

Data transfer over DMA lab 6

- Objective

- Learn how to setup DMA transfer in CubeMX
- Create simple DMA memory to memory transfer from RAM to RAM

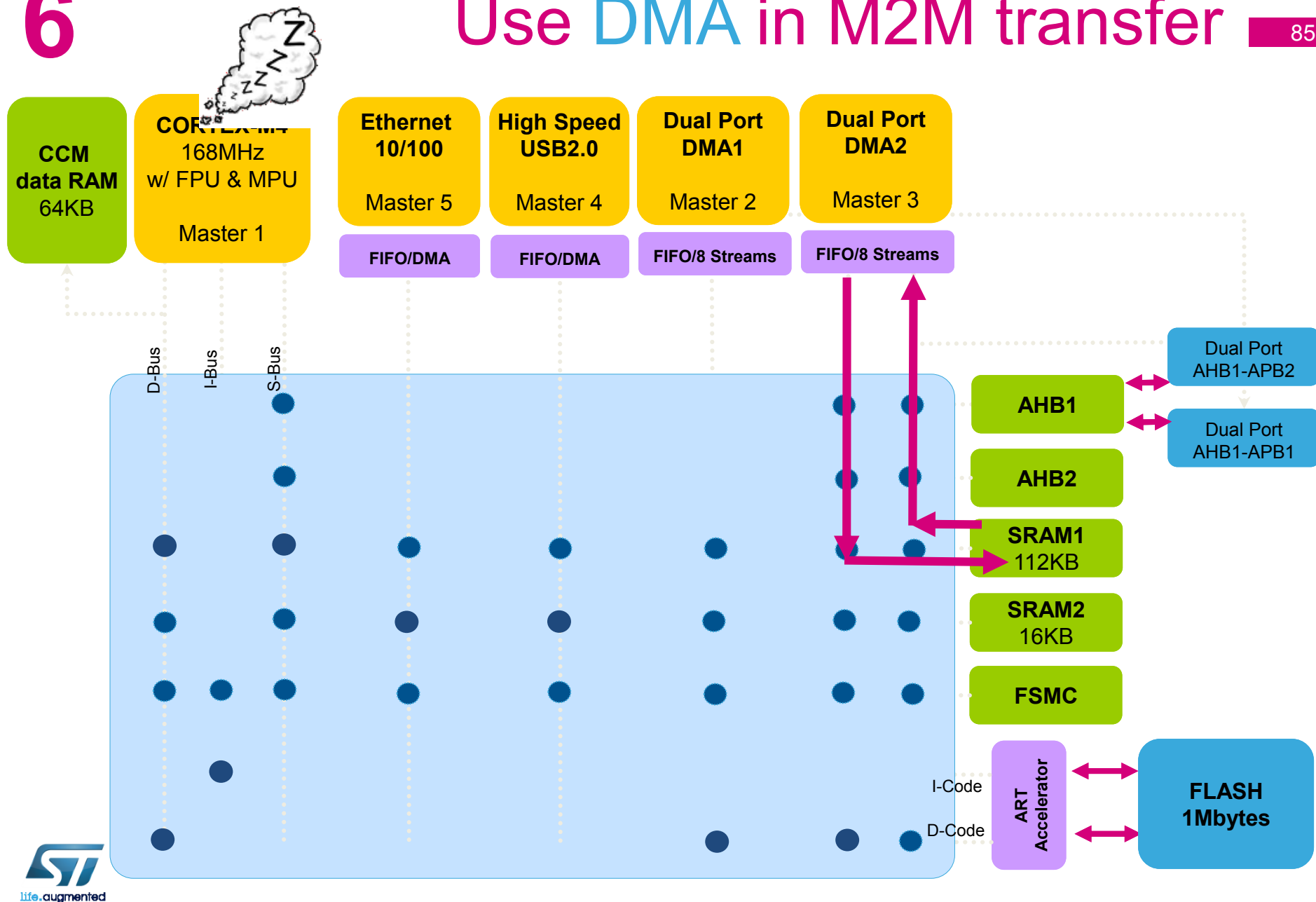
- Goal

- Use CubeMX and Generate Code with DMA
- Learn how to setup the DMA in HAL
- Verify the correct functionality by comparing transferred buffers

