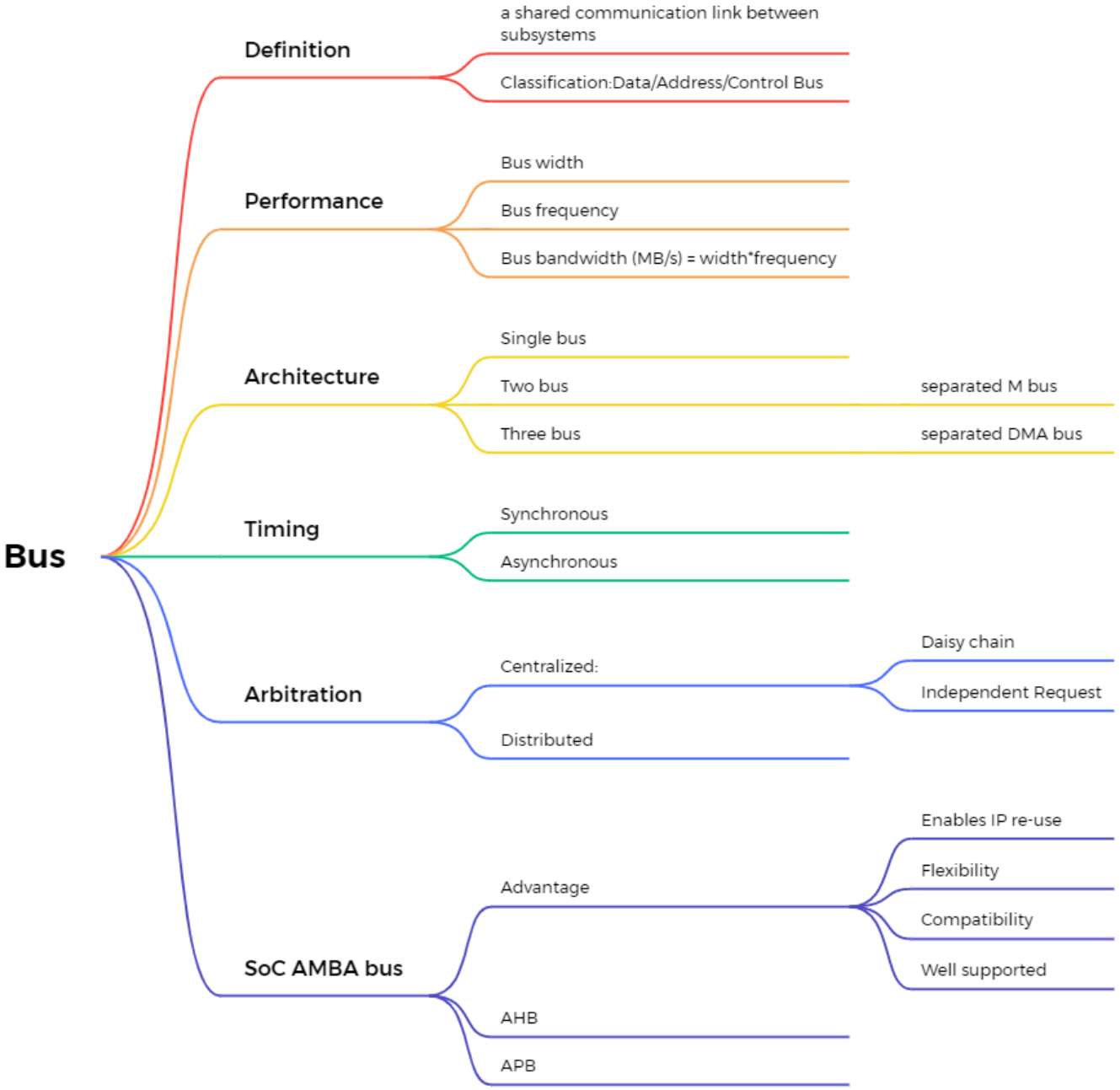
# CS301
# Embedded System and Microcomputer Principle

## Lecture 11: DMA

2023 Fall

# Recap


Bus
Definition
a shared communication link between subsystems
Classification:Data/Address/Control Bus
Performance
Bus width
Bus frequency
Bus bandwidth (MB/s) = width*frequency
Architecture
Single bus
Two bus
separated M bus
Three bus
separated DMA bus
Timing
Synchronous
Asynchronous
Arbitration
Centralized:
Daisy chain
Independent Request
Distributed
SoC AMBA bus
Advantage
Enables IP re-use
Flexibility
Compatibility
Well supported
AHB
APB

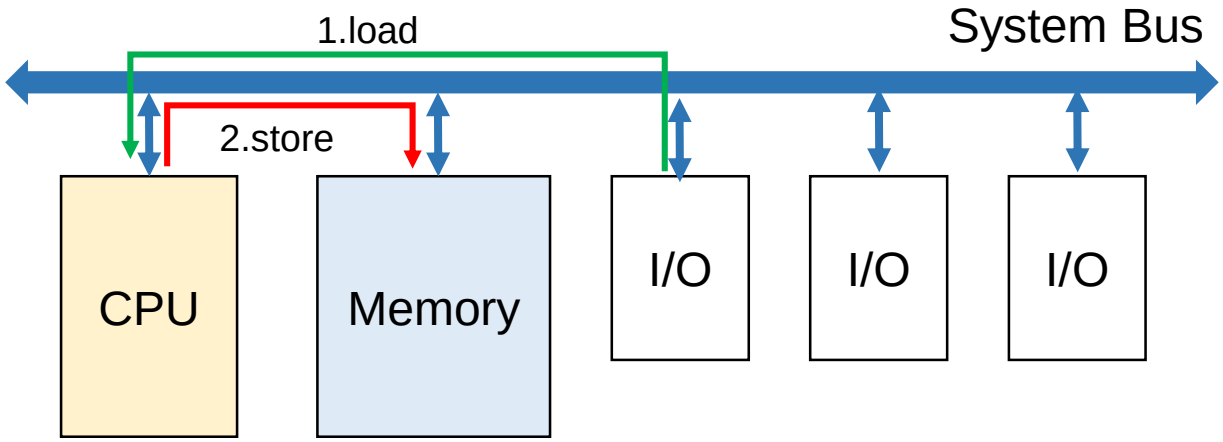
# Outline

- **Why DMA**
- DMA Architecture
- STM32 DMA Controller

# Data Transfer Schemes

- In a computer, the data transfer happens between any of these combinations
  - CPU and memory
  - CPU and I/O devices
  - I/O devices and memory
- How to move data between I/O and Memory?
  1. Load from peripheral (memory mapped IO)
  2. Store into memory

```
LDA r0, [r1]
STA r0, [r2]
```

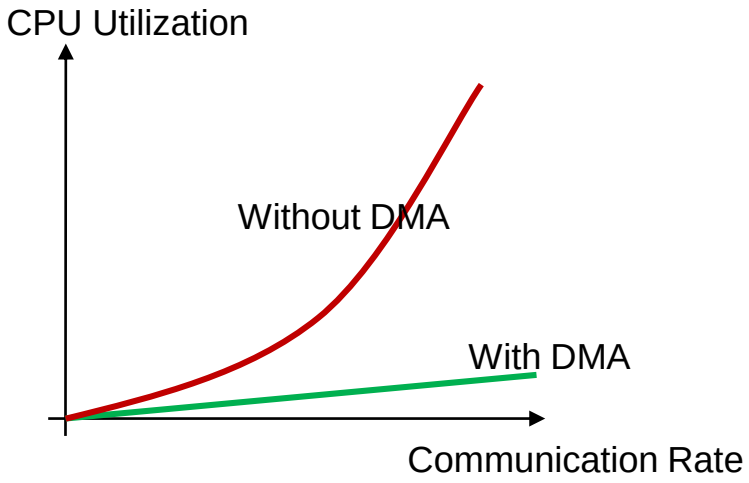


# Data Transfer Schemes

- However, I/O devices may not be ready to transfer data as soon as the microprocessor issues the instruction for this purpose.
- Common ways to **transfer data between I/O and memory** are:
  - Polling
  - Interrupt
  - Direct Memory Access (DMA)

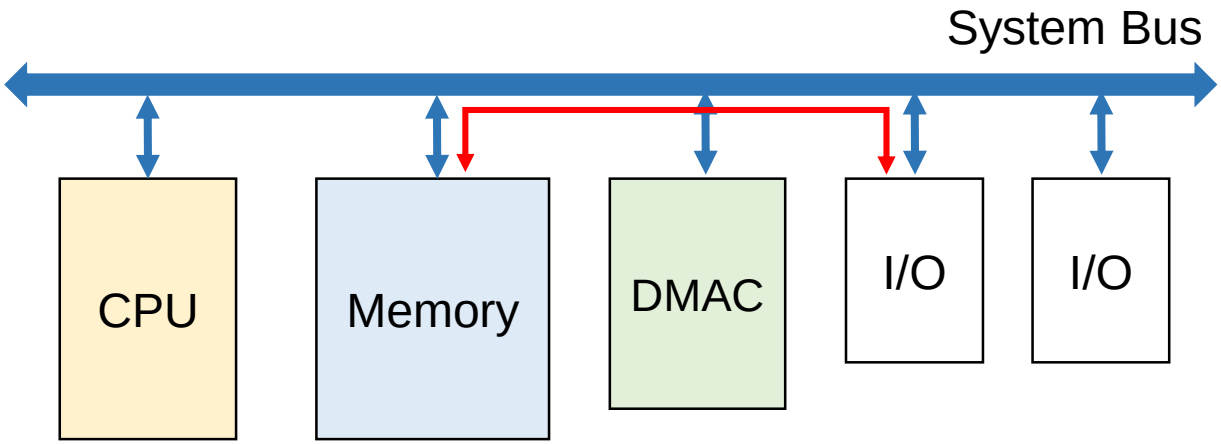
# Why Need DMA?

