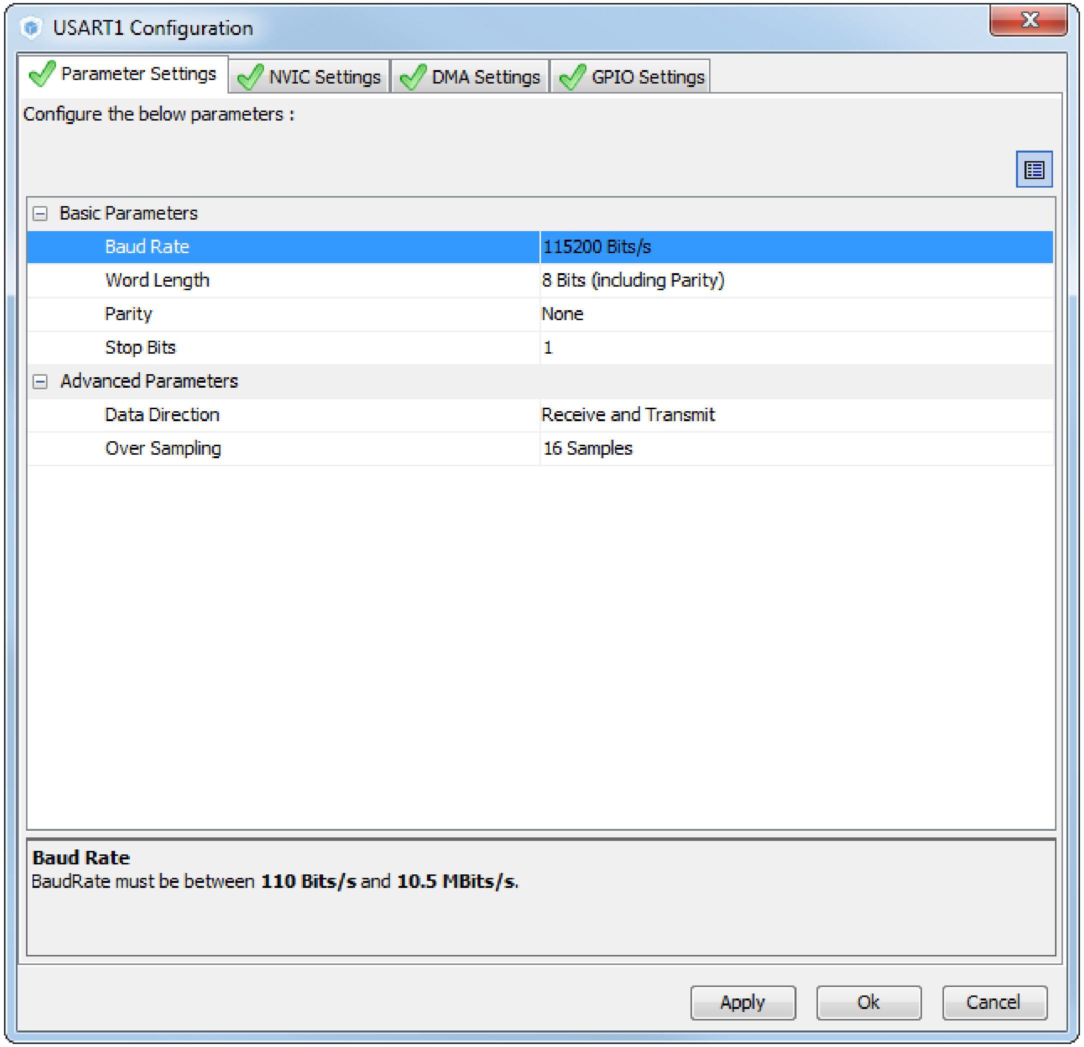


Configuration
- MiddleWares
  - FATFS
    - □ User-defined
  - FREERTOS
    - □ Enabled
- IPs
  - CRC
    - □ Activated
  - DMA2D
    - □ Activated
  - IWDG
    - □ Activated
  - RNG
    - □ Activated
  - TIM6
    - □ Activated
    - □ One Pulse Mode
  - TIM7
    - □ Activated
    - □ One Pulse Mode
  - USART1
    - Mode:Asynchronous
  - WWDG
    - □ Activated

USART: **U**niversal **S**ynchronous/**A**synchronous **R**eceiver **T**ransmitter
configured. You can generate code using the current values.

11 Use UART with DMA transfer

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



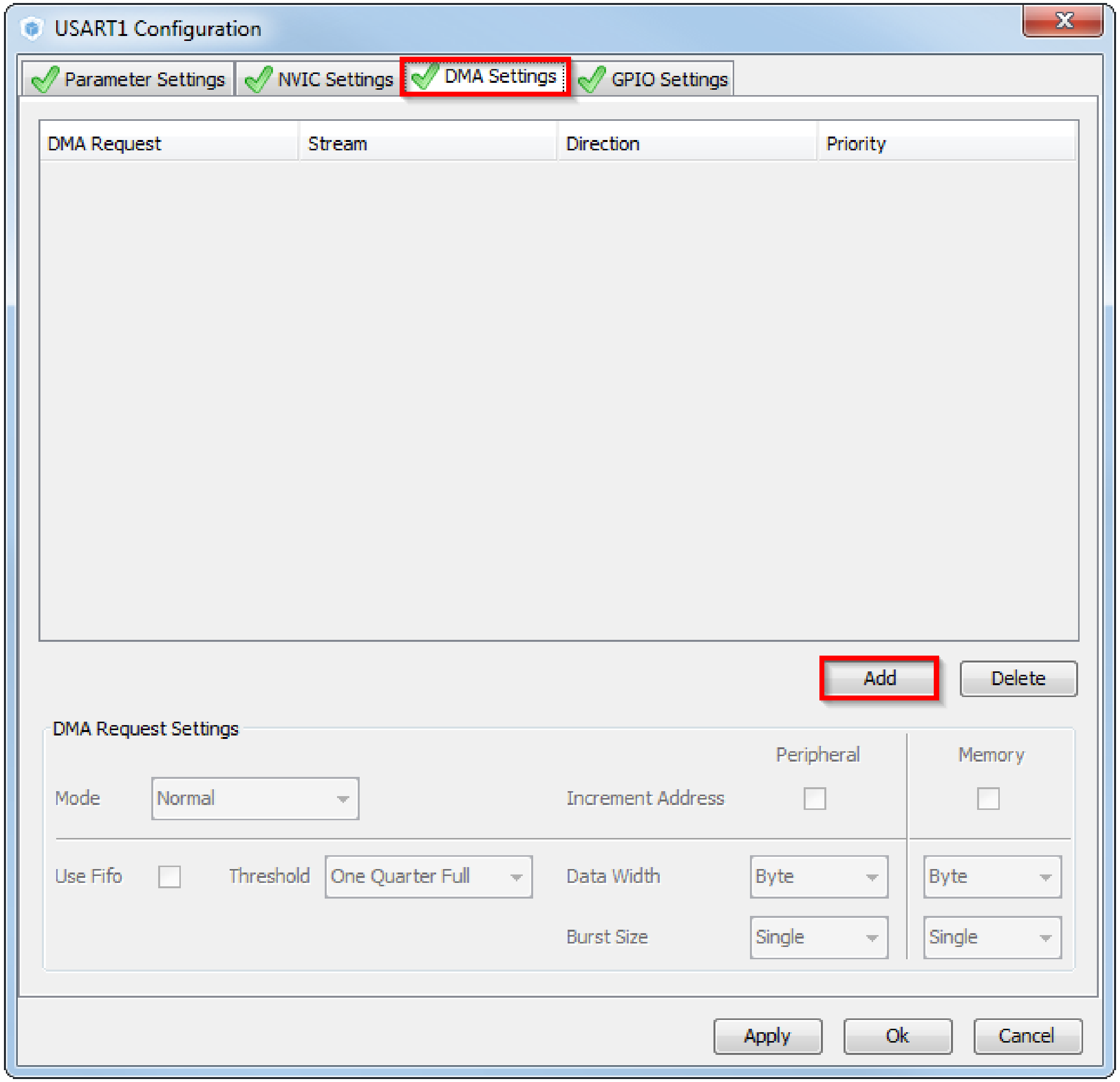
Configure the below parameters :

⊟ Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
⊟ Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

Baud Rate
BaudRate must be between **110 Bits/s** and **10.5 MBits/s**.