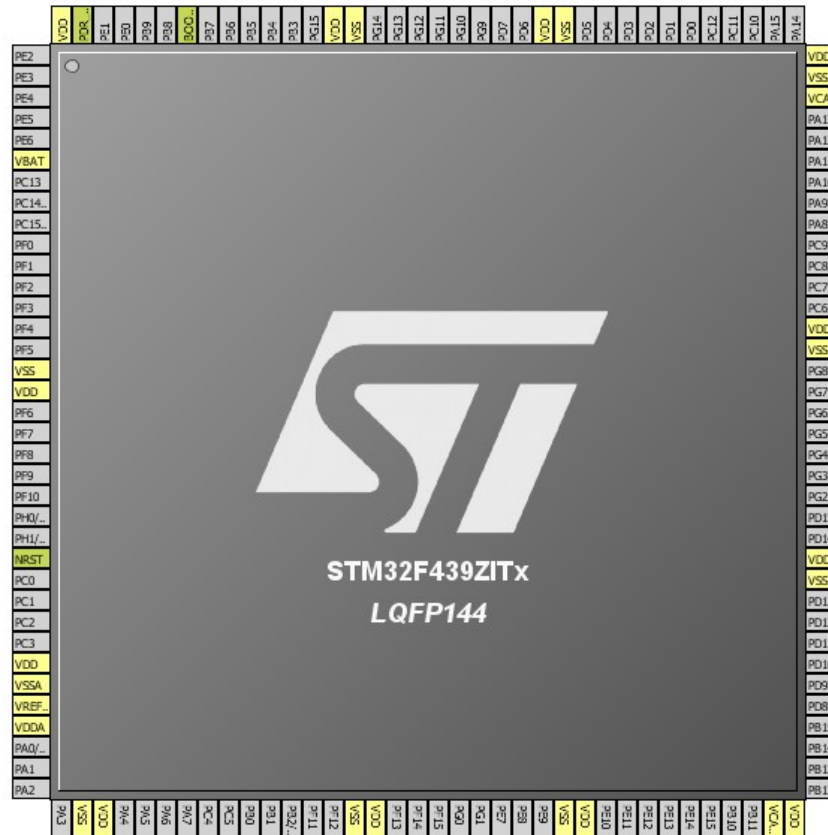
6

Use DMA in M2M transfer

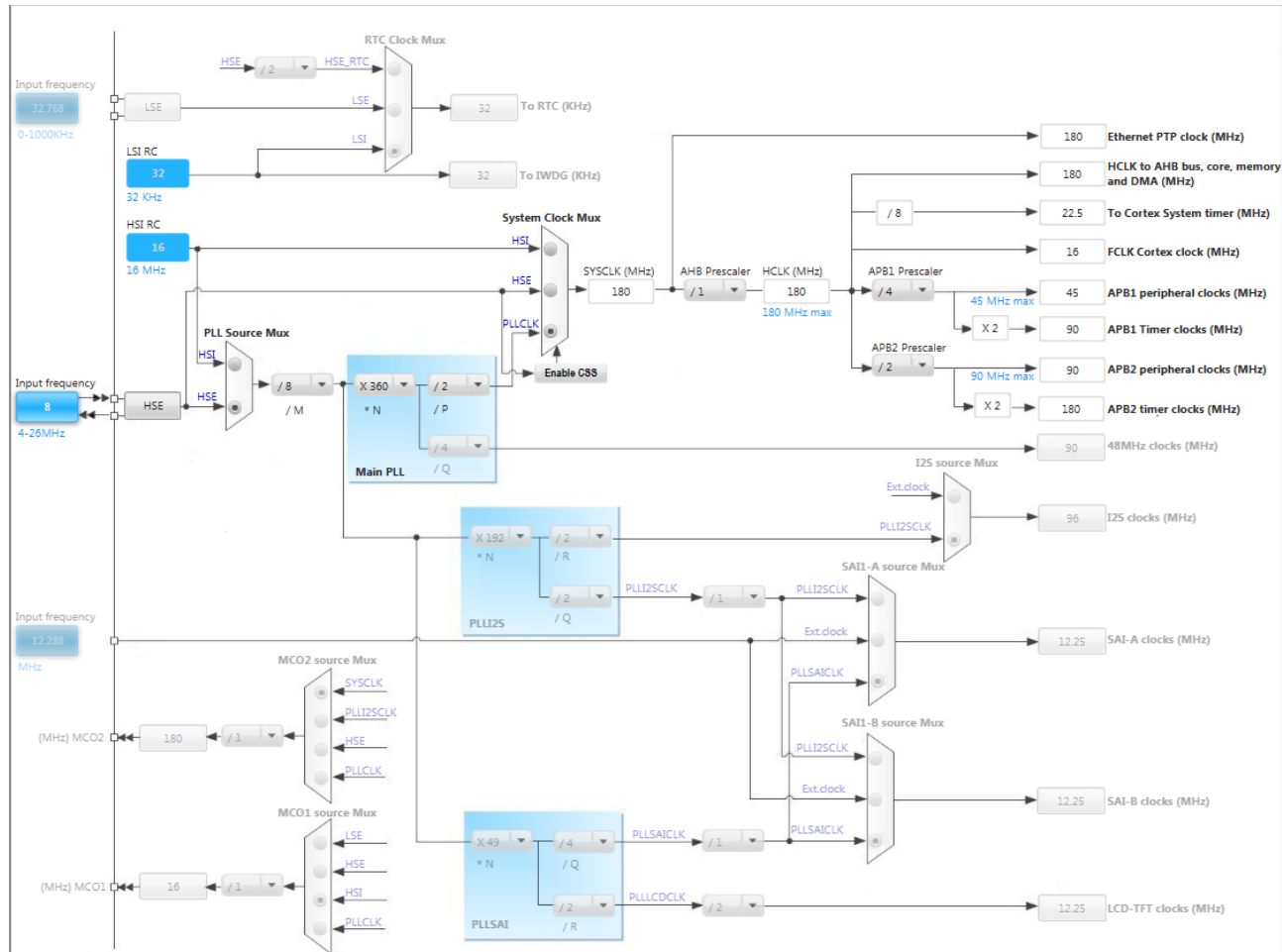
85



- Create project in CubeMX
 - Menu > File > New Project
 - Select STM32F4 > STM32F429/439 > LQFP144 > STM32F439ZITx
- For DMA we don't need to configure any pins



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



6

Use DMA in M2M transfer

88

- DMA configuration

- TAB>Configuration
- System>DMA
- TAB>DMA2
- Button ADD

1. TAB > Configuration

2. System DMA

3. TAB>DMA 2

4. Add DMA channel

The screenshot shows the STM32CubeMX software interface. The 'Configuration' tab is selected in the top menu. The 'DMA Configuration' dialog box is open, showing the 'DMA2' channel selected. The 'Add' button is highlighted. The 'System' view on the right shows the 'DMA' component selected. The 'DMA Request' table is empty. The 'DMA Request Settings' section shows 'Mode' set to 'Normal', 'Peripheral' set to 'Address', and 'Memory' set to 'Byte'.