- If transferring data without the use of DMA
  - Interrupt: interrupt is generated when a new data is available and CPU has to transfer it
  - Polling: the CPU waits for a new data to become available and then transfers it
- In both case, each pure data transfer requires 2 cycles (load + store)
  - waste of CPU instructions: Instructions are needed to increment memory address and keeping track of how many bytes are moved.



# Direct Memory Access (DMA)

- To transfer large blocks of data at high speed, an alternative approach is used, called DMA.
- Blocks of data are transferred between an external device and the main memory, or between memory to memory, or external device to another external device, without continuous intervention by the processor.
- DMAC is the control unit for DMA transfer, it is just another peripheral whose job is moving data.

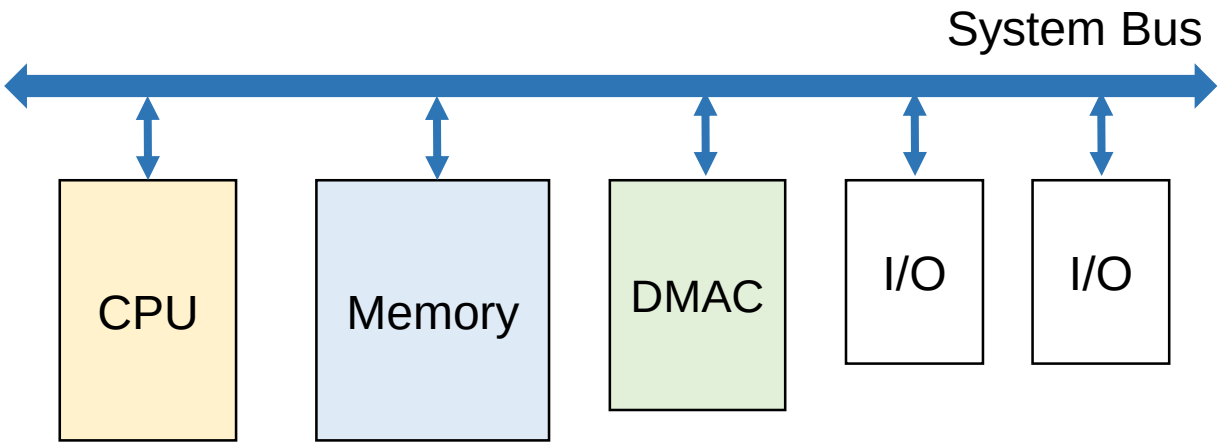


# Outline

- Why DMA
- **DMA Architecture**
- STM32 DMA Controller

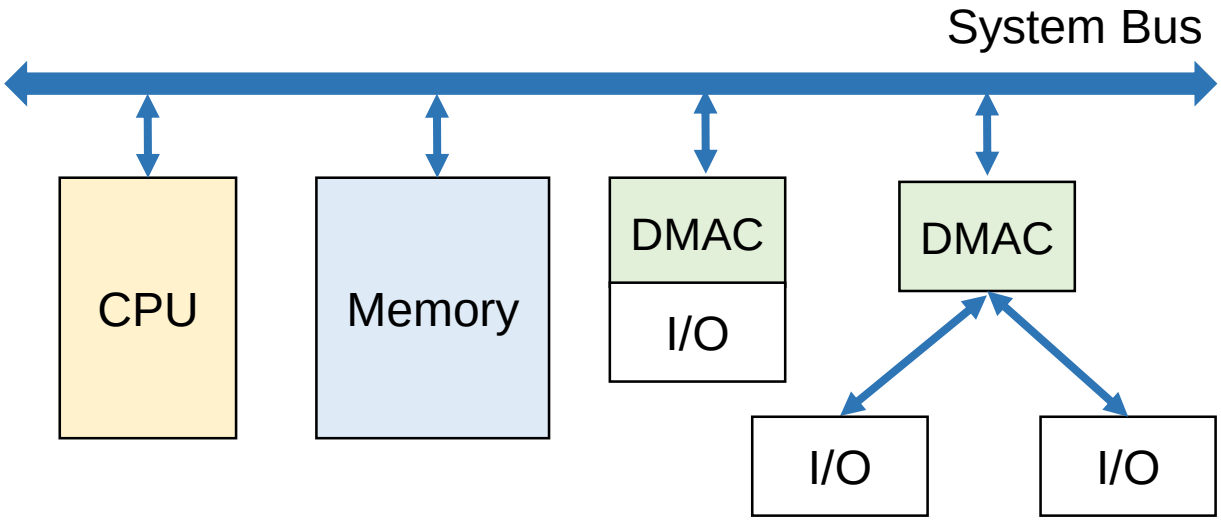
# DMA Configurations

- Single Bus, Detached DMA controller
  - Each transfer uses bus twice
    - I/O to DMAC then DMAC to memory
  - CPU is suspended twice



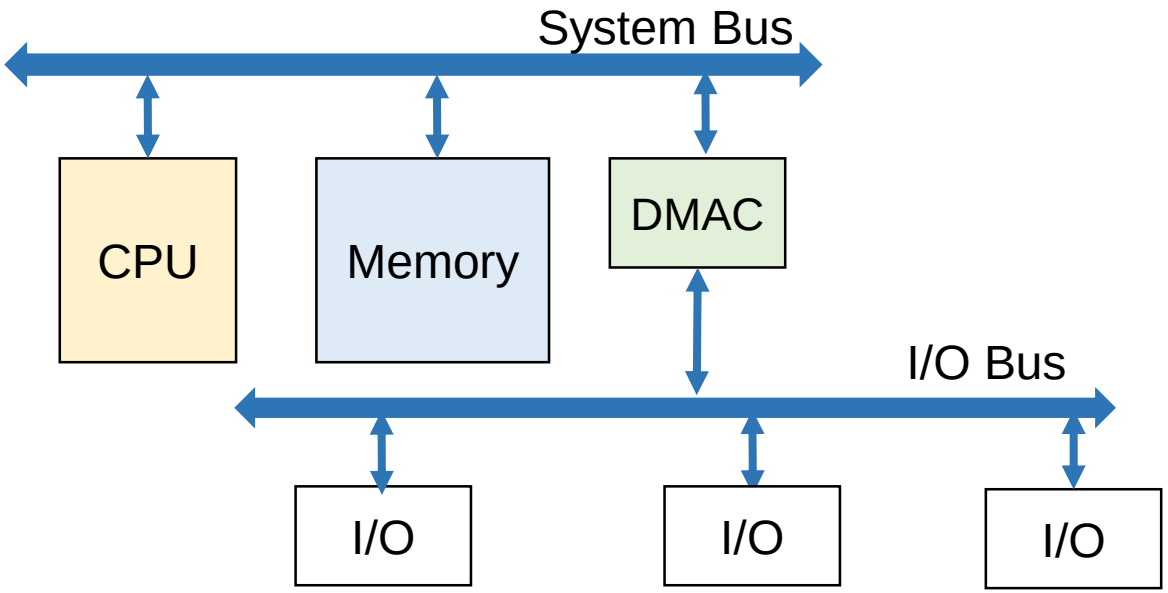
# DMA Configurations

- Single Bus, Integrated DMA controller
  - Controller may support more than 1 device
  - Each transfer uses bus once
    - DMAC to memory
  - CPU is suspended once



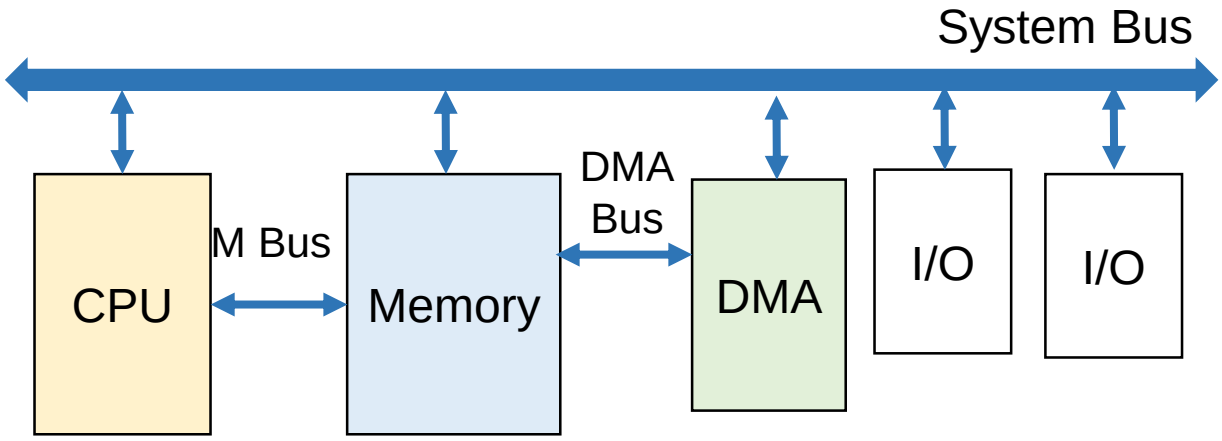
# DMA Configurations

- Separate I/O Bus
  - Bus supports all DMA enabled devices
  - Each transfer uses bus once
    - DMAC to memory
  - CPU is suspended once