Use UART with DMA transfer

- CubeMX USART configuration DMA settings
  - TAB>DMA Settings
  - Button ADD



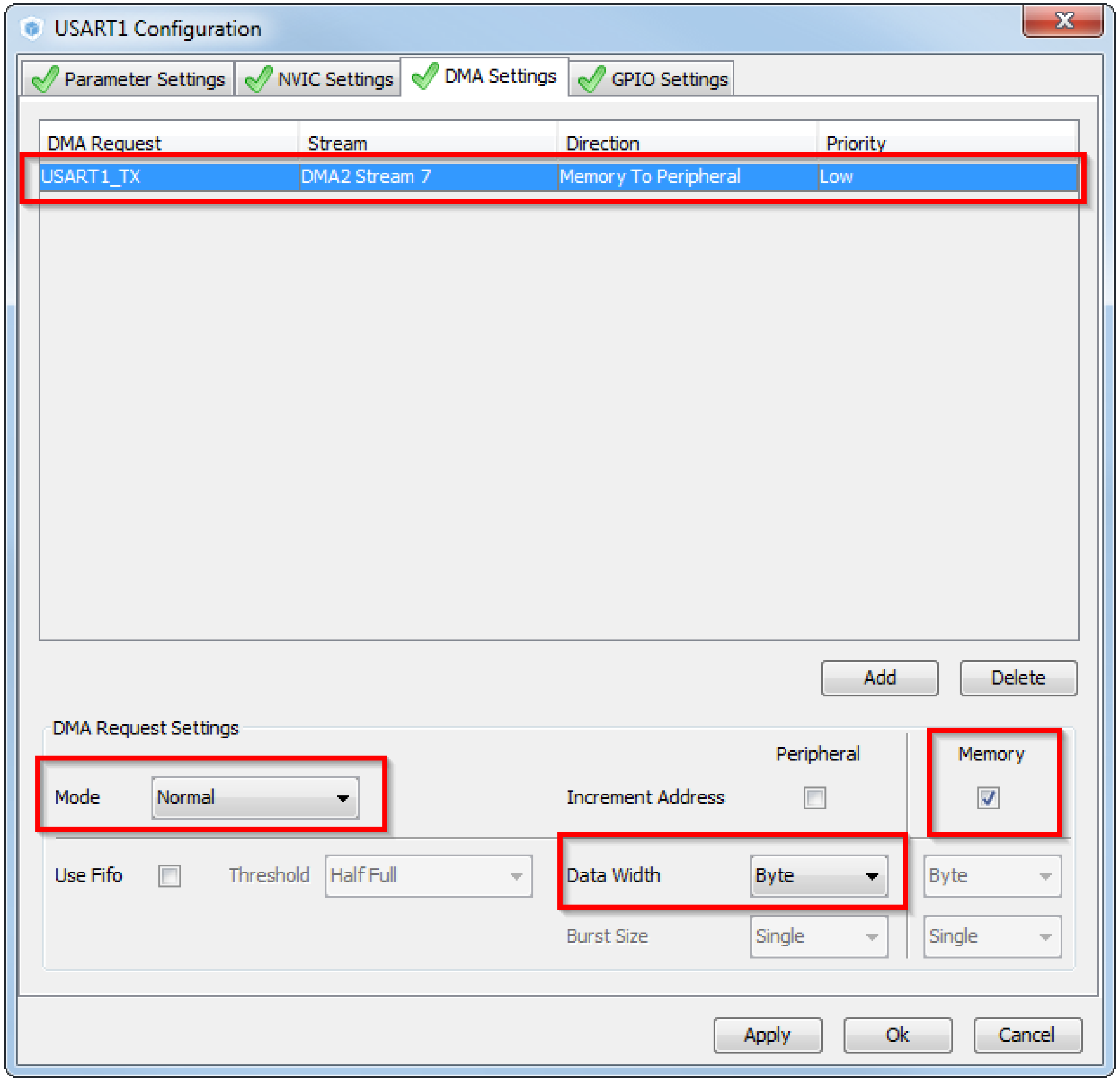
DMA Request	Stream	Direction	Priority

DMA Request Settings

		Peripheral	Memory
Mode	Normal		
	Increment Address	□	□
Use Fifo □	Threshold One Quarter Full		
	Data Width	Byte	Byte
	Burst Size	Single	Single

Use UART with DMA transfer

- CubeMX USART configuration DMA Tx settings
  - Set USART1_TX request
  - Memory to peripheral direction
  - Normal mode
  - Byte data width
  - Increment memory address



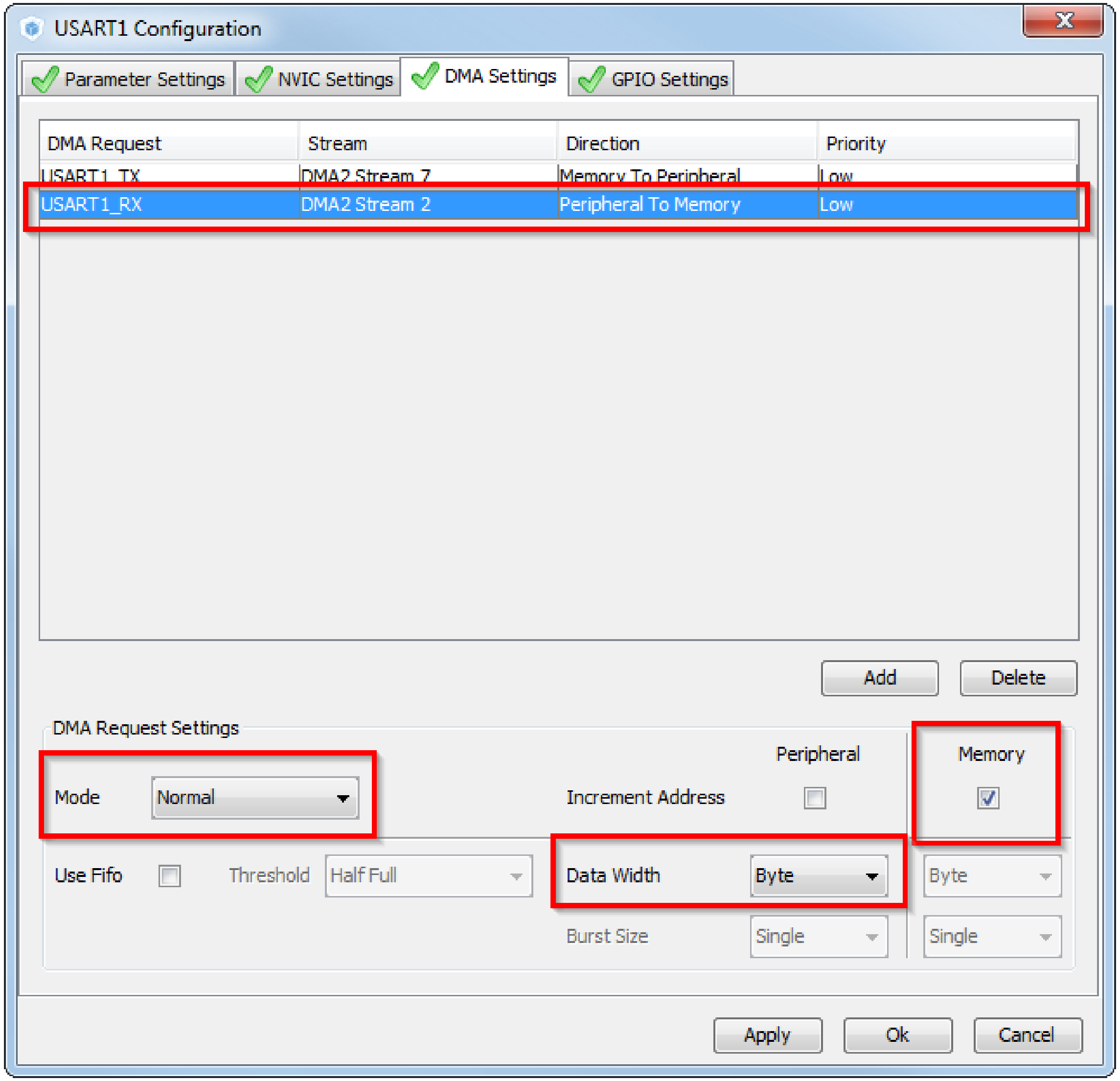
DMA Request	Stream	Direction	Priority
USART1_TX	DMA2 Stream 7	Memory To Peripheral	Low

DMA Request Settings

		Peripheral	Memory
Mode	Normal		
	Increment Address	□	☑
Use Fifo □	Threshold Half Full		
	Data Width	Byte	Byte
	Burst Size	Single	Single

11 Use UART with DMA transfer

- CubeMX USART configuration DMA Rx settings
  - Button ADD
  - Set USART1_RX request
  - Peripheral to memory direction
  - Normal mode
  - Byte data width
  - Increment memory address



DMA Request	Stream	Direction	Priority
USART1_TX	DMA2 Stream 7	Memory To Peripheral	Low
USART1_RX	DMA2 Stream 2	Peripheral To Memory	Low

DMA Request Settings

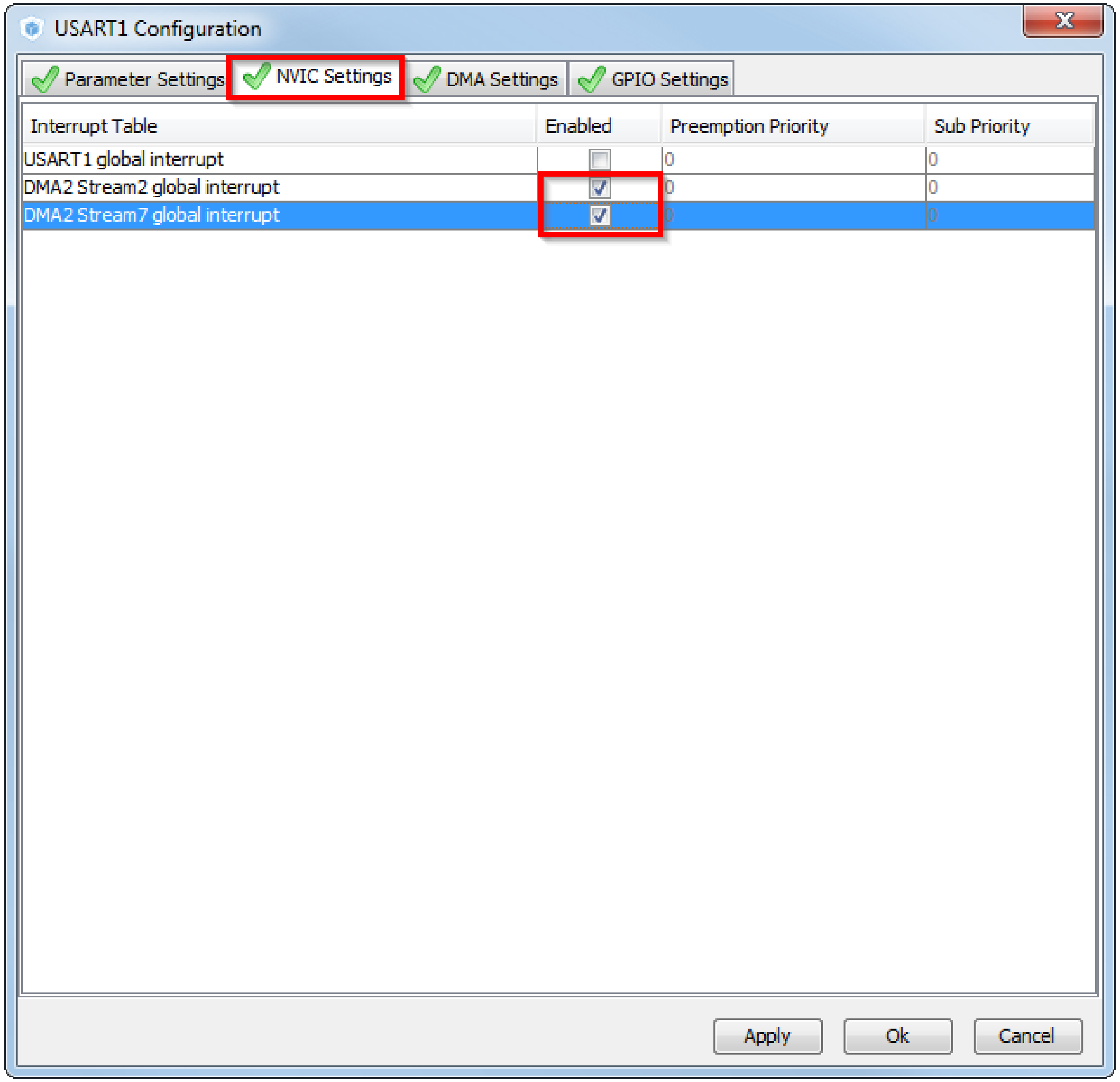
Mode: Normal

	Peripheral	Memory
Increment Address	□	☑
Data Width	Byte	Byte
Burst Size	Single	Single