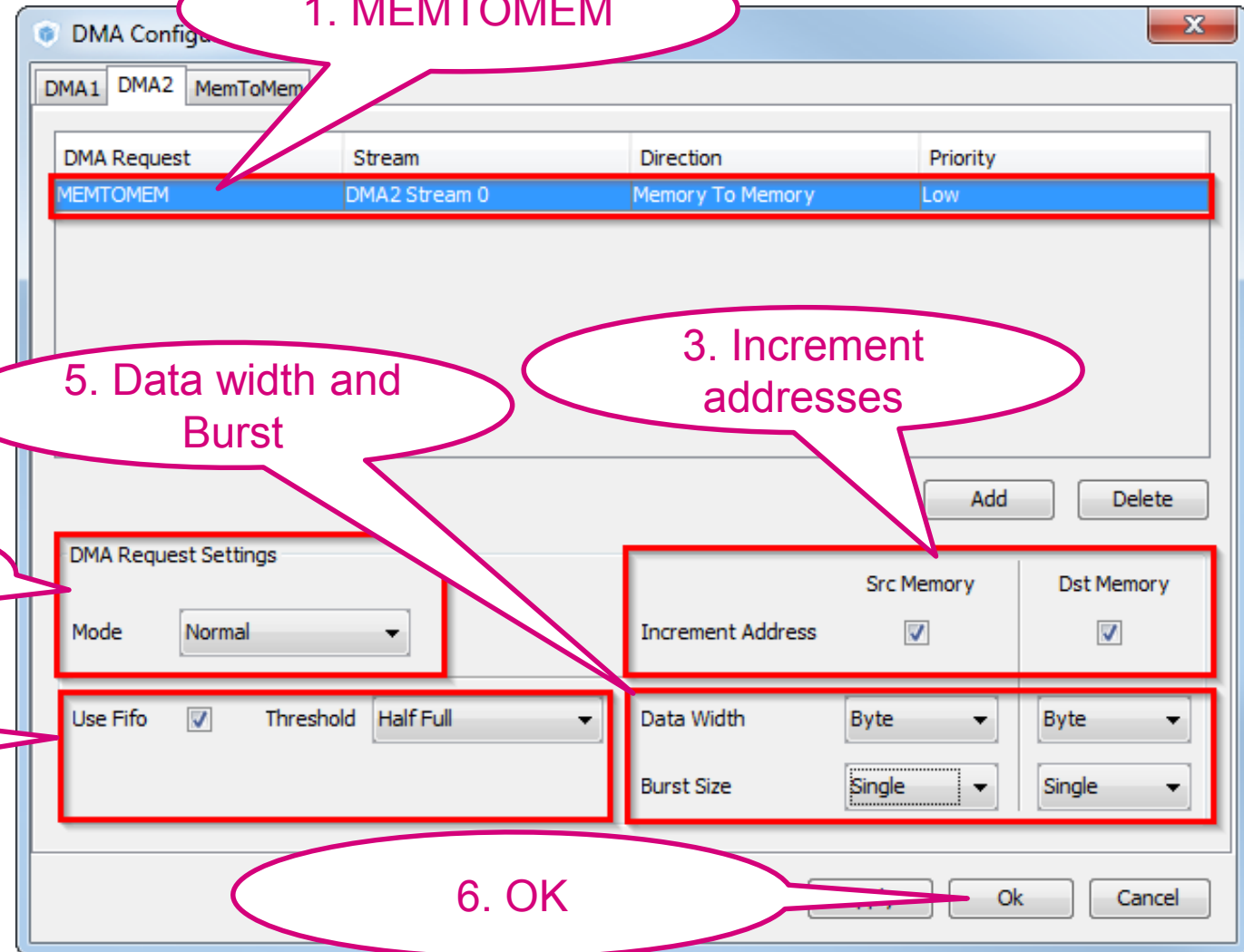
6

Use DMA in M2M transfer

89

- DMA configuration

- Select MEMTOMEM DMA request
- Normal mode
- Increment source and destination address
- FIFO setup
- Byte data width
- Burst size
- Button OK