# DMA Configurations

- Separate DMA Bus
  - CPU doesn't need to be suspended
  - Conflict when both CPU and DMA transfer access memory



# DMA Transfer Mode

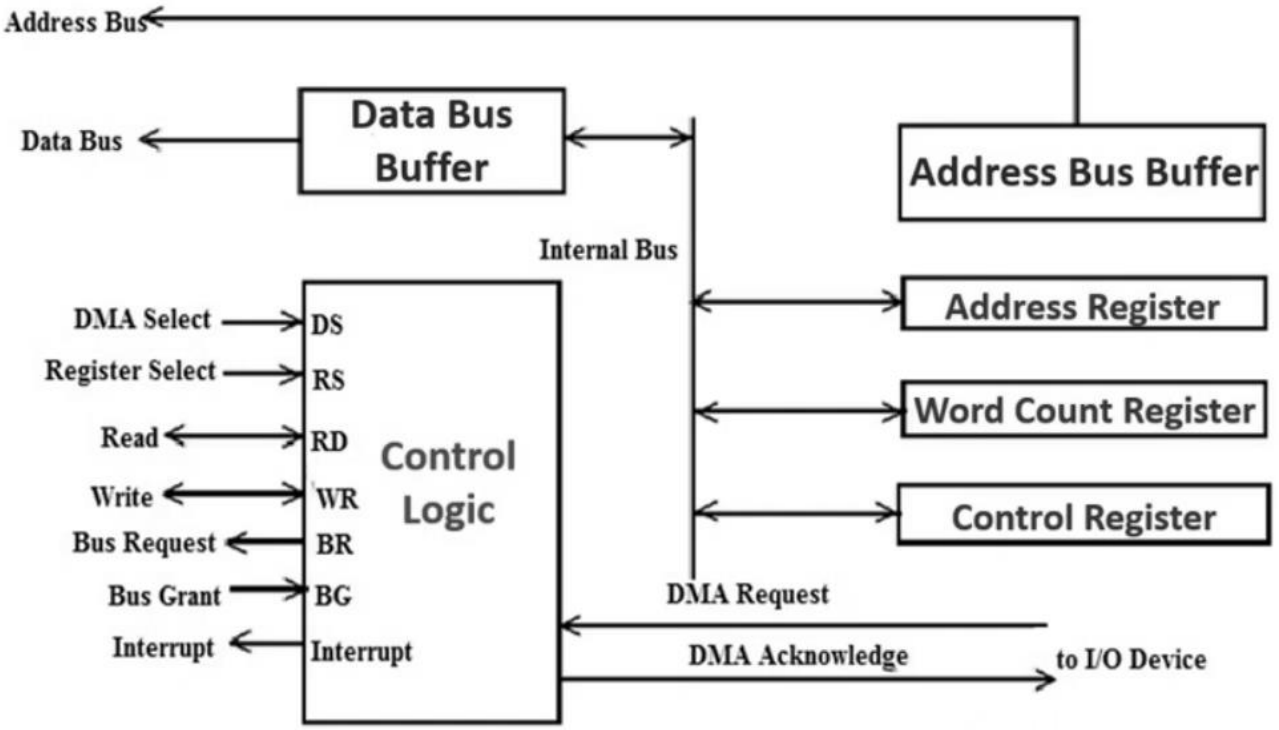
- Burst Mode
  - An entire block of data is transferred in one continuous operation, making it highly efficient.
  - The CPU remains inactive for relatively long periods of time during data transfer.
- Interleaving Mode
  - DMA transfers data only when there’s no conflict with CPU for system bus or memory access, ensuring no interference
- Cycle Stealing Mode
  - DMA controller takes control, transfers one word of data, releases control back to the CPU, and then repeats this process. This allows the CPU to continue executing its instructions between data transfers

# CPU Stall

- When DMAC takes over the bus access, or when there's a conflict accessing memory, CPU is stalled
- Stall is not an interrupt, as CPU does not switch context
- However, CPU still might execute instruction taking from cache if there's no cache miss

# DMA Controller (DMAC)

- Three registers:
  - **Address register**: contains the address to specify the desired location in memory
  - **Word count register**: contains the number of words to be transferred
  - **Control register**: specifies the transfer mode



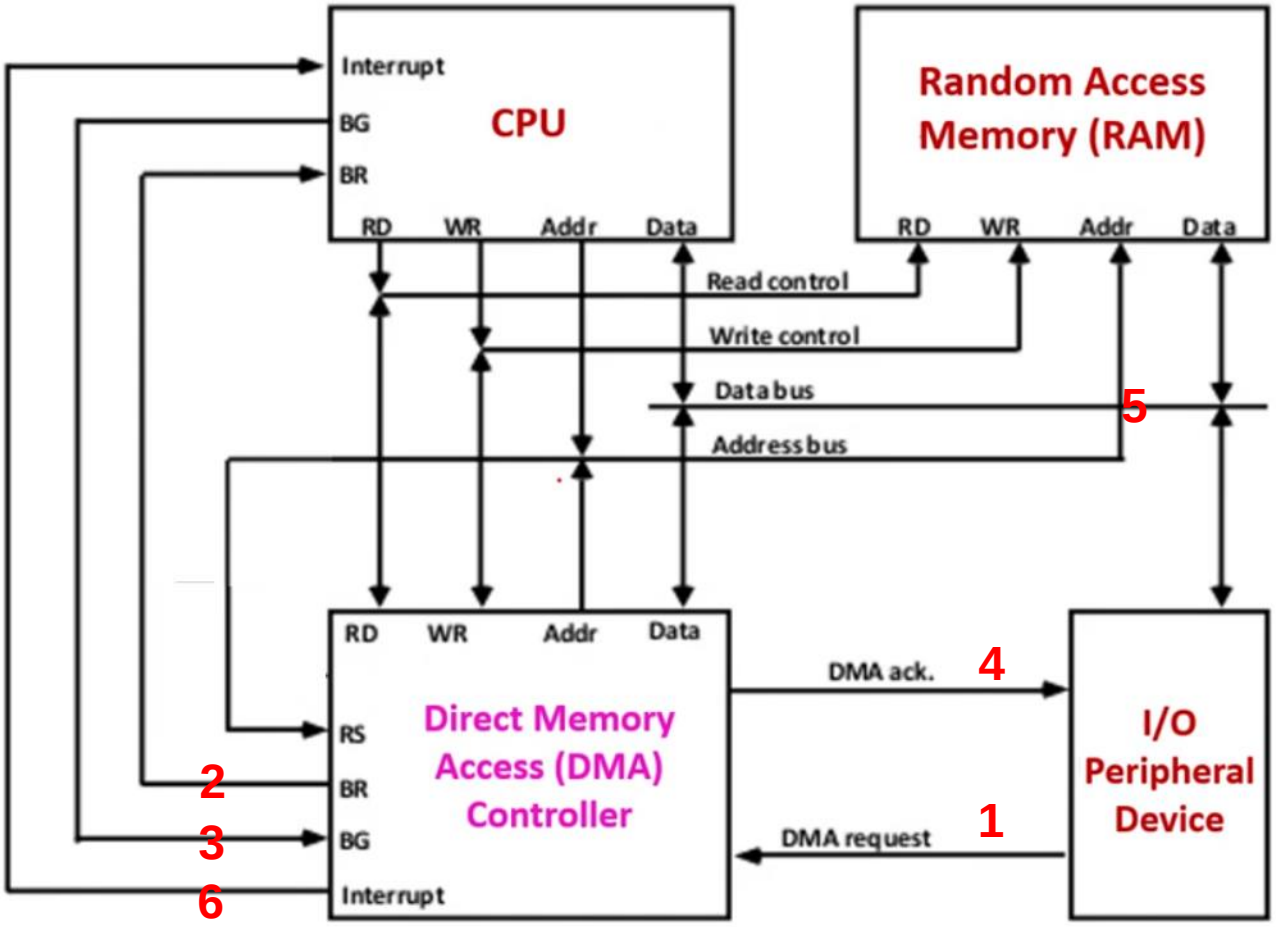
# DMA Operation

- Interaction with CPU
  - CPU tells DMA controller:
    - Data transfer direction between Mem and I/O
      - Read: e.g. read a file from disk to memory
      - Write: e.g. write memory content into disk
    - I/O Device address
    - Starting address of memory block for data
    - Amount of data to be transferred
  - CPU carries on with other work
  - DMA controller deals with transfer
  - DMA controller sends interrupt when finished

# DMA Operation

- Interaction with peripheral
  - When a word of data is available, the I/O device places a signal on the DMA-request wire.
  - The signal causes the DMA controller to seize the memory bus,
    - To place the desired address on the memory-address wire
    - To place a signal on the DMA-acknowledge wire
  - When the I/O device receives the DMA-acknowledge signal,
    - The transfer between external device and DMA begins