Use Fifo □ Threshold Half Full

Use UART with DMA transfer

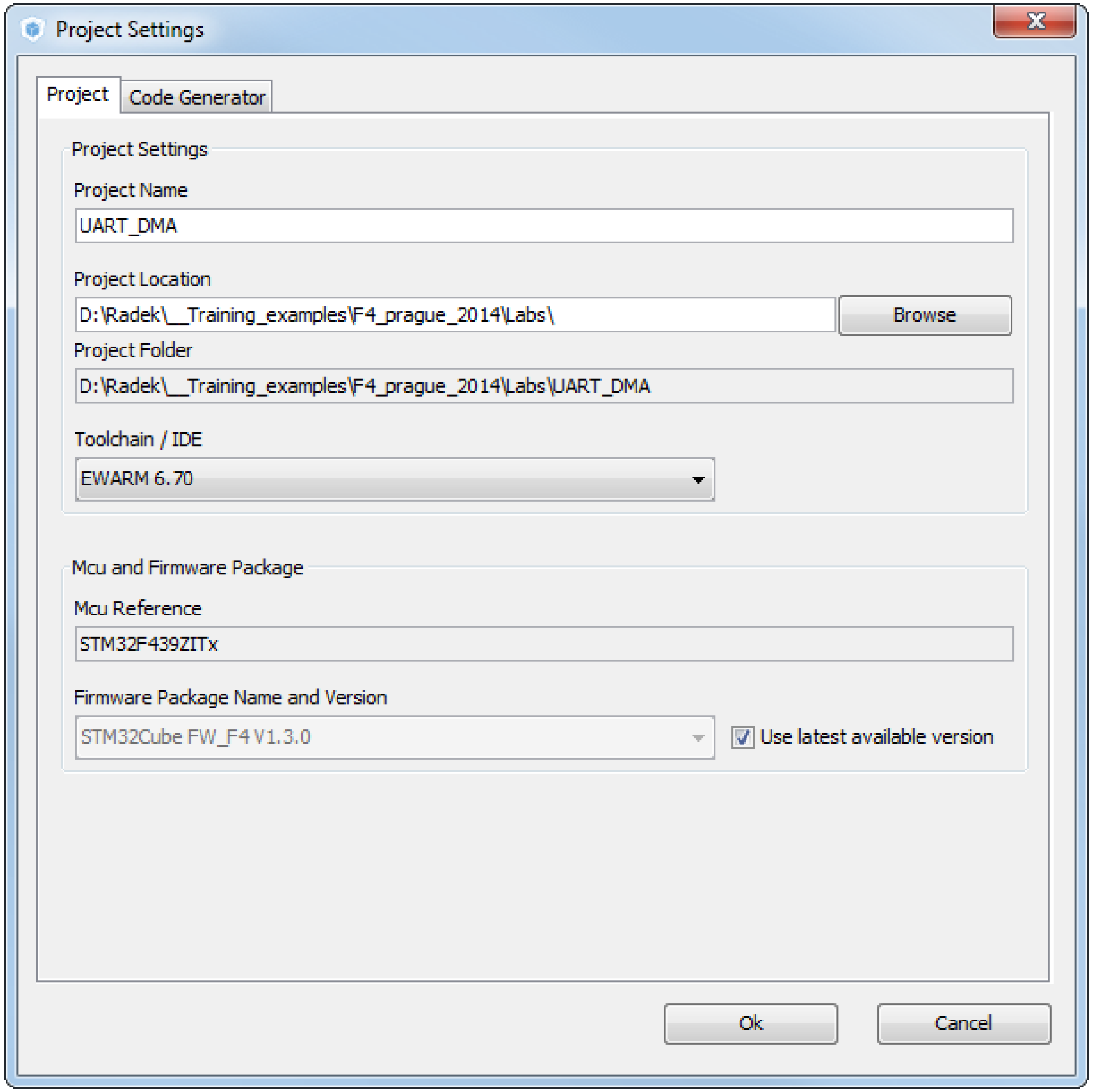
- CubeMX USART configuration NVIC settings
  - TAB>NVIC Settings
  - Enable DMA2 interrupts for USART1
  - Button OK



Interrupt Table	Enabled	Preemption Priority	Sub Priority
USART1 global interrupt	□	0	0
DMA2 Stream2 global interrupt	☑	0	0
DMA2 Stream7 global interrupt	☑	0	0