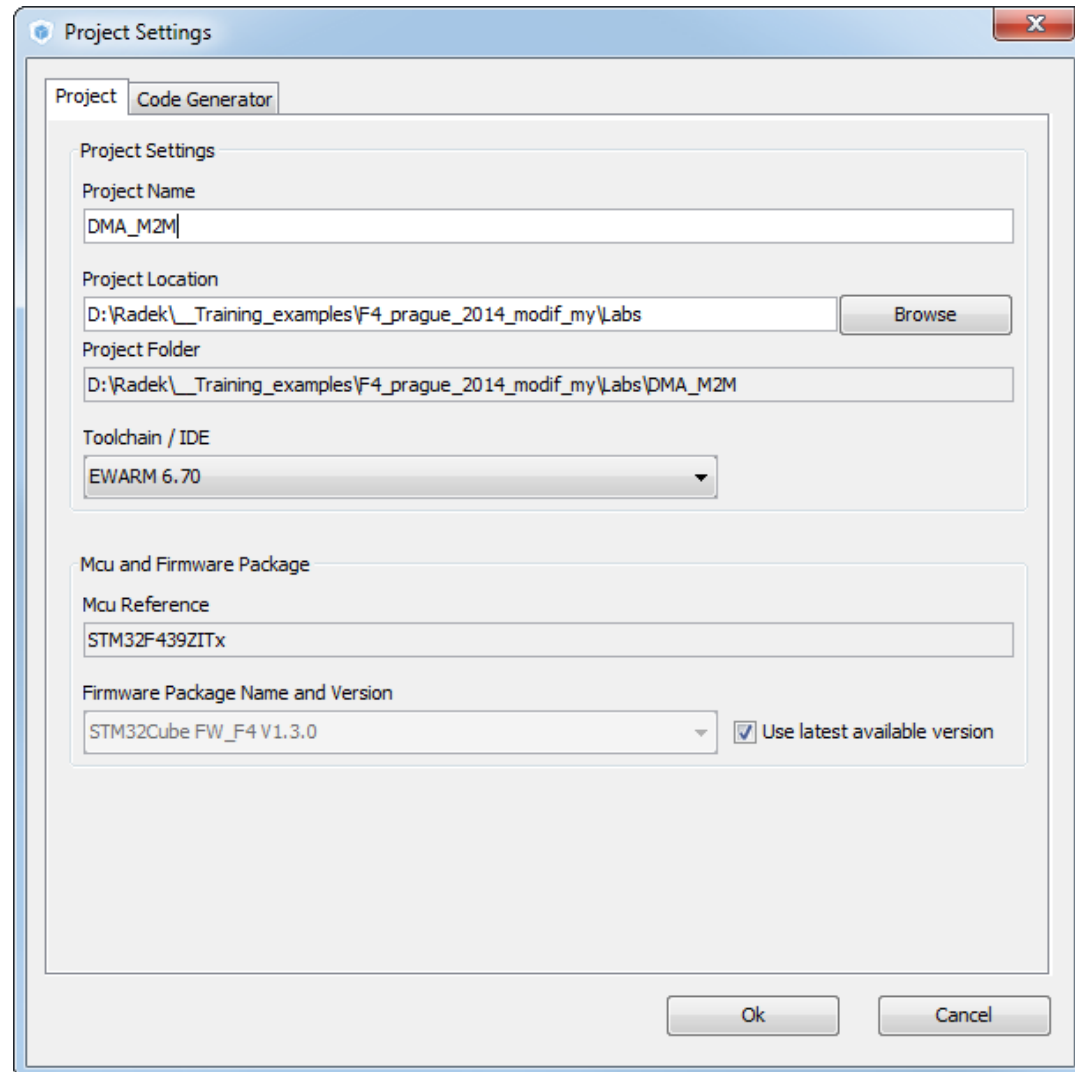


- Now we set the project details for generation

- Menu > Project > Project Settings
- Set the project name
- Project location
- Type of toolchain

- Now we can Generate Code

- Menu > Project > Generate Code

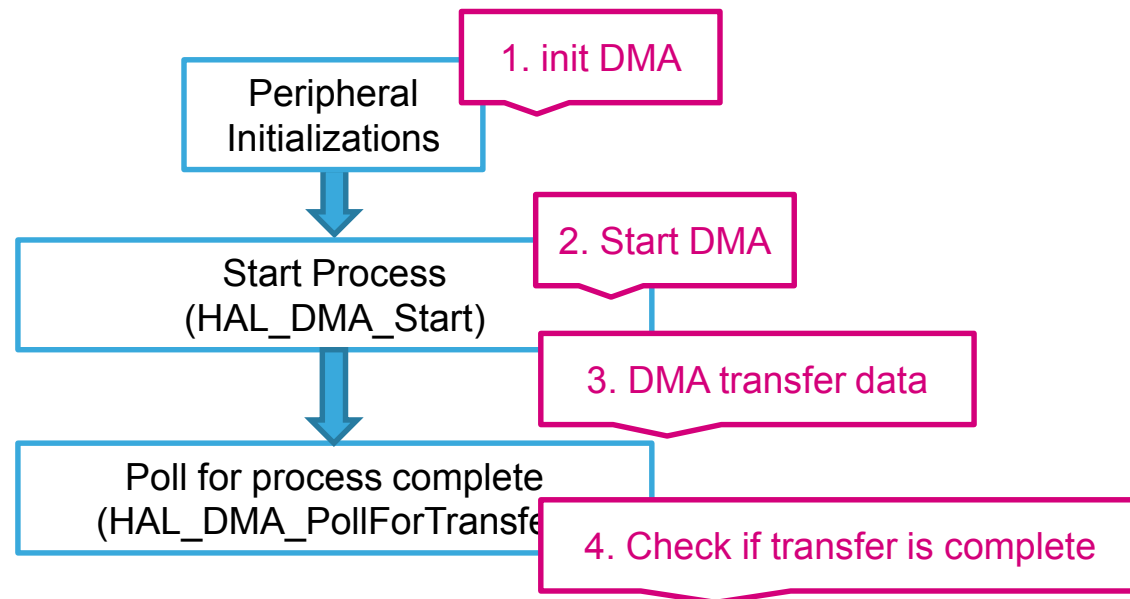


6

Use DMA in M2M transfer

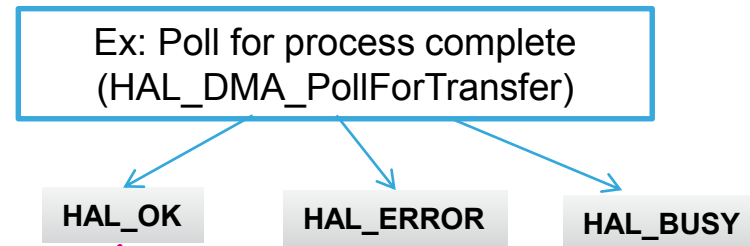
91

- Start process DMA (same for TIM, ADC)
 - Non blocking start process
 - The end of the process must be checked by polling



- Return values

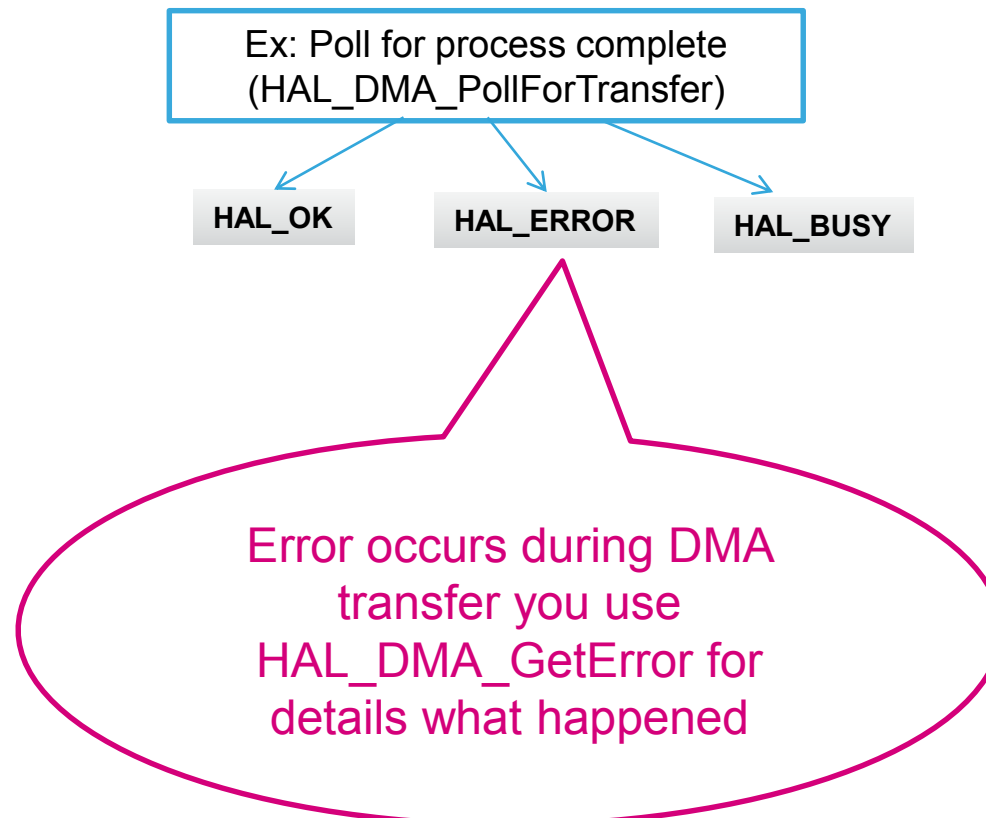
- Most of CubeMX functions have return values, which indicate, if operation was successful, timeout occurs or function ends with error
- Is recommended to handle these return values to be sure that the program is working as expected