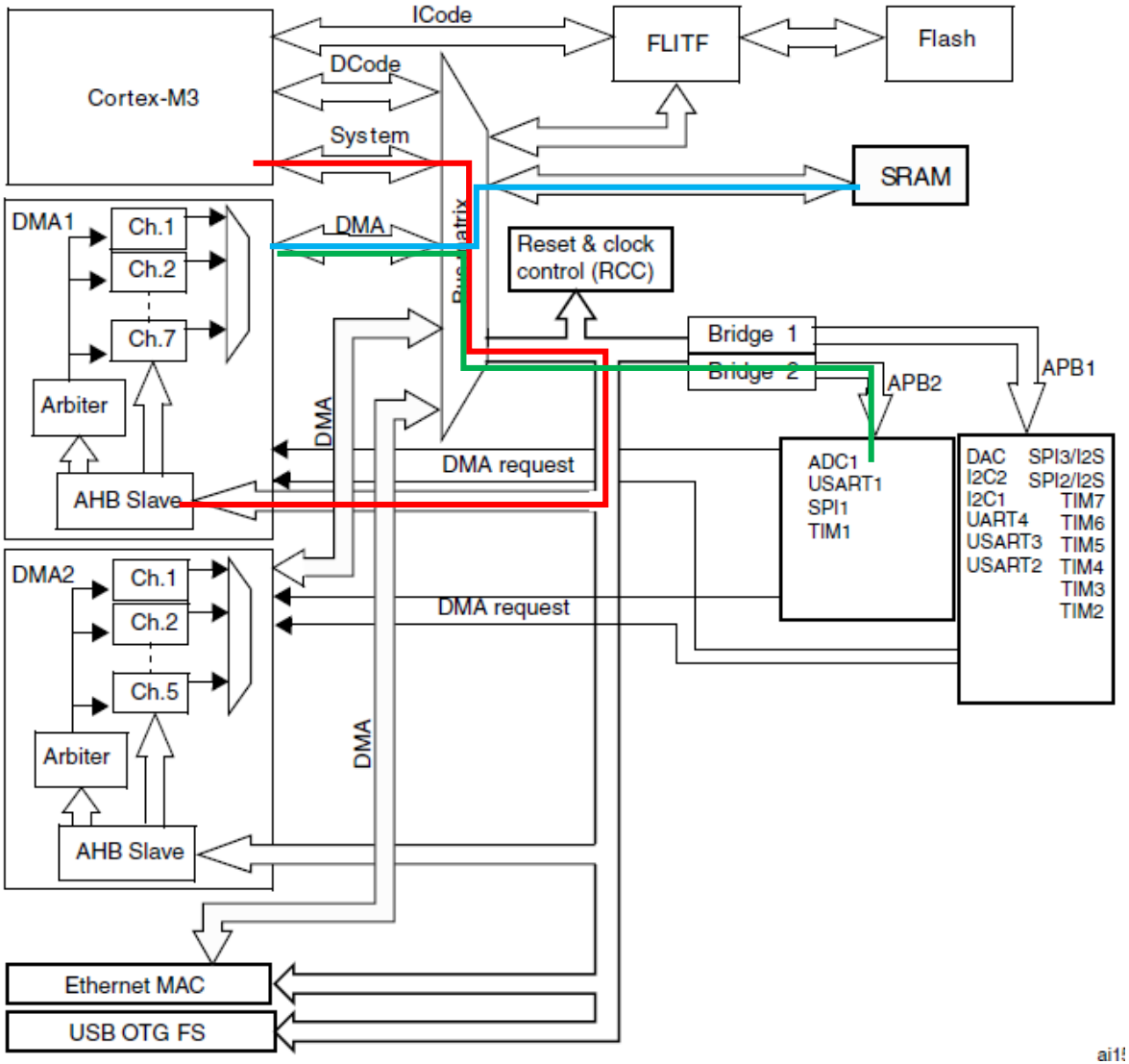
# DMA Operation



# Outline

- Why DMA
- DMA Architecture
- **STM32 DMA Controller**

# STM32 DMA Block Diagram



# STM32 DMA Channels

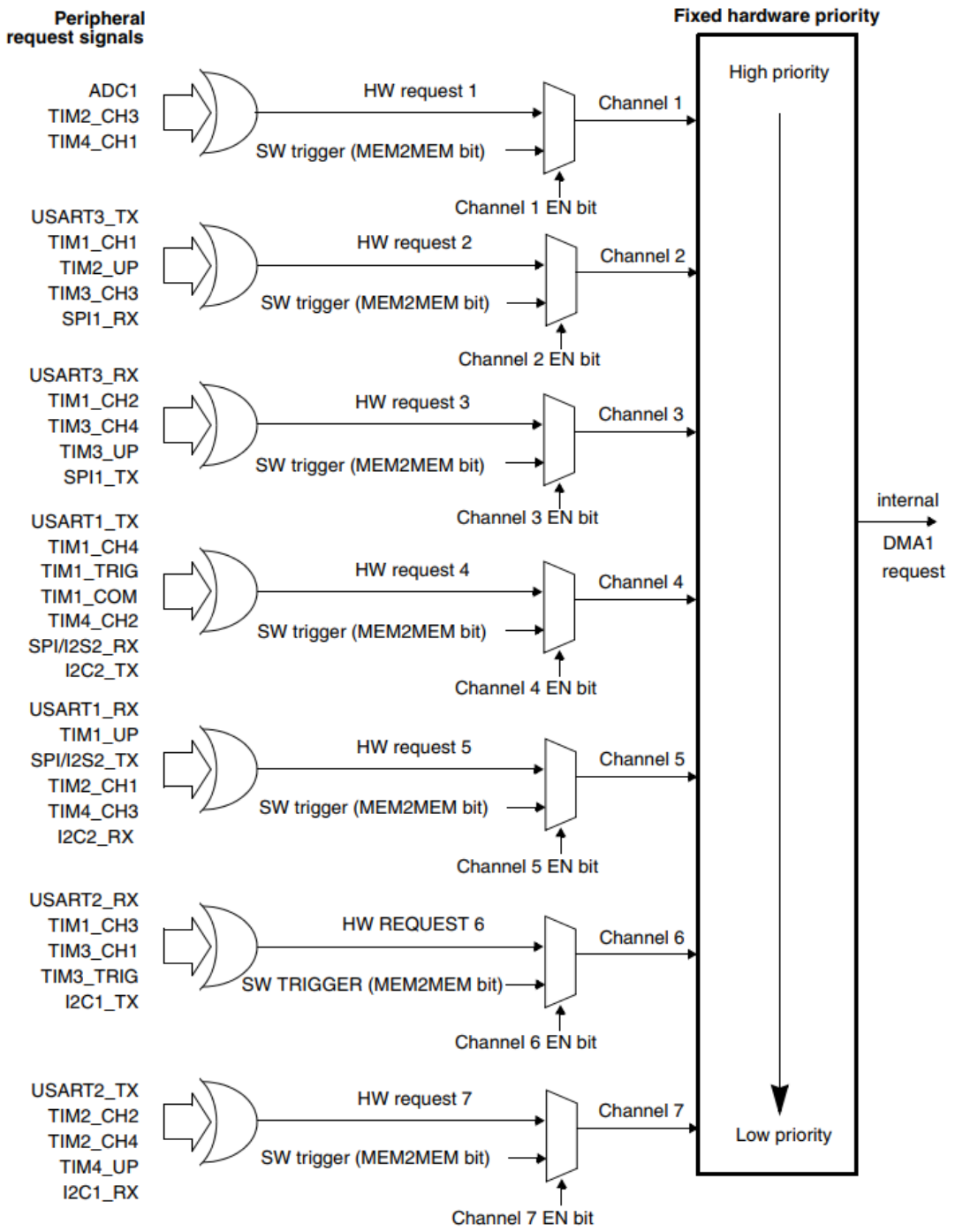
- DMA releases CPU from moving data
  - between peripherals and memory, or
  - between memory to memory, or
  - between one peripheral and another peripheral.
- DMA uses bus matrix to allow concurrent transfers

STM32 DMA1

| Peripherals | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7 |
|---|---|---|---|---|---|---|---|
| ADC1 | ADC1 | | | | | | |
| SPI | | SPI1_RX | SPI1_TX | SPI2_RX | SPI2_TX | | |
| USART | | USART3_TX | USART3_RX | USART1_TX | USART1_RX | USART2_RX | USART2_TX |
| I2C | | | | I2C2_TX | I2C2_RX | I2C1_TX | I2C1_RX |
| TIM2 | TIM2_CH3 | TIM2_UP | | | TIM2_CH1 | | TIM2_CH2<br>TIM2_CH4 |
| TIM3 | | TIM3_CH3 | TIM3_CH4<br>TIM3_UP | | | TIM3_CH1<br>TIM3_TRIG | |
| TIM4 | TIM4_CH1 | | | TIM4_CH2 | TIM4_CH3 | | TIM4_UP |
| TIM6<br>DAC_Ch1 | | TIM6_UP<br>DAC_Ch1 | | | | | |
| TIM7<br>DAC_CH2 | | | TIM7_UP<br>DAC_CH2 | | | | |

