# Design Specification of DMA

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Author** | **Changes** |
| 2022/3/15 | 1.0 | QinDong | Initial version. |

## Overview:

DMA传输将数据从一个地址空间复制到另一个地址空间，提供在外设和存储器之间或者存储器和存储器之间的高速数据传输，CPU 只是初始化这个传输动作，传输动作本身是由 DMA 控制器来完成。

## Competitor comparison:

|  |  |  |
| --- | --- | --- |
|  | STM32F103 | AT103 |
| channel | 12 (7+5) | 8 |
| data width | 8/16/32 | 8/16/32 |
| max block size | 65535 | 4095 |
| address increment | Increment/No change | Increment/Decrement/No change |
| handshake | hw | hw |
| block circular | yes | 通过reload功能支持 |
| irq | half-transfer/tfr/err | tfr/err |
| m2p/p2m | 通道绑定外设 | 有一定灵活度 |
| priority | 4 | 8 |
| burst size | 个别外设支持 | yes |

## Design:

### **HAL DMA API list:**

|  |  |
| --- | --- |
| STM32F103VB | AT103 |
| DMA\_DeInit(...) | |
| DMA\_Init(...) | |
| DMA\_StructInit(...) | |
| DMA\_Cmd(...) | |
| DMA\_ITConfig(...) | no half-transfer irq |
| DMA\_SetCurrDataCounter(...) | |
| DMA\_GetCurrDataCounter(...) | After the transfer has started, returns the number of completing data units in the current DMA Channelx transfer.  Before the transmission starts or after the transmission ends, the return values are the values set before the transfer starts. |
| DMA\_GetFlagStatus(...) | |
| DMA\_ClearFlag(...) | |
| DMA\_GetITStatus(...) | |
| DMA\_ClearITPendingBit(...) | |
|  | DMA\_Channelx\_IRQHandler(wake) |

typedef struct  
{  
 uint32\_t DMA\_PeripheralBaseAddr; /\*\*< Specifies the peripheral base address for DMAy Channelx. \*/  
  
 uint32\_t DMA\_MemoryBaseAddr; /\*\*< Specifies the memory base address for DMAy Channelx. \*/  
  
 uint32\_t DMA\_DIR; /\*\*< Specifies if the peripheral is the source or destination.  
 This parameter can be a value of @ref DMA\_data\_transfer\_direction \*/  
  
 uint32\_t DMA\_BufferSize; /\*\*< Specifies the buffer size, in data unit, of the specified Channel.   
 The data unit is equal to the configuration set in DMA\_PeripheralDataSize  
 or DMA\_MemoryDataSize members depending in the transfer direction. \*/  
  
 uint32\_t DMA\_PeripheralInc; /\*\*< Specifies whether the Peripheral address register is incremented or not.  
 This parameter can be a value of @ref DMA\_peripheral\_incremented\_mode \*/  
  
 uint32\_t DMA\_MemoryInc; /\*\*< Specifies whether the memory address register is incremented or not.  
 This parameter can be a value of @ref DMA\_memory\_incremented\_mode \*/  
  
 uint32\_t DMA\_PeripheralDataSize; /\*\*< Specifies the Peripheral data width.  
 This parameter can be a value of @ref DMA\_peripheral\_data\_size \*/  
  
 uint32\_t DMA\_MemoryDataSize; /\*\*< Specifies the Memory data width.  
 This parameter can be a value of @ref DMA\_memory\_data\_size \*/  
  
 uint32\_t DMA\_Mode; /\*\*< Specifies the operation mode of the DMAy Channelx.  
 This parameter can be a value of @ref DMA\_circular\_normal\_mode. \*/  
  
 uint32\_t DMA\_Priority; /\*\*< Specifies the software priority for the DMAy Channelx.  
 This parameter can be a value of @ref DMA\_priority\_level \*/  
  
 uint32\_t DMA\_M2M; /\*\*< Specifies if the DMAy Channelx will be used in memory-to-memory transfer.  
 This parameter can be a value of @ref DMA\_memory\_to\_memory \*/  
 uint32\_t DMA\_PeripheralHandshake; /\*\*< Specifies the hardware handshake between the peripheral and DMA Channelx.  
 This parameter can be a value of @ref DMA\_peripherals\_handshake\_definition \*/  
}DMA\_InitTypeDef;

## How to use this driver:

DMA\_DeInit(DMA\_Channel1);  
 DMA\_InitStructure.DMA\_PeripheralBaseAddr = Base\_Addr;  
 DMA\_InitStructure.DMA\_MemoryBaseAddr = (uint32\_t)TxBuffer;  
 DMA\_InitStructure.DMA\_DIR = DMA\_DIR\_PeripheralDST;  
 DMA\_InitStructure.DMA\_BufferSize = TxBufferSize;  
 DMA\_InitStructure.DMA\_PeripheralInc = DMA\_PeripheralInc\_Disable;  
 DMA\_InitStructure.DMA\_MemoryInc = DMA\_MemoryInc\_Enable;  
 DMA\_InitStructure.DMA\_PeripheralDataSize = DMA\_PeripheralDataSize\_Byte;  
 DMA\_InitStructure.DMA\_MemoryDataSize = DMA\_MemoryDataSize\_Byte;  
 DMA\_InitStructure.DMA\_Mode = DMA\_Mode\_Normal;  
 DMA\_InitStructure.DMA\_Priority = DMA\_Priority\_VeryHigh;  
 DMA\_InitStructure.DMA\_M2M = DMA\_M2M\_Disable;  
 DMA\_InitStructure.DMA\_PeripheralHandshake = DMA\_PeripheralHandshake\_USART1\_TX;  
 /\* Configure DMA Channel1 \*/  
 DMA\_Init(DMA\_Channel1, &DMA\_InitStructure);  
  
 USART\_InitStructure.USART\_BaudRate = 115200;  
 USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;  
 USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;  
 USART\_InitStructure.USART\_Parity = USART\_Parity\_No;  
 USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;  
 /\* Configure USART1 \*/  
 USART\_Init(USART1, &USART\_InitStructure);  
  
 /\* Enable USART1 DMA TX request \*/  
 USART\_DMACmd(USART1, USART\_DMAReq\_Tx, ENABLE);  
  
 /\* Enable USART1 TX DMA Channel \*/  
 DMA\_Cmd(DMA\_Channel1, ENABLE);

## Known behavior and issue: