# DMA (Direct Memory Access) — Complete Guide

---

## Introduction

**DMA (Direct Memory Access)** is a **hardware feature that allows peripherals or memory modules to transfer data directly to/from memory without CPU intervention**.

- Improves CPU efficiency.
- Essential for **high-speed data acquisition**, audio/video processing, UART communication, SPI transfers, and ADC sampling.
- Widely used in **microcontrollers (STM32, AVR, PIC) and embedded SoCs**, as well as **microprocessors via external DMA controllers**.

---

## History of DMA

- **1970s–1980s:** Early DMA implemented as **external controllers** in microprocessor systems (e.g., Intel 8237 for 8086).
- **1990s–2000s:** Integrated DMA in microcontrollers (STM32, NXP, PIC).
- **Modern systems (x86, ARM Cortex-A):** DMA integrated in **SoC peripherals, memory controllers, and PCIe devices**.

**Key point:**

- Microcontrollers → DMA is **on-chip**.
- Microprocessors → DMA often **external** or in **chipset/PCH**.

---

## Why DMA is important

| Problem without DMA | Solution with DMA |
|---|---|
| CPU copies data manually | DMA copies data automatically |
| High CPU usage | CPU free for other tasks |
| Limited sampling rates | High-speed ADC / SPI / UART |
| Real-time data may be missed | Reliable, fast transfers |

---

## How DMA works

1. **Setup phase** (CPU configures DMA registers):
   - Source (CPAR)
   - Destination (CMAR)
   - Transfer size (CNDTR)
   - Direction, increment, circular mode (CCR)
2. **Enable DMA channel** → CCR.EN = 1
3. **Peripheral triggers DMA** (or memory-to-memory transfer)
4. DMA reads/writes automatically → decrements **CNDTR**
5. Optional **interrupt** when transfer completes
6. In **circular mode**, DMA automatically restarts

**Analogy:** CPU = chef, DMA = assistant moving ingredients from fridge to stove.

---

## STM32 DMA Architecture

- STM32F1 has **DMA1** (7 channels) and **DMA2** (5–7 channels depending on MCU).
- **Global registers:**
  - `DMAx_ISR` → status flags (TCIF, HTIF, TEIF)
  - `DMAx_IFCR` → clear flags
- **Channel registers:**
  - `CCR` → configuration
  - `CNDTR` → number of transfers
  - `CPAR` → peripheral address
  - `CMAR` → memory address

---

## DMA Channel Registers (Detailed)

### DMA_CCRx (Channel Configuration Register)

| Bit | Function |
|---|---|
| 0 | EN: Channel enable |
| 1 | TCIE: Transfer complete interrupt enable |
| 2 | HTIE: Half-transfer interrupt enable |
| 3 | TEIE: Transfer error interrupt enable |
| 4 | DIR: 0=Periph→Mem, 1=Mem→Periph |
| 5 | CIRC: Circular mode |
| 6 | PINC: Peripheral increment |

| | |
|---|---|
| 7 | MINC: Memory increment |
| 8-9 | PSIZE: Peripheral size (8/16/32-bit) |
| 10-11 | MSIZE: Memory size (8/16/32-bit) |
| 12-13 | PL: Priority level |
| 14 | MEM2MEM: Memory-to-memory mode |

---

### DMA_CNDTRx

- Sets the **number of data items** to transfer.
- Automatically decremented by DMA.

### DMA_CPARx

- Peripheral source/destination address.

### DMA_CMARx

- Memory source/destination address.