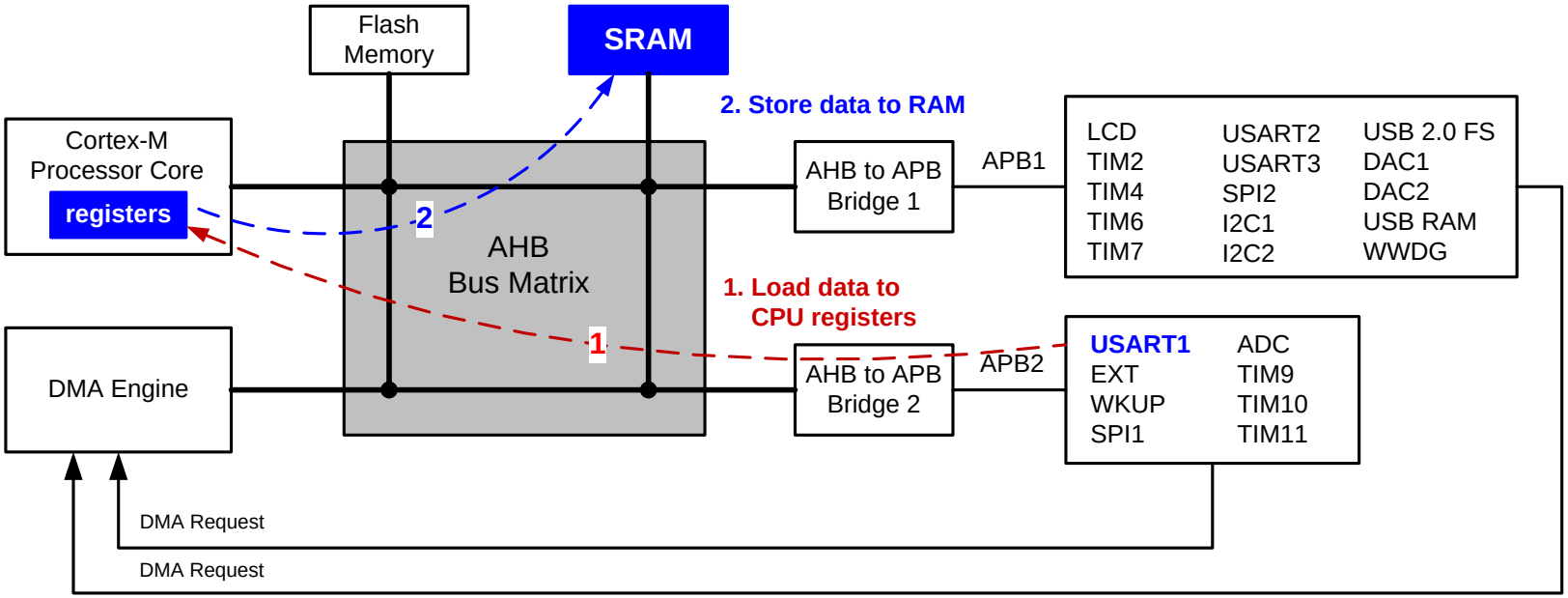
# STM32 DMA

- 12 channels
  - DMA1 has 7 channels
  - DMA2 has 5 channels



# Programmed I/Os

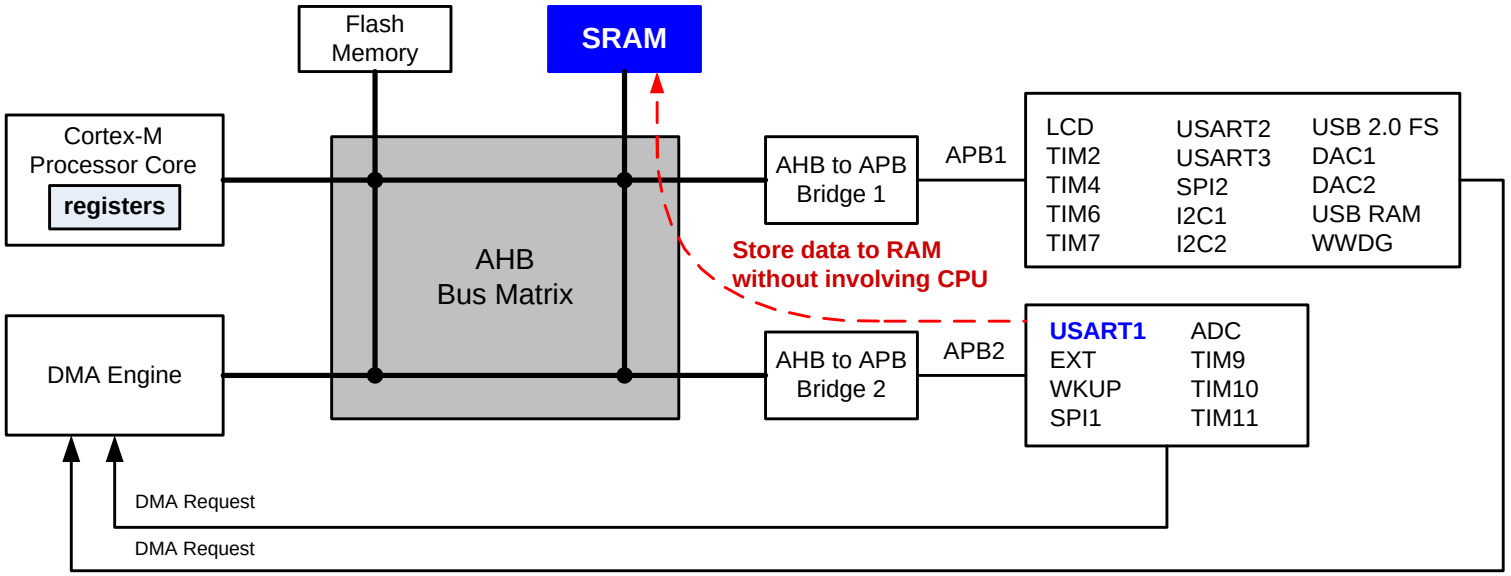
- Processor executes a lot Loads/Stores to move data
- High overhead and slow




Receiving data from USART serial port without using DMA

# DMA Sets Core Free

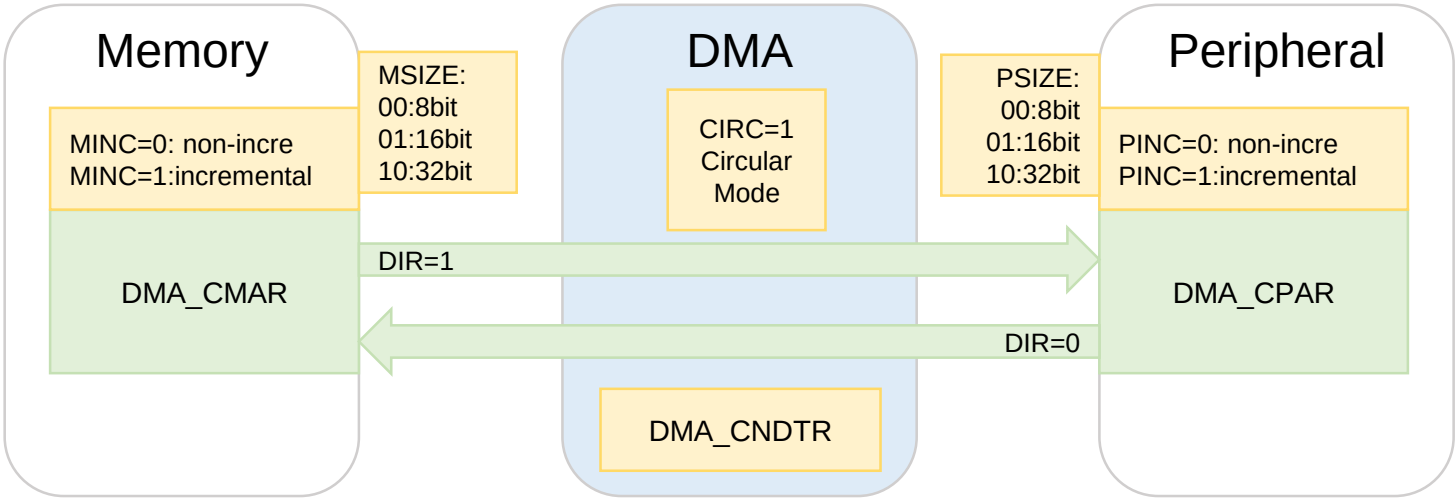
- CPU delegates reads/writes to DMA controller
- Low overhead and fast