11 Use UART with DMA transfer

- 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: UART_DMA

Project Location: D:\Radek\__Training_examples\F4_prague_2014\Labs\

Project Folder: D:\Radek\__Training_examples\F4_prague_2014\Labs\UART_DMA

Toolchain / IDE: EWARM 6.70

Mcu and Firmware Package

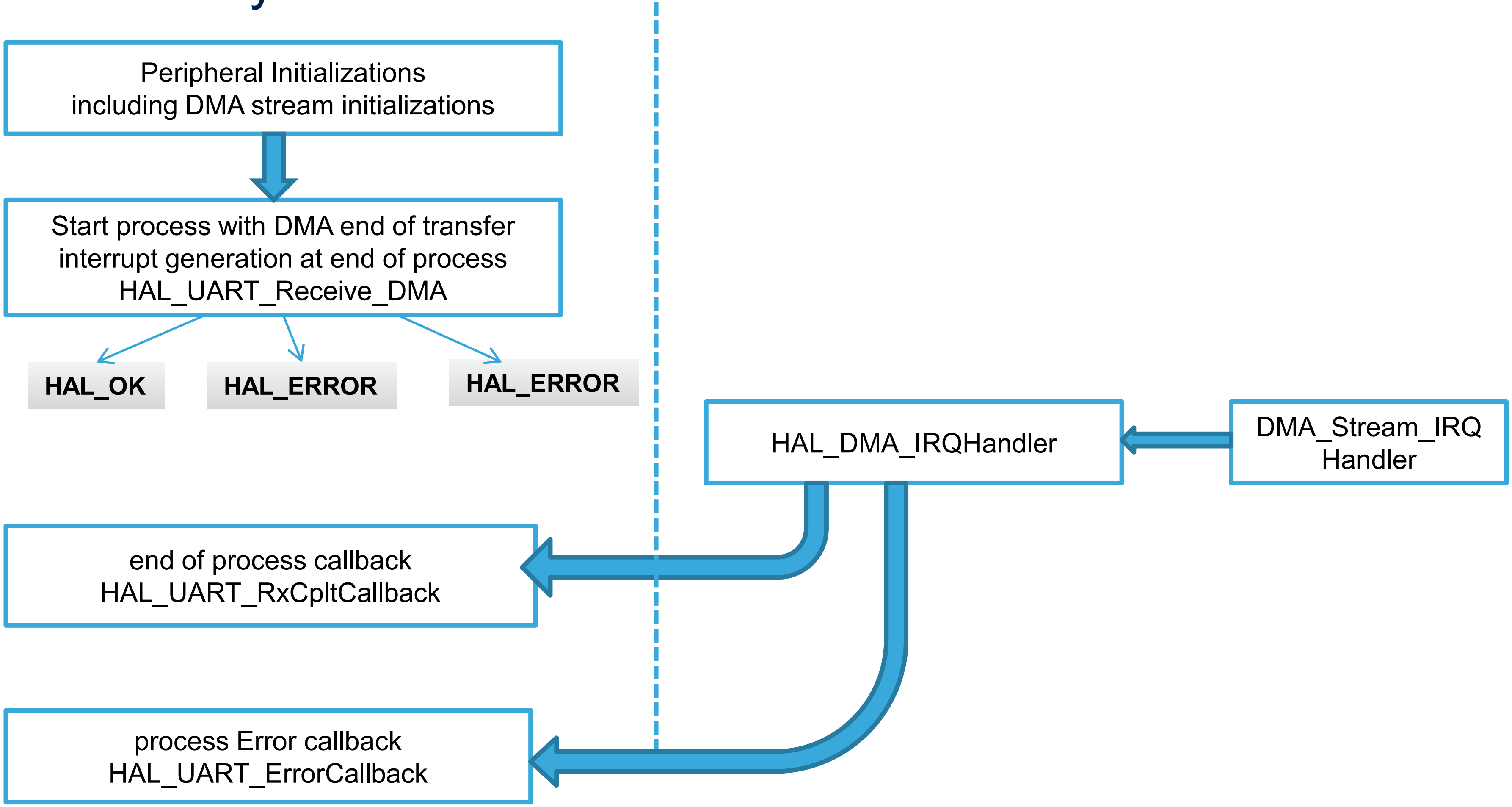
Mcu Reference: STM32F439ZITx

Firmware Package Name and Version: STM32Cube FW_F4 V1.3.0

☑ Use latest available version

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

HAL_DMA_IRQHandler

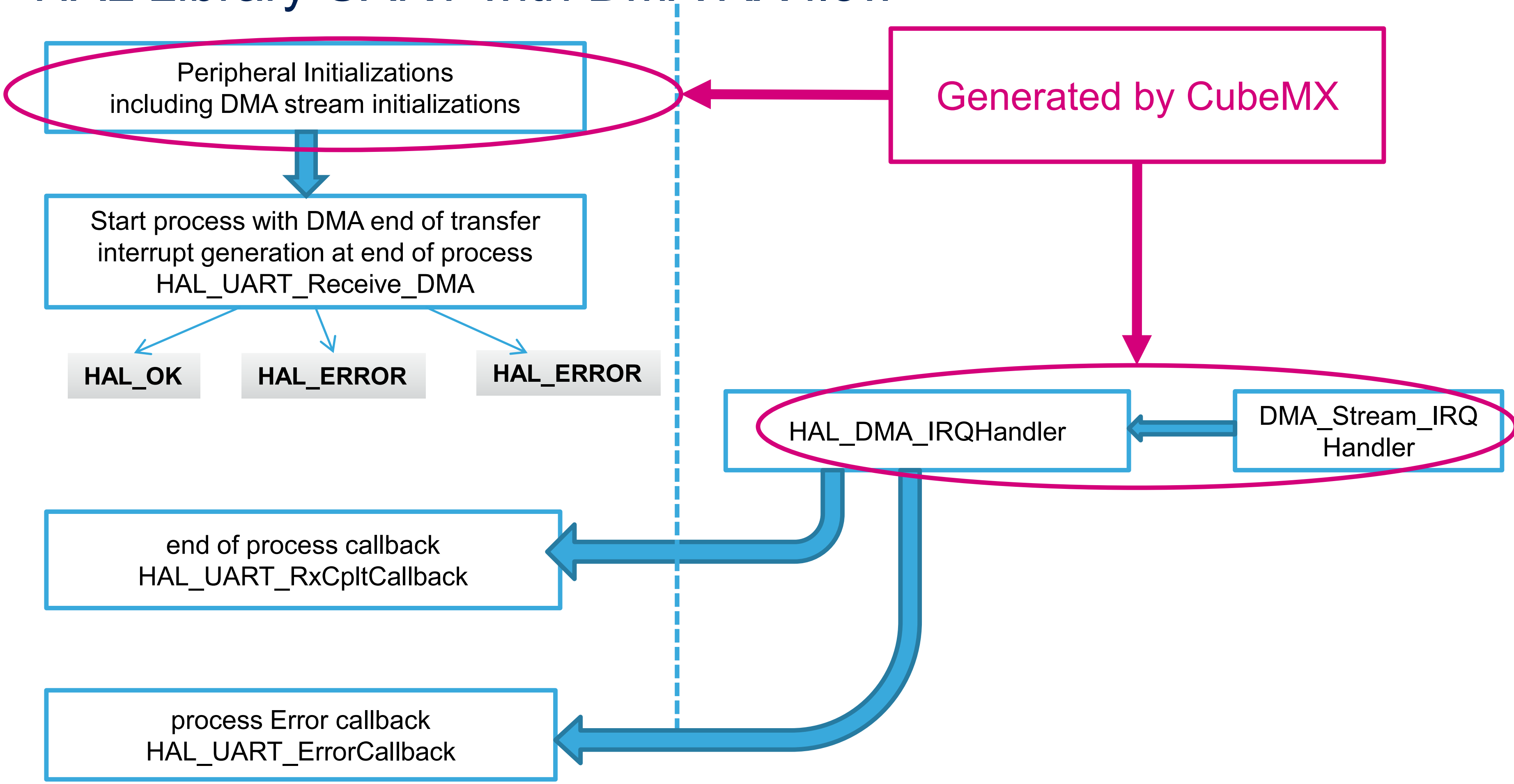
DMA_Stream_IRQ Handler

end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations
including DMA stream initializations

Generated by CubeMX

Start process with DMA end of transfer
interrupt generation at end of process
HAL_UART_Receive_DMA

HAL_DMA_IRQHandler

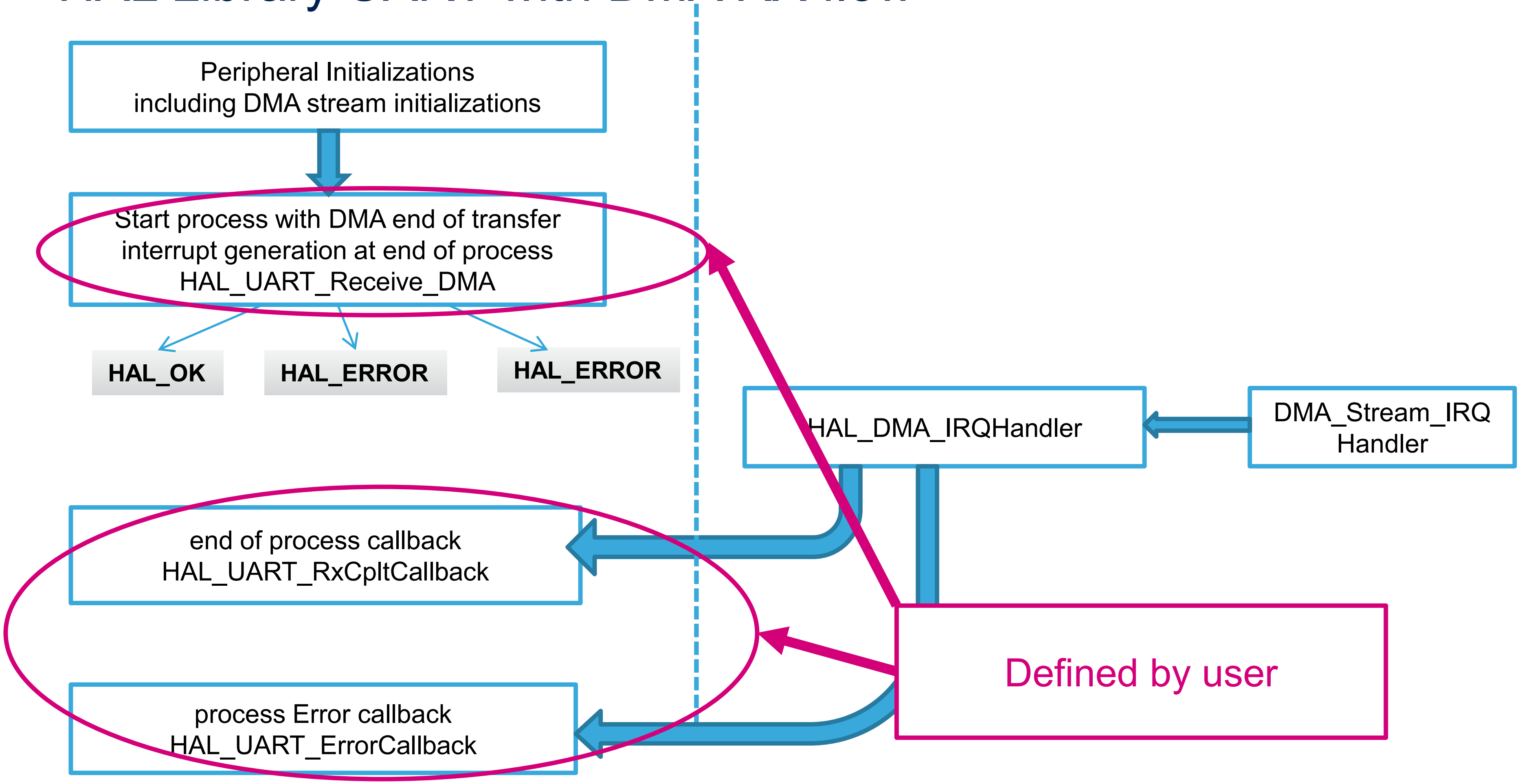
DMA_Stream_IRQ
Handler

end of process callback
HAL_UART_RxCpltCallback

process Error callback
HAL_UART_ErrorCallback

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

HAL_DMA_IRQHandler

DMA_Stream_IRQ Handler

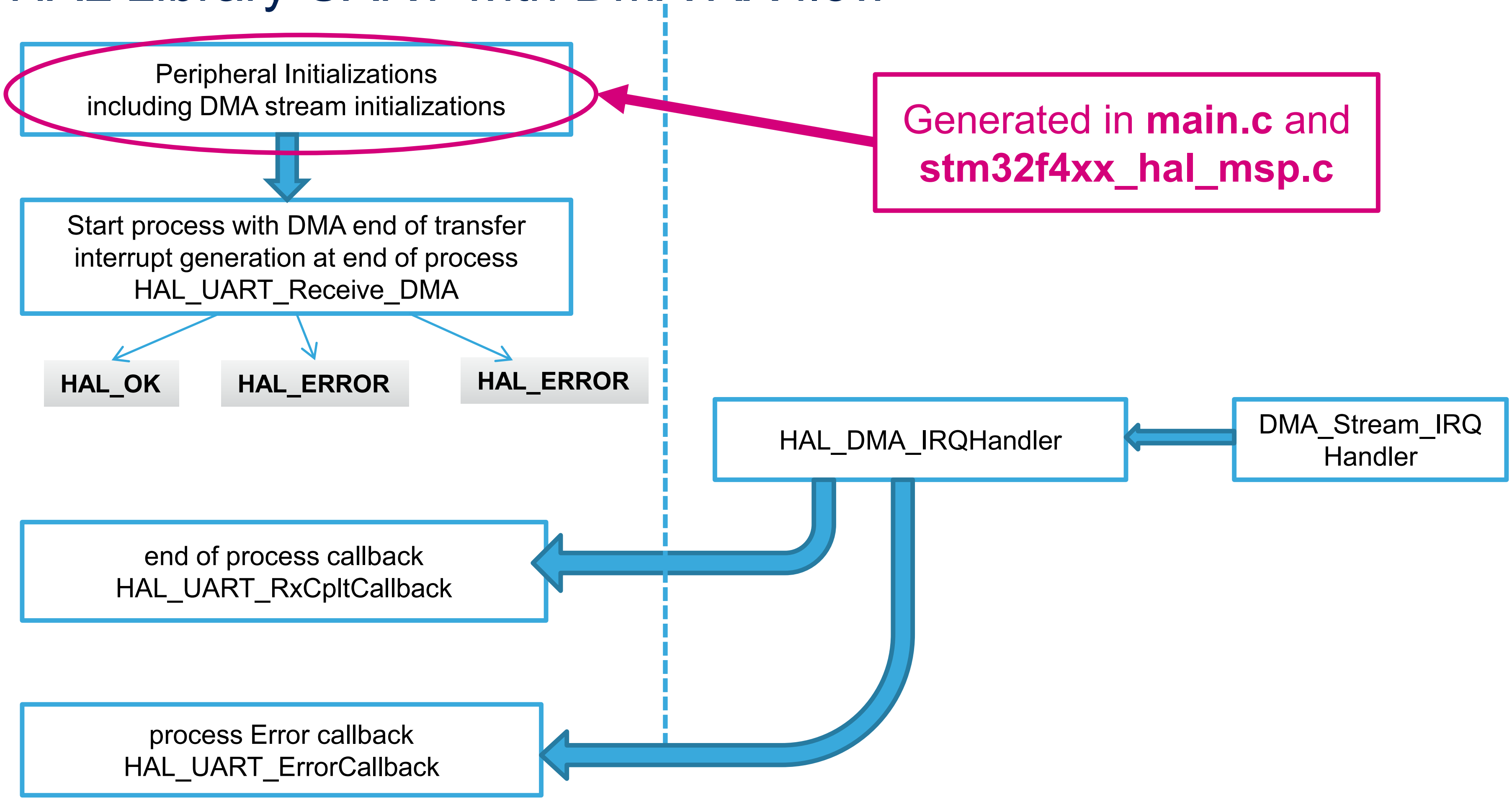
end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