DMA transfer was successfully finished and data was transferred to destination without error

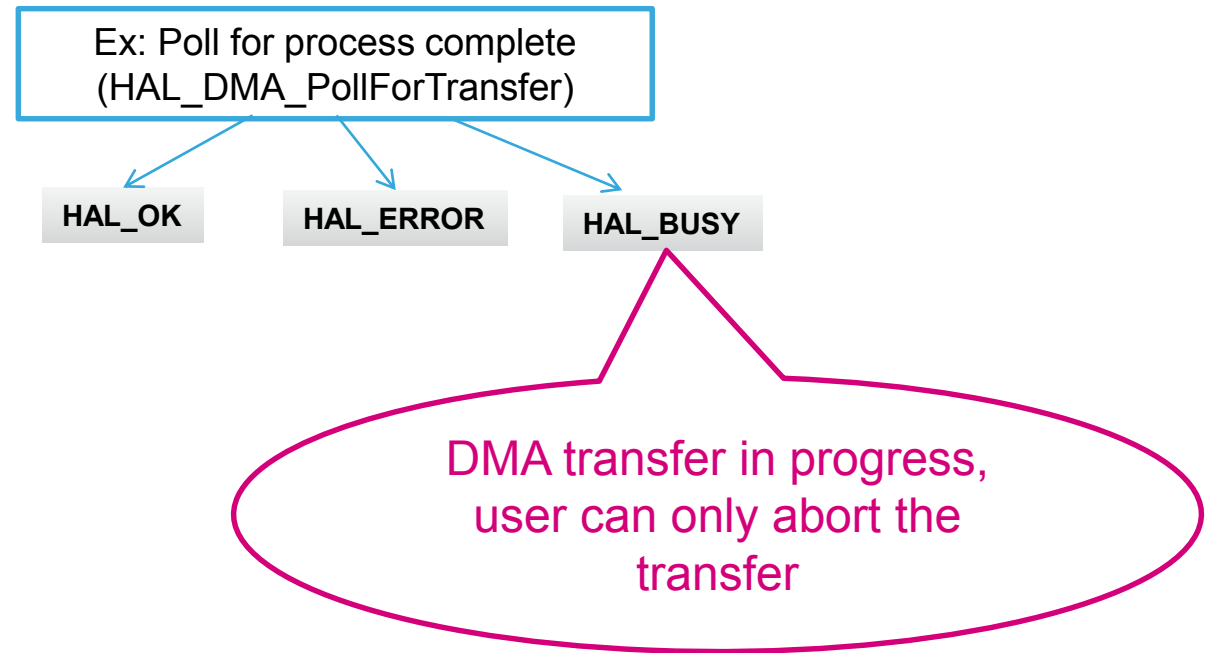
- Return values

- Most of CubeMX functions have return values, which indicate, if operation was successful, timeout occurs or function ends with error
- Is recommended to handle these return values to be sure that the program is working as expected



- Return values

- Most of CubeMX functions have return values, which indicate, if operation was successful, timeout occurs or function ends with error
- It is recommended to handle these return values to be sure that the program is working as expected



- Now we 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
- HAL functions for DMA
 - HAL_DMA_Start(DMA_HandleTypeDef *hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)
 - HAL_DMA_PollForTransfer(DMA_HandleTypeDef *hdma, uint32_t CompleteLevel, uint32_t Timeout)

- We create two buffers
 - One with source data
 - Second as destination buffer

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

6

Use DMA in M2M transfer

97

- HAL_DMA_Start start the M2M data transfer
- HAL_DMA_PollForTransfer check if the transfer ends successfully

```
/* USER CODE BEGIN 2 */
HAL_DMA_Start(&hdma_memtomem_dma2_stream0, (uint32_t) (Buffer_Src), (uint32_t) (Buffer_Dest), 10);
while(HAL_DMA_PollForTransfer(&hdma_memtomem_dma2_stream0, HAL_DMA_FULL_TRANSFER, 100) != HAL_OK)
{
    __NOP();
}
/* USER CODE END 2 */
```