Receiving data from USART serial port using DMA

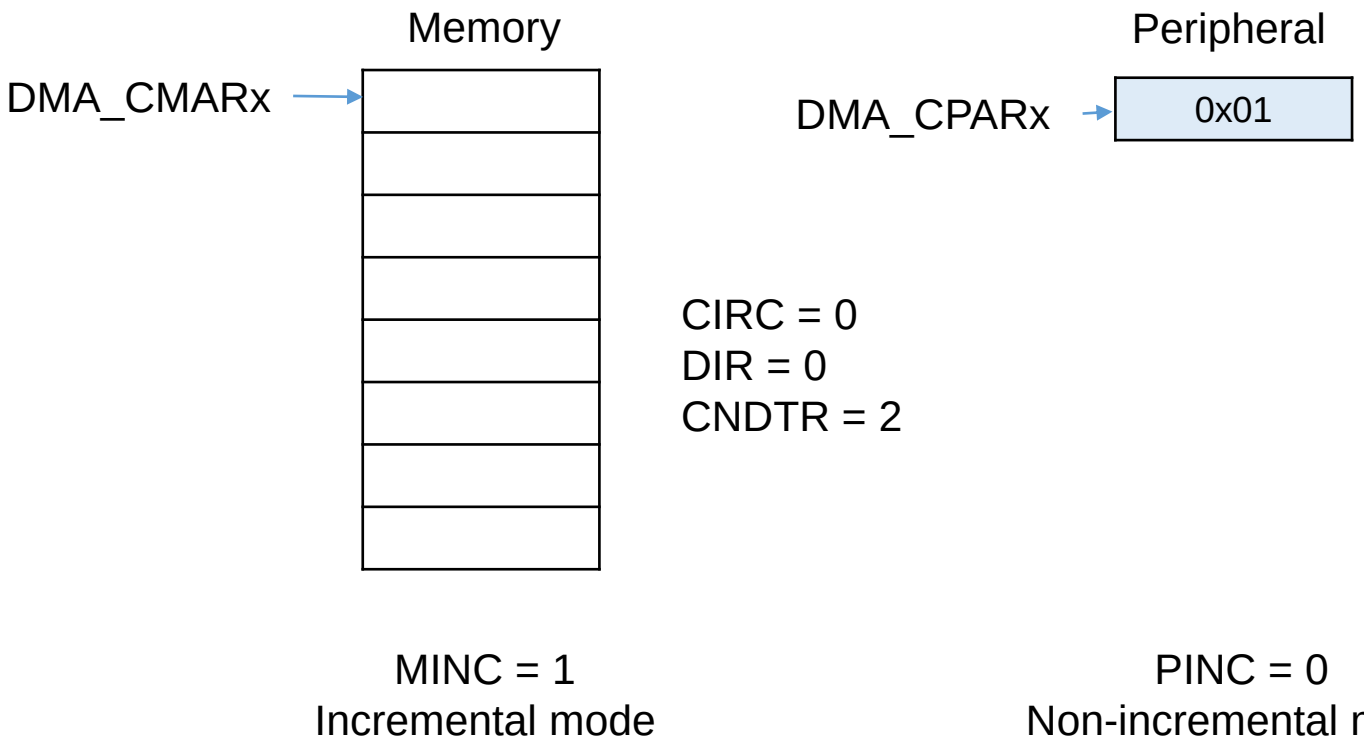
# DMA Controller

- Key DMA Controller Registers
  - DMA memory address register (CMAR)
  - DMA peripheral address register (CPAR)
  - DMA number of data register (CNDTR)
  - DMA configuration register (CCR)
- DMA are often used together with interrupts



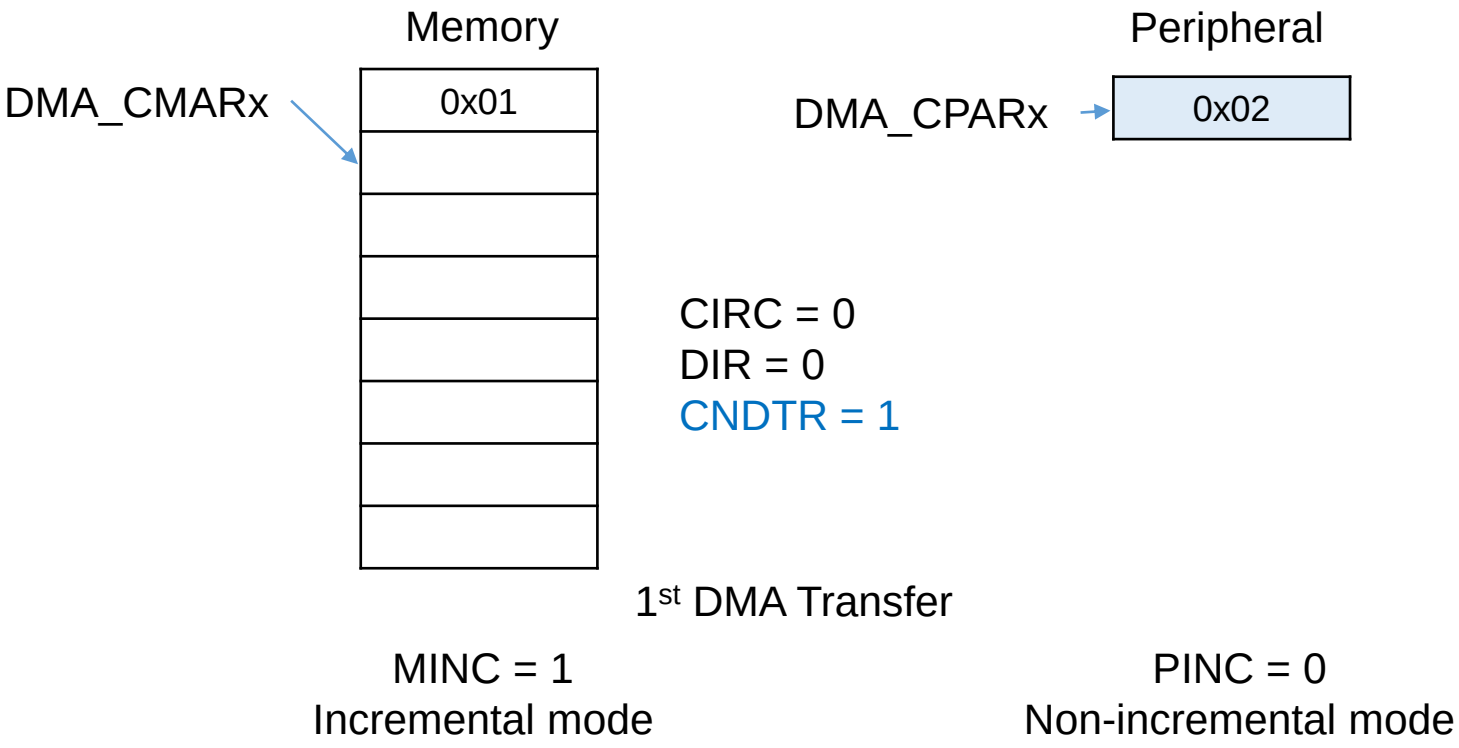
# STM32 DMA Transfer

- Incremental mode in one side
  - MINC = 1, PINC = 0
  - MSIZE = 0: 8bit, PSIZE = 0: 8bit



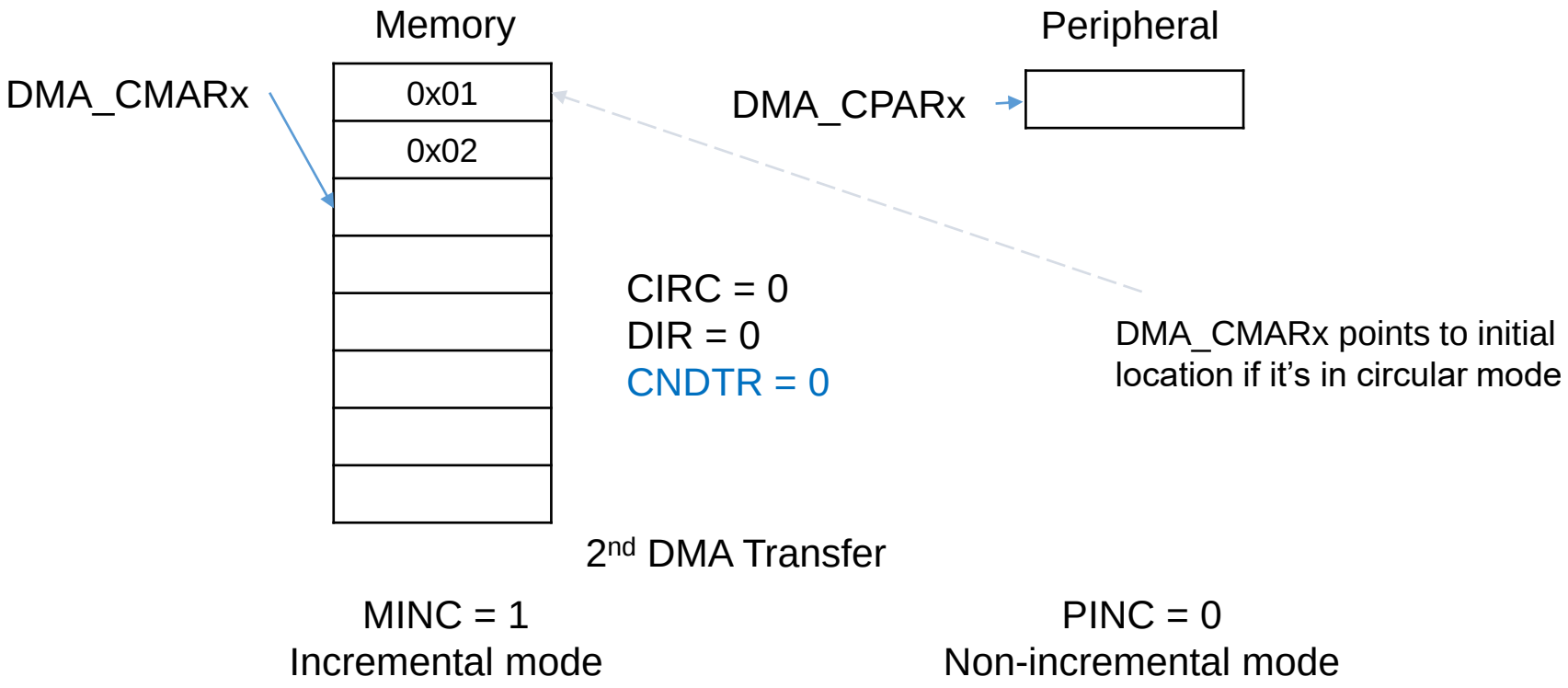
# STM32 DMA Transfer

- Incremental mode in one side
  - MICN = 1, PINC = 0
  - MSIZE = 0: 8bit, PSIZE = 0: 8bit
  - DIR = 0: Transfer data from peripheral to Memory