Defined by user

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations
including DMA stream initializations

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

Start process with DMA end of transfer interrupt generation at end of process
HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

HAL_DMA_IRQHandler

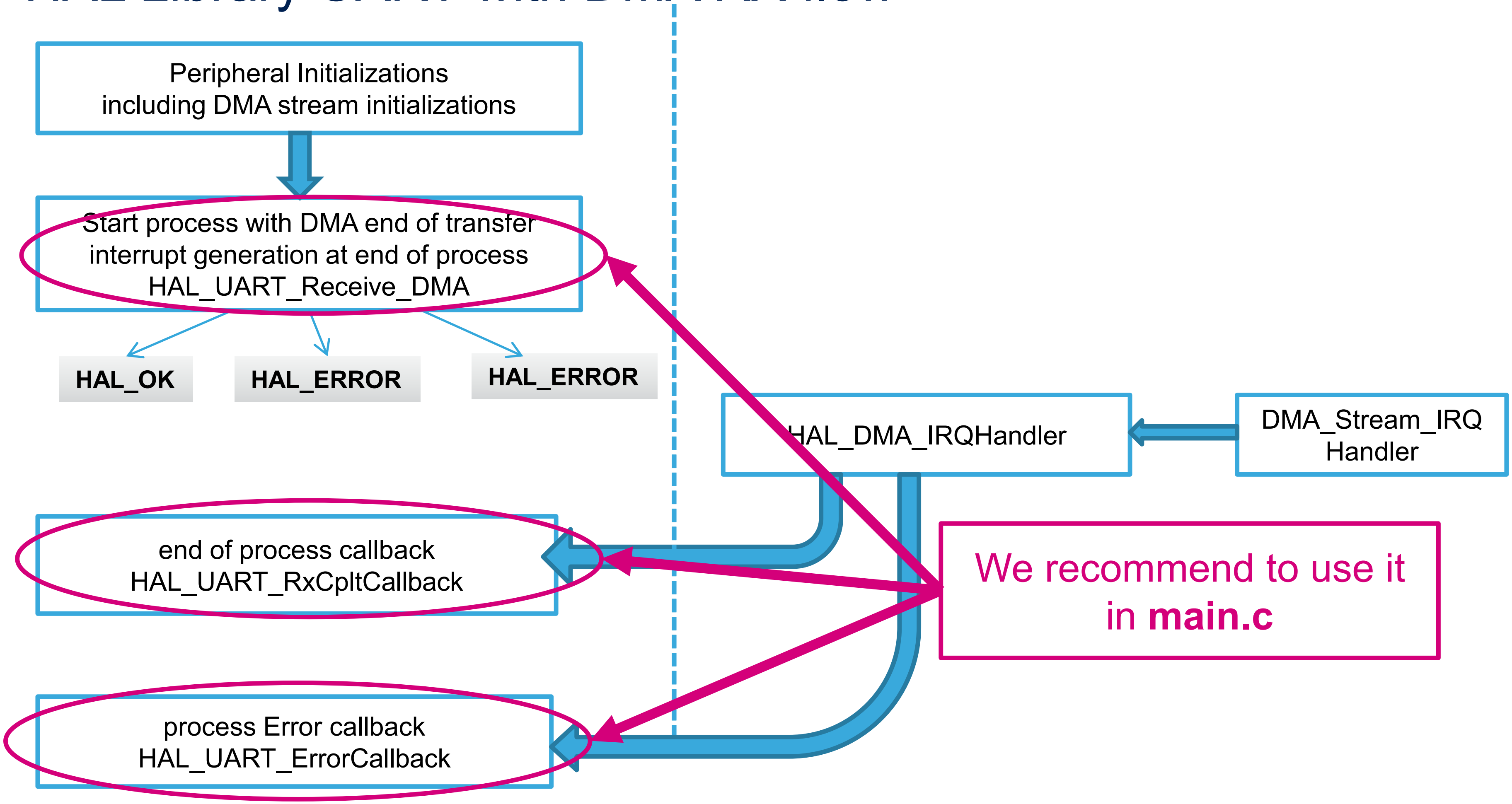
DMA_Stream_IRQ Handler

end of process callback
HAL_UART_RxCpltCallback

process Error callback
HAL_UART_ErrorCallback

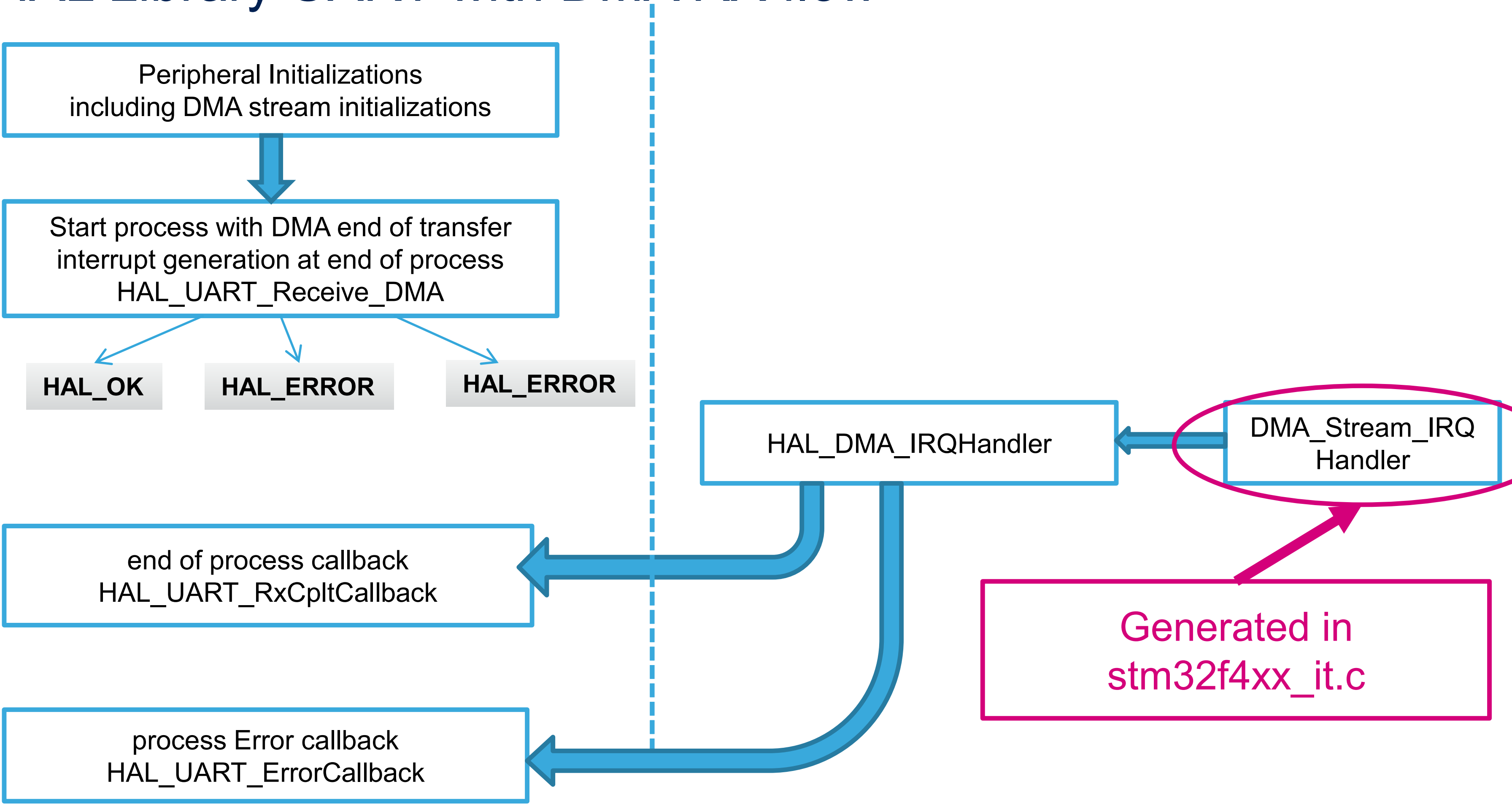
Use UART with DMA transfer

HAL Library UART with DMA RX flow


Peripheral Initializations including DMA stream initializations
Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA
HAL_OK
HAL_ERROR
HAL_ERROR
HAL_DMA_IRQHandler
DMA_Stream_IRQ Handler
end of process callback HAL_UART_RxCpltCallback
process Error callback HAL_UART_ErrorCallback