---

### DMA_ISR / DMA_IFCR

- `ISR` → shows flags (Transfer Complete, Half-Transfer, Transfer Error)
- `IFCR` → clear flags by writing 1

**Example:**

```
if(DMA1->ISR & DMA_ISR_TCIF1) { /* Transfer complete */ }
DMA1->IFCR |= DMA_IFCR_CTCIF1; // Clear flag
```

---

## DMA Transfer Types

| Type | Direction | Use case |
|---|---|---|
| Peripheral → Memory | e.g., ADC → buffer | High-speed sampling |
| Memory → Peripheral | e.g., UART TX | Send large data |
| Memory → Memory | Copy buffers | No CPU usage |
| Circular Mode | Continuous transfer | ADC sampling, audio, sensors |
| Double Buffer | Ping-pong buffers | Real-time high-speed processing |

## Practical Steps to Use DMA (STM32)

1. **Enable DMA clock**

```
RCC->AHBENR |= RCC_AHBENR_DMA1EN;
```

2. **Configure CPAR, CMAR, CNDTR**

```
DMA1_Channel1->CPAR = (uint32_t)&ADC1->DR;
DMA1_Channel1->CMAR = (uint32_t)adc_buffer;
DMA1_Channel1->CNDTR = 64;
```

3. **Configure CCR** (direction, size, increment, circular)

```
DMA1_Channel1->CCR = DMA_CCR_MINC | DMA_CCR_PSIZE_0 | DMA_CCR_MSIZE_0 |
DMA_CCR_CIRC;
```

4. **Clear flags**

```
DMA1->IFCR = 0x0F;
```

5. **Enable channel**

```
DMA1_Channel1->CCR |= DMA_CCR_EN;
```

6. **Enable peripheral DMA request**

```
ADC1->CR2 |= ADC_CR2_DMA | ADC_CR2_ADON;
```

## Example Code: ADC with DMA

```
#define N 64
uint16_t adc_buffer[N];

void DMA_ADC_Init(void){
    RCC->AHBENR |= RCC_AHBENR_DMA1EN;
    DMA1_Channel1->CCR &= ~DMA_CCR_EN;

    DMA1_Channel1->CPAR  = (uint32_t)&ADC1->DR;
    DMA1_Channel1->CMAR  = (uint32_t)adc_buffer;
    DMA1_Channel1->CNDTR = N;

    DMA1_Channel1->CCR = DMA_CCR_MINC | DMA_CCR_PSIZE_0 | DMA_CCR_MSIZE_0 |
DMA_CCR_CIRC;

    DMA1->IFCR = 0x0F;
    DMA1_Channel1->CCR |= DMA_CCR_EN;

    ADC1->CR2 |= ADC_CR2_DMA | ADC_CR2_ADON;
}
```

- Now the ADC automatically fills `adc_buffer` continuously.

---

## Advanced DMA Concepts

1. **Interrupts:** Half-transfer, transfer complete, error
2. **Priority Levels:** Low, Medium, High, Very High
3. **Double Buffer Mode:** Alternate between two memory buffers
4. **Memory-to-Memory Transfer:** High-speed buffer copying
5. **Bus Mastering (in microprocessors/PCIe):** DMA works independently of CPU

---

## DMA Advantages

- Reduces CPU load
- Enables real-time data capture
- Supports very high-speed transfers
- Circular mode for continuous streaming
- Essential for **embedded oscilloscope, audio, graphics**

---

## Self-Study Exercises

1. **Exercise 1:** Configure DMA1 Channel1 for ADC1 in circular mode and store 128 samples. Print via UART.
2. **Exercise 2:** Configure UART TX with DMA and send a string of 256 bytes.
3. **Exercise 3:** Implement memory-to-memory DMA to copy a large buffer.
4. **Exercise 4:** Use half-transfer interrupt to process first half of ADC buffer while second half fills.
5. **Exercise 5:** Compare CPU-only ADC sampling vs DMA ADC sampling speed.
6. **Exercise 6:** Read DMA ISR flags and handle transfer complete.

---

## Summary

**DMA is the ultimate tool for high-speed embedded systems**. Key points:

- DMA moves data **without CPU intervention**
- STM32 DMA = on-chip peripheral, 7 channels per DMA1, 5–7 for DMA2
- Configure **CPAR, CMAR, CNDTR, CCR**, then enable DMA
- Supports **circular, double buffer, memory-to-memory, peripheral-to-memory**
- Essential for **ADC sampling, UART/SPI transfer, audio/video, graphics**

---