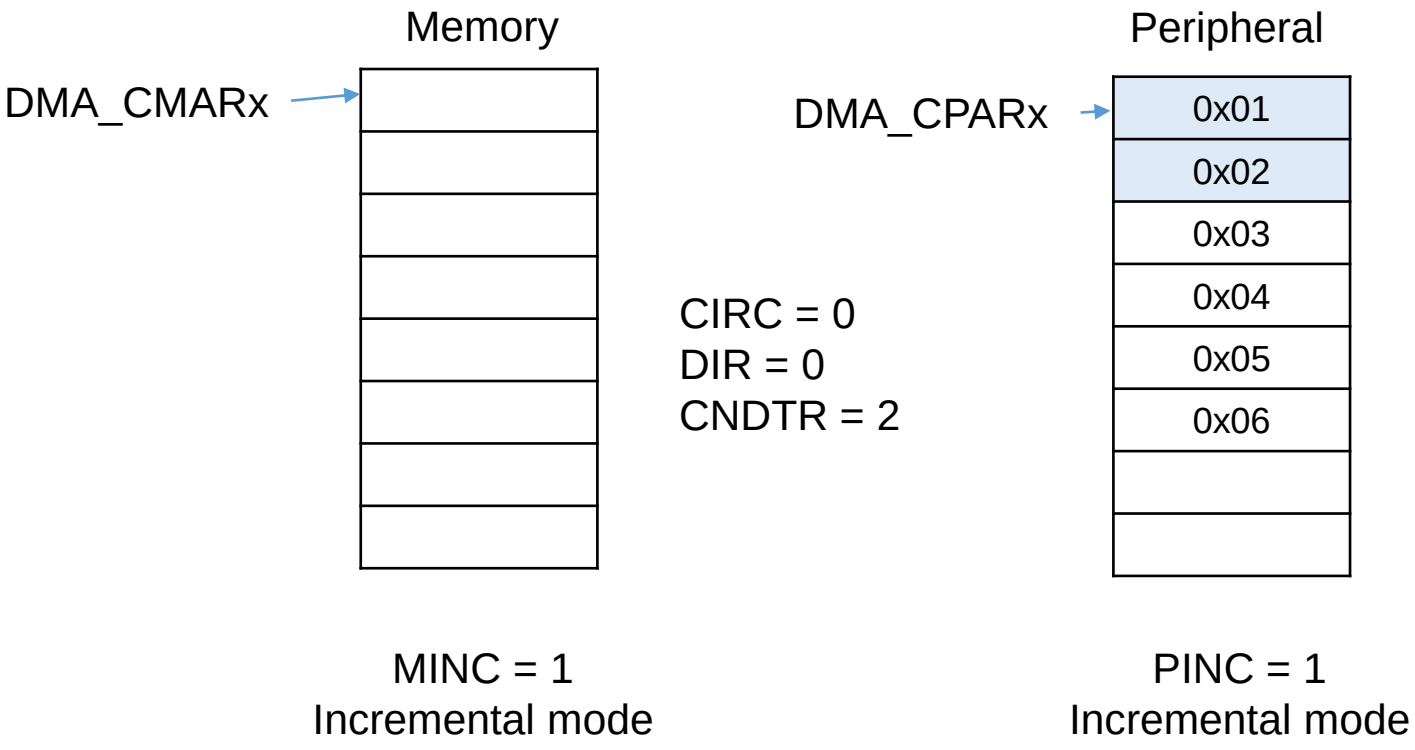
# STM32 DMA Transfer

- Incremental mode in one side
  - MICN = 1, PINC = 0
  - MSIZE = 0: 8bit, PSIZE = 0: 8bit
- DMA stops since CNDTR is zero now.



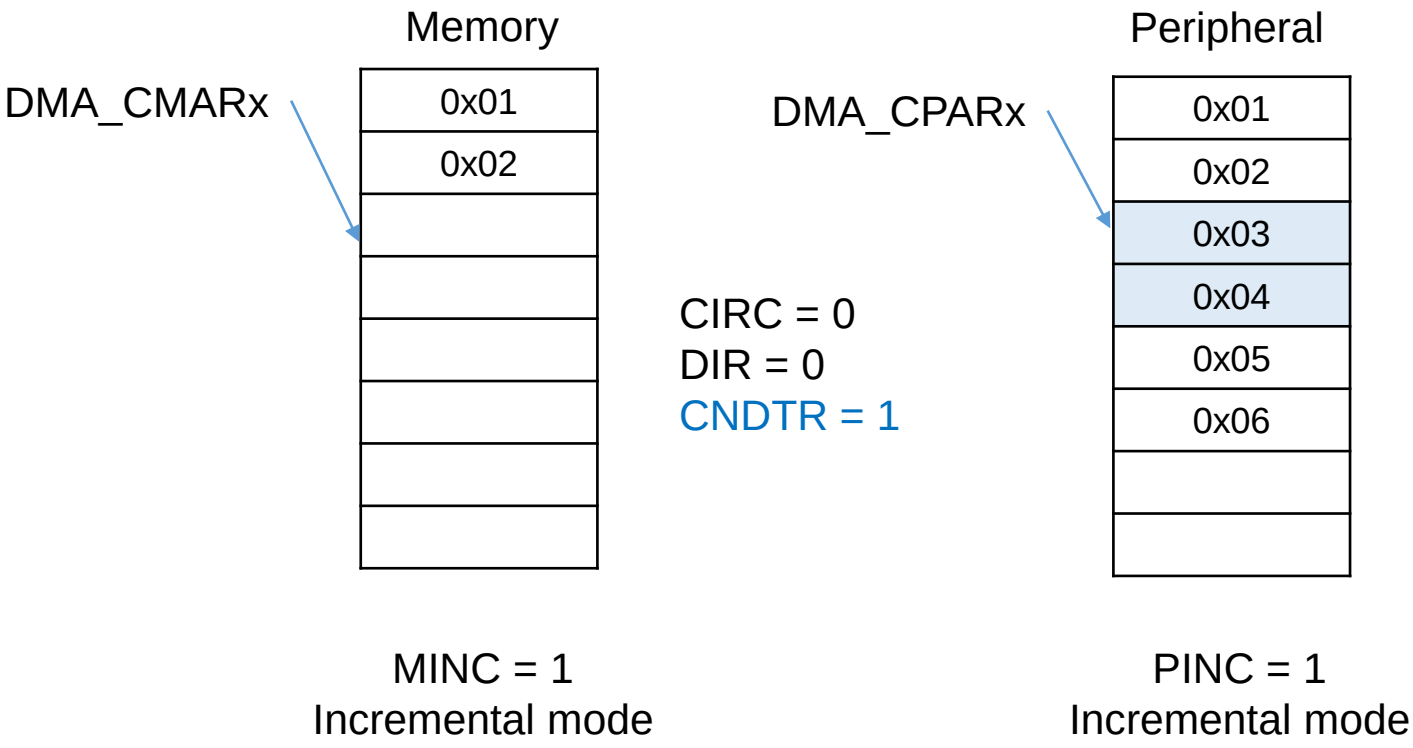
# STM32 DMA Transfer

- Incremental mode in both sides
  - MICN = 1, PINC = 1
  - MSIZE = 1: 16bit, PSIZE = 1: 16bit