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

HAL Library UART with DMA RX flow



Peripheral Initializations
including DMA stream initializations

Start process with DMA end of transfer
interrupt generation at end of process
HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

HAL_DMA_IRQHandler

DMA_Stream_IRQ
Handler

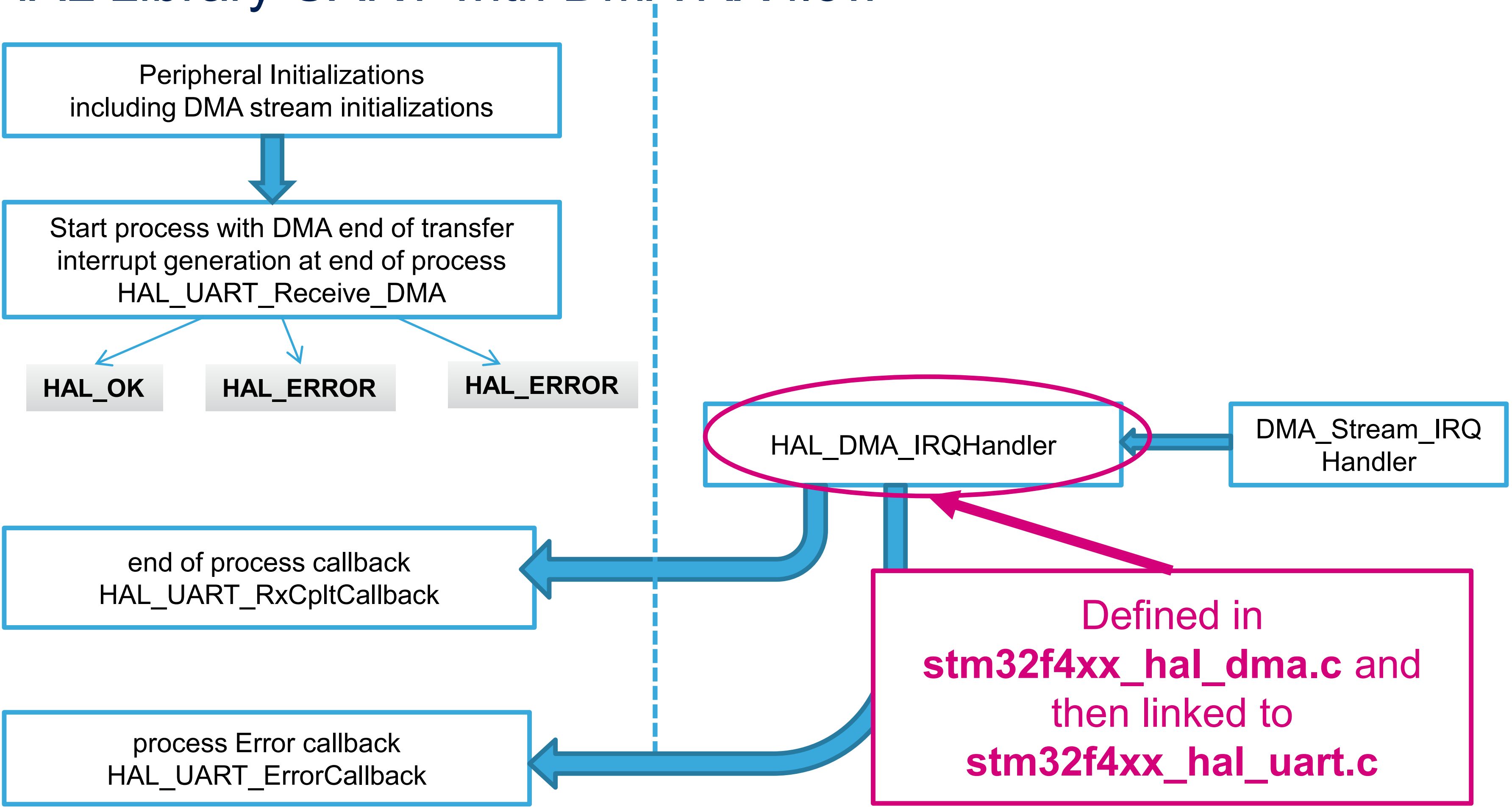
end of process callback
HAL_UART_RxCpltCallback

process Error callback
HAL_UART_ErrorCallback

Generated in
stm32f4xx_it.c

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

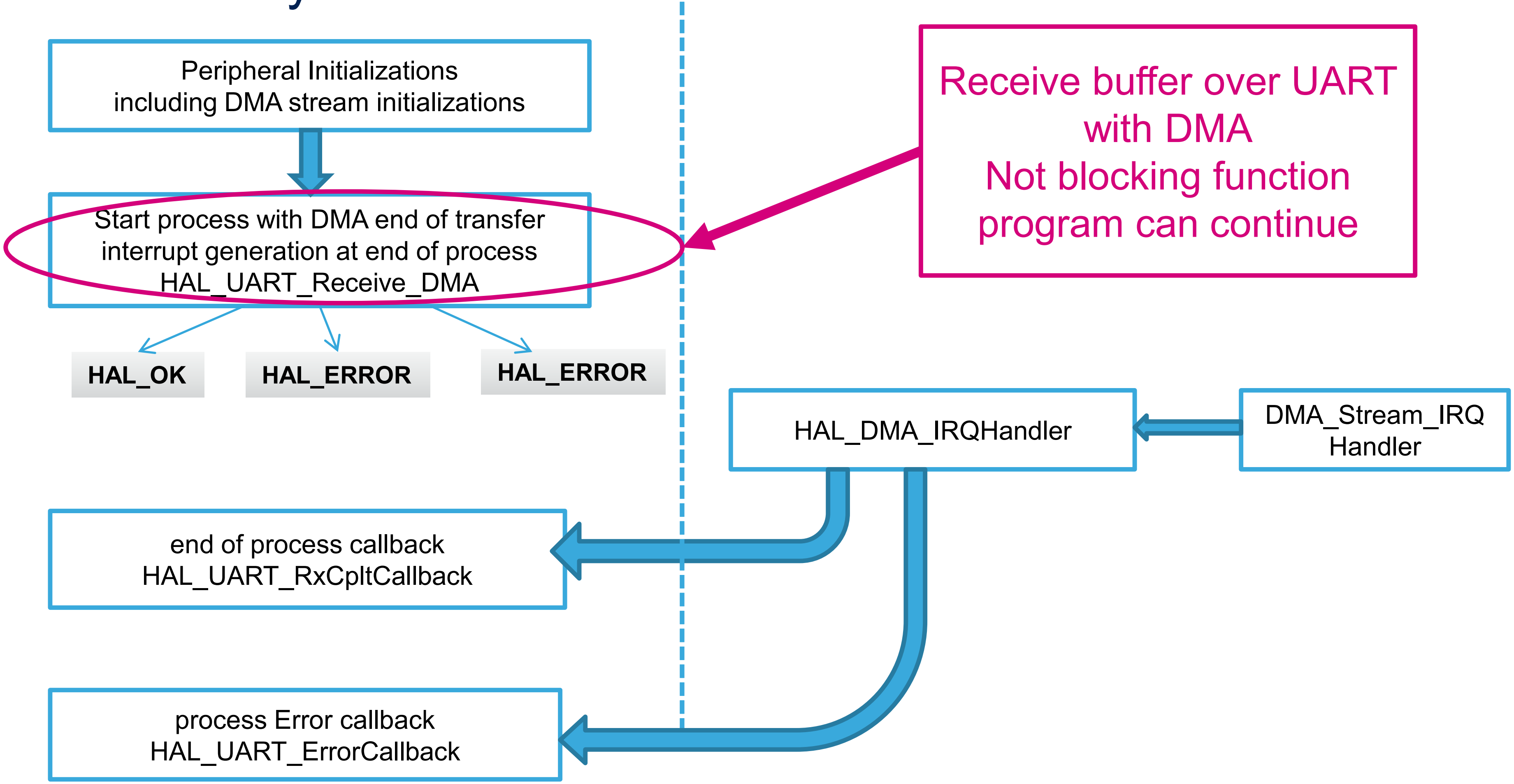
end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

Defined in **stm32f4xx_hal_dma.c** and then linked to **stm32f4xx_hal_uart.c**

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations
including DMA stream initializations

Start process with DMA end of transfer
interrupt generation at end of process
HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