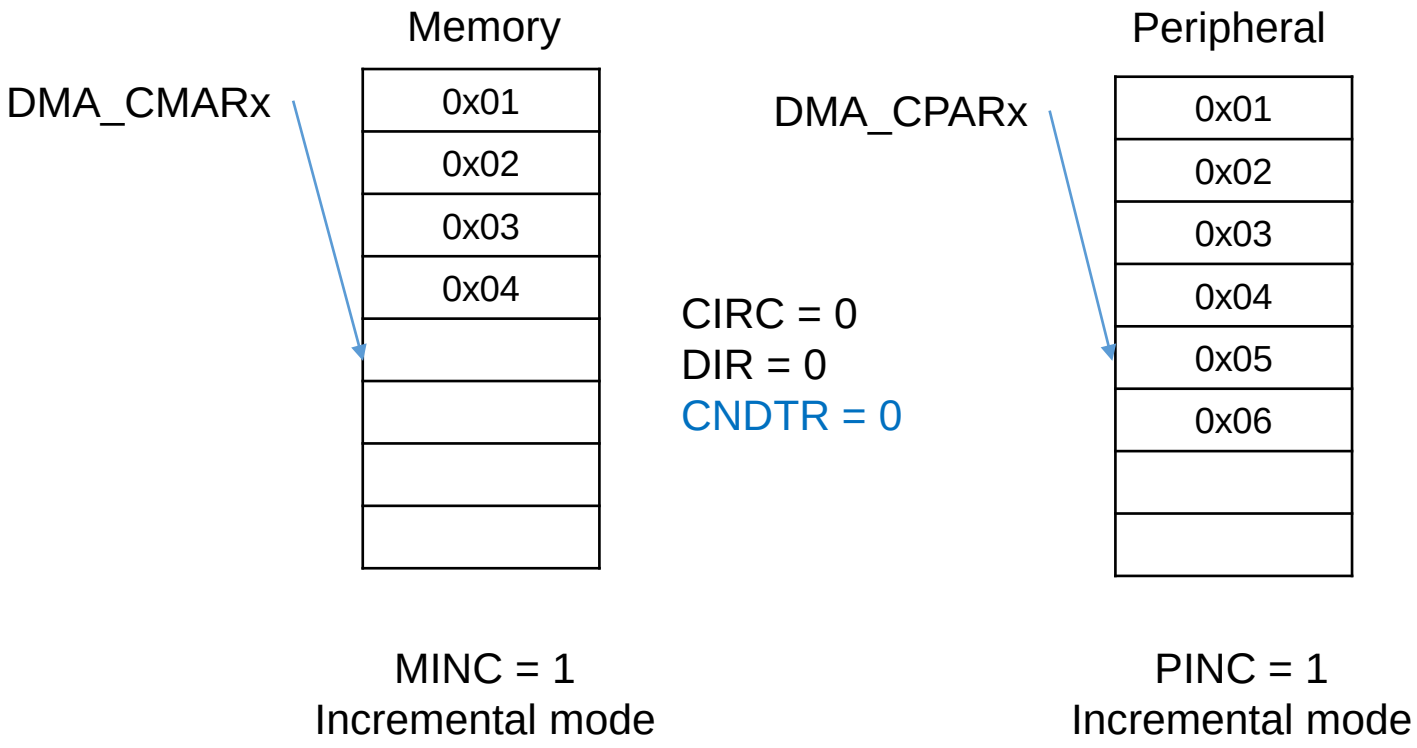
# STM32 DMA Transfer

- Incremental mode in both sides
  - MICN = 1, PINC = 1
  - MSIZE = 1: 16bit, PSIZE = 1: 16bit



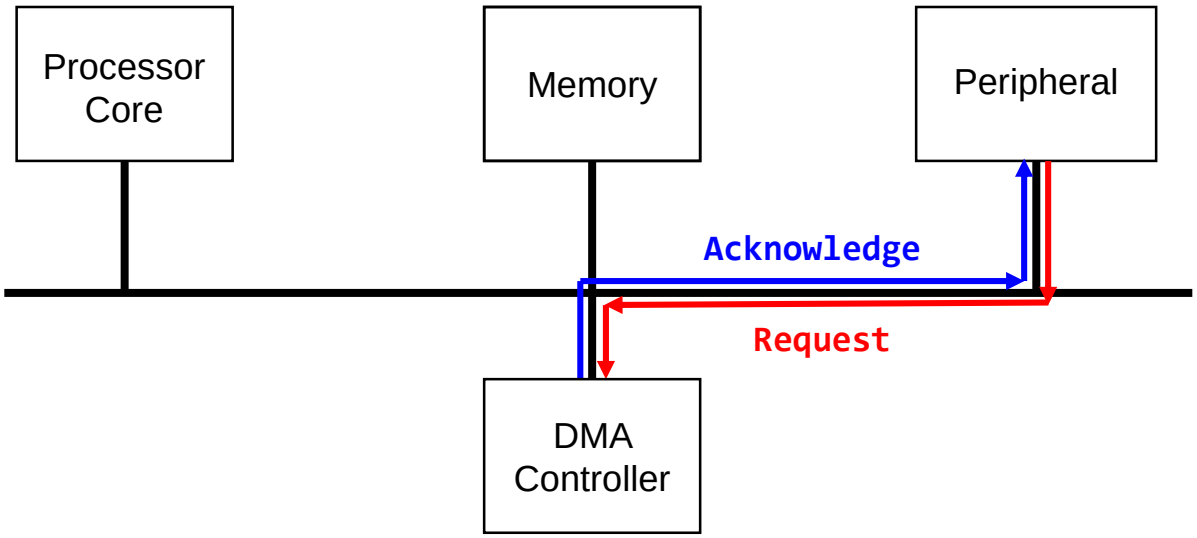
# STM32 DMA Transfer

- Incremental mode in both sides
  - MICN = 1, PINC = 1
  - MSIZE = 1: 16bit, PSIZE = 1: 16bit
- DMA stops since CNDTR is zero now



# DMA Request

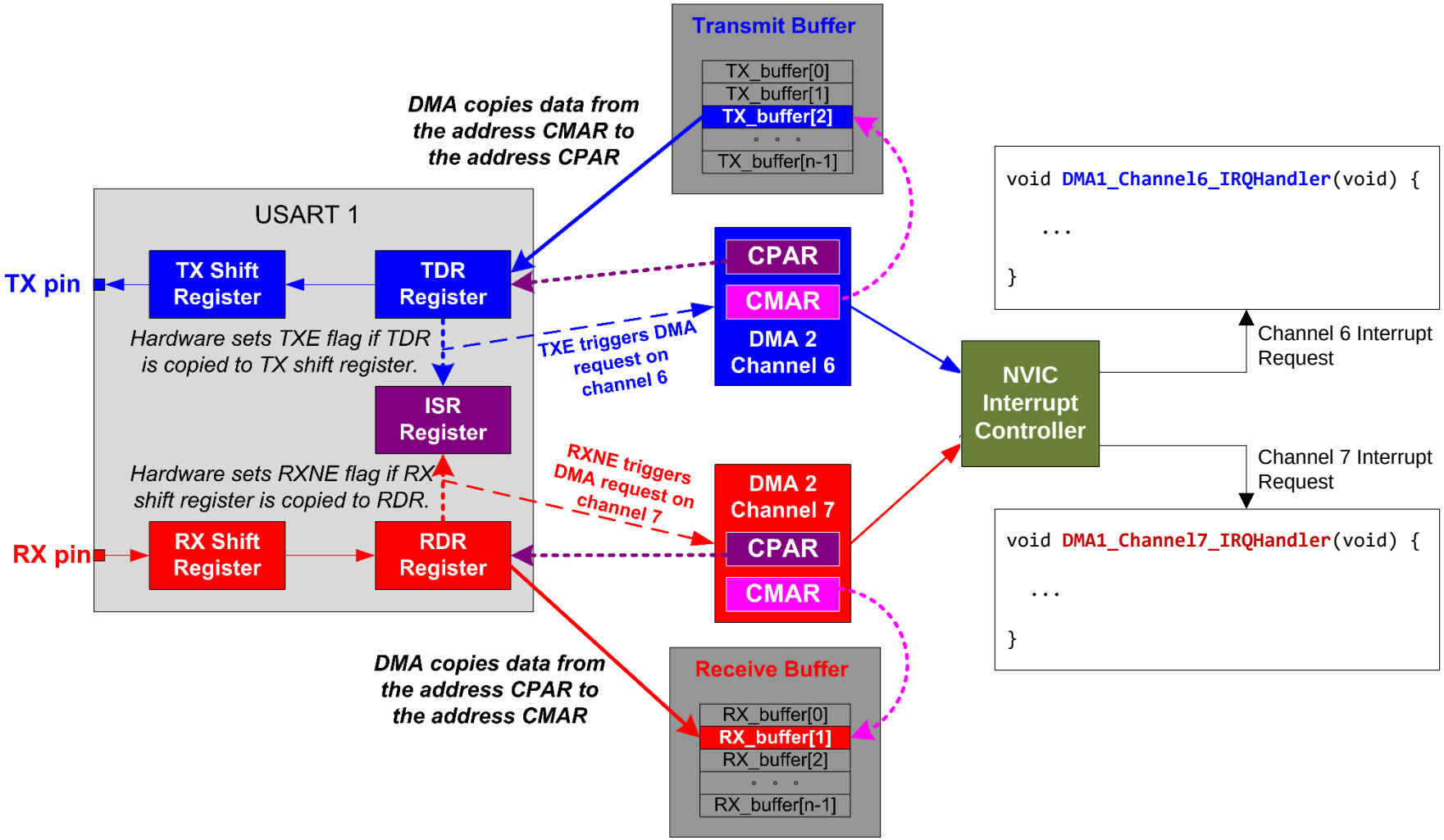
- When does the DMA transfer start?
  - When the peripheral is ready to send or receive data, the peripheral will generate a DMA request signal to the DMA controller to request a data transfer.



# DMA Interrupts

- Programmable and Independent source and destination transfer data size:
  - Byte, Halfword or Word
- Three event flags:
  - DMA Half Transfer
  - DMA Transfer complete
  - DMA Transfer Error

# UART DMA: Receiving & Sending



# UART DMA: Receiving & Sending