Receive buffer over UART
with DMA
Not blocking function
program can continue

HAL_DMA_IRQHandler

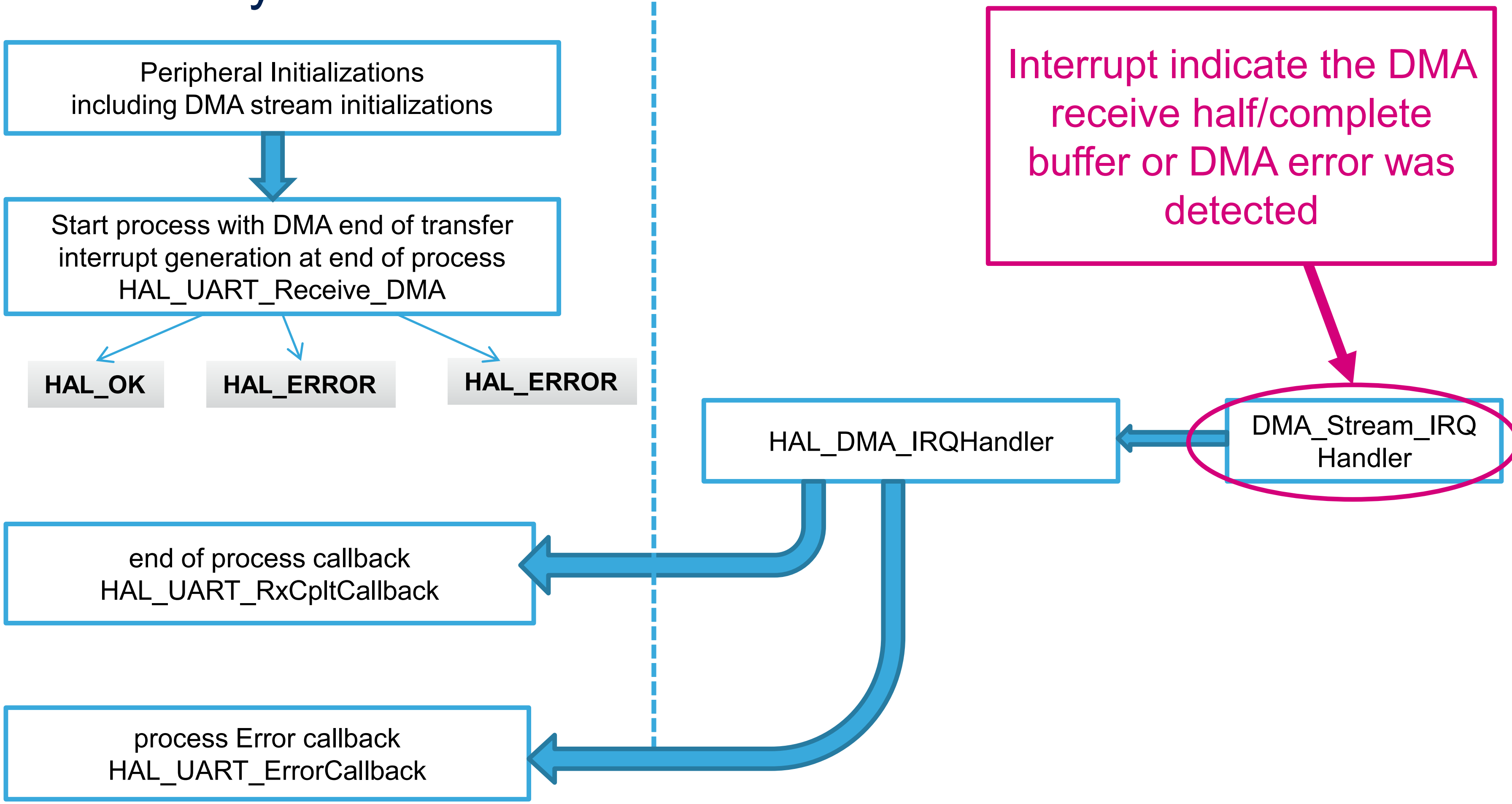
DMA_Stream_IRQ
Handler

end of process callback
HAL_UART_RxCpltCallback

process Error callback
HAL_UART_ErrorCallback

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

Interrupt indicate the DMA receive half/complete buffer or DMA error was detected

HAL_DMA_IRQHandler

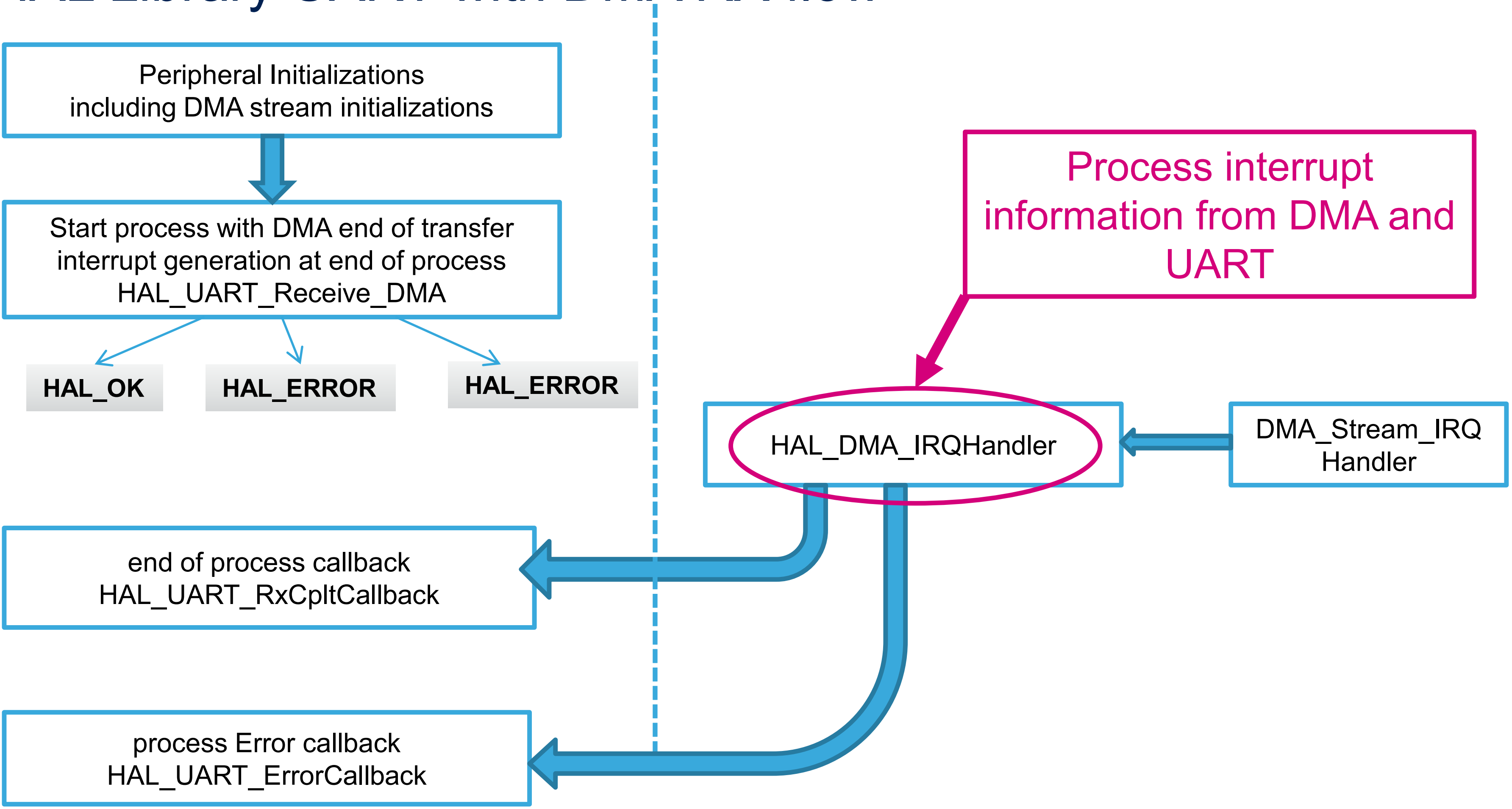
DMA_Stream_IRQ Handler

end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

Process interrupt information from DMA and UART

HAL_DMA_IRQHandler

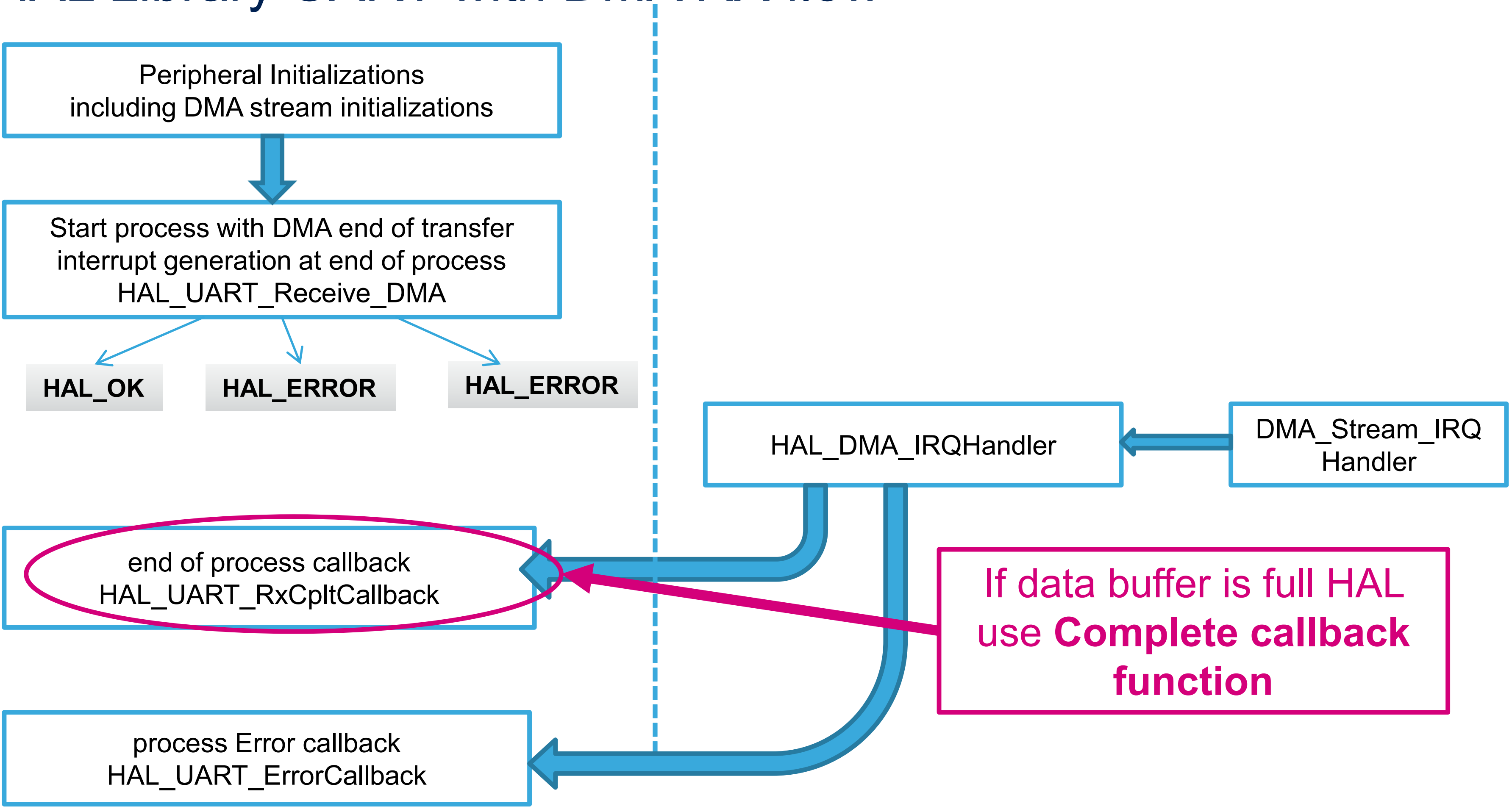
DMA_Stream_IRQ Handler

end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

HAL_DMA_IRQHandler

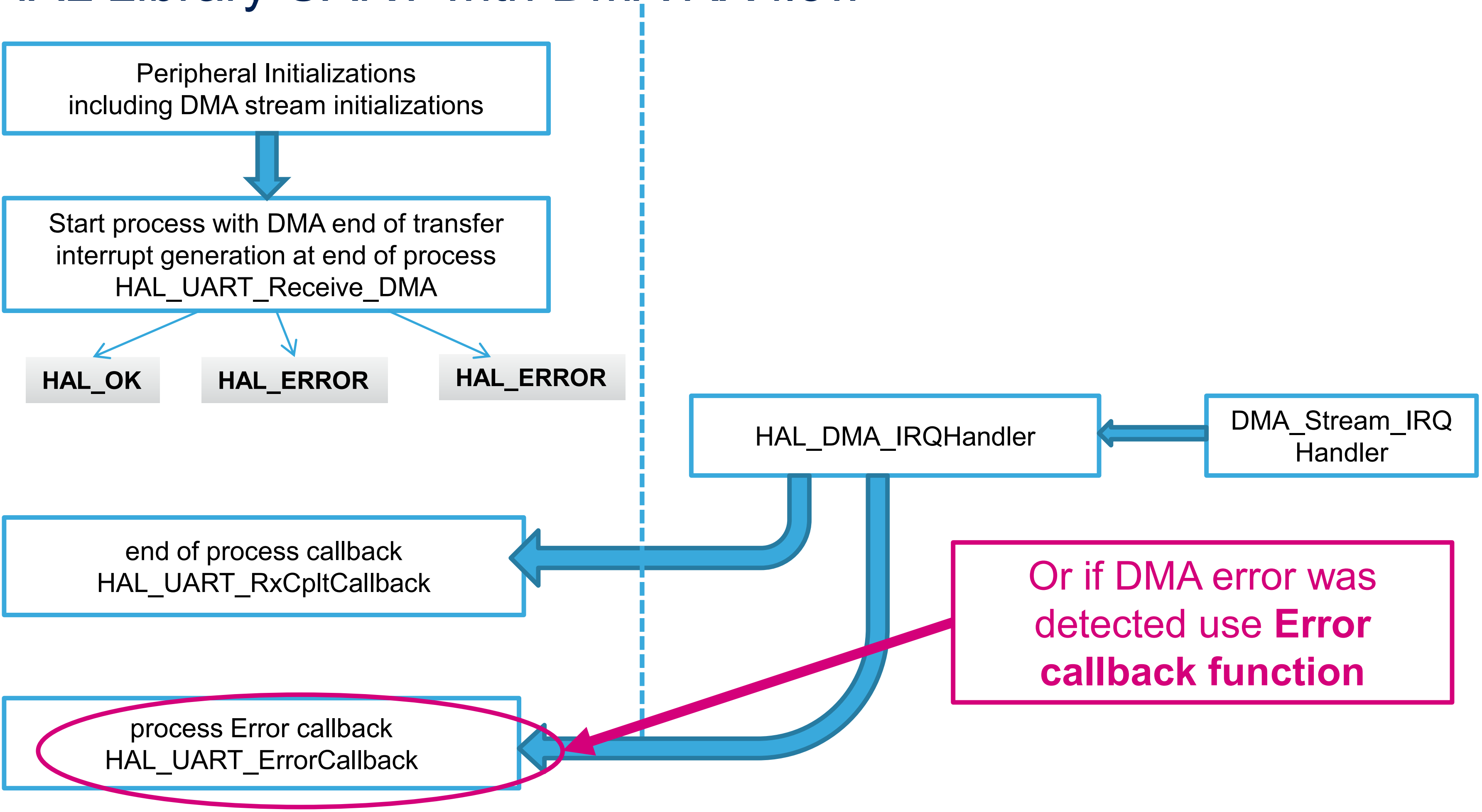
DMA_Stream_IRQ Handler

end of process callback HAL_UART_RxCpltCallback

process Error callback HAL_UART_ErrorCallback

If data buffer is full HAL use **Complete callback function**

HAL Library UART with DMA RX flow



Peripheral Initializations
including DMA stream initializations

Start process with DMA end of transfer
interrupt generation at end of process
HAL_UART_Receive_DMA

HAL_OK

HAL_ERROR

HAL_ERROR

HAL_DMA_IRQHandler

DMA_Stream_IRQ
Handler

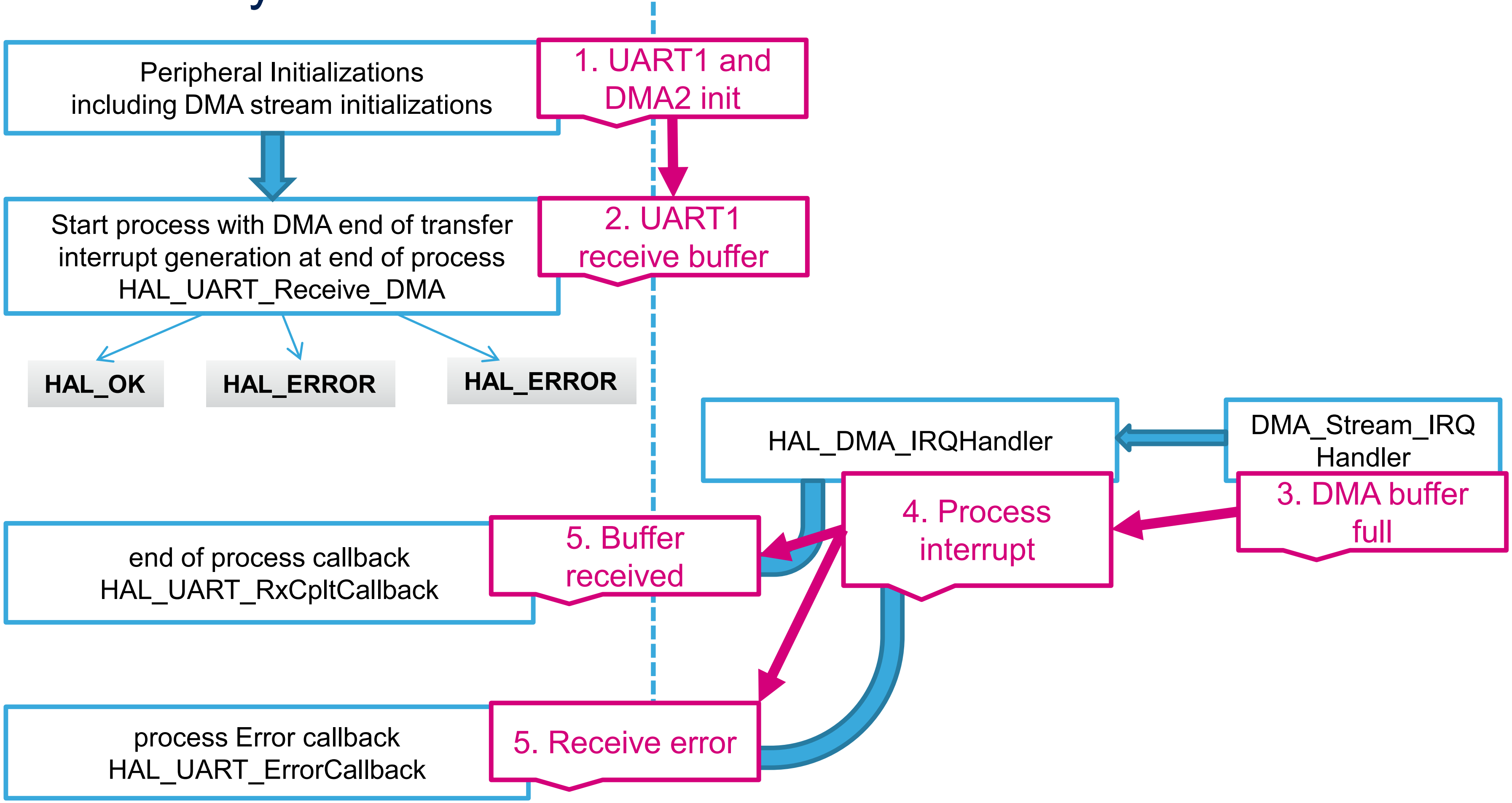
end of process callback
HAL_UART_RxCpltCallback

process Error callback
HAL_UART_ErrorCallback

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

Use UART with DMA transfer

HAL Library UART with DMA RX flow



Peripheral Initializations including DMA stream initializations

1. UART1 and DMA2 init

Start process with DMA end of transfer interrupt generation at end of process HAL_UART_Receive_DMA

2. UART1 receive buffer

HAL_DMA_IRQHandler

DMA_Stream_IRQ Handler

3. DMA buffer full

4. Process interrupt

end of process callback HAL_UART_RxCpltCallback

5. Buffer received

process Error callback HAL_UART_ErrorCallback

5. Receive error

11 Use UART with DMA transfer

- 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_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
- For receive use function
  - HAL_UART_Receive_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);

Use UART with DMA transfer

- Buffer definition

```
/* USER CODE BEGIN 0 */
uint8_t tx_buff[]={0,1,2,3,4,5,6,7,8,9};
uint8_t rx_buff[10];
/* USER CODE END 0 */
```

- Sending and receiving methods with DMA

```
/* USER CODE BEGIN 2 */
 HAL_UART_Receive_DMA(&huart1,rx_buff,10);
 HAL_UART_Transmit_DMA(&huart1,tx_buff,10);
 /* USER CODE END 2 */
```

11 Use UART with DMA transfer

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

```
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  __NOP();//check if we receive all data
}
/* USER CODE END 4 */
```