二、　　一个因初始化顺序而导致异常的话题

有 STM32 用户反映，他使用 STM32F4 系列芯片进行开发，通过 STM32CubeMx 配置初始化代码，使用了 UART 的 DMA 传输。但他发现 DMA 根本不工作。后来他无意中发现，是因为他在用户代码里不经意地调整过 UART 外设和 DMA 外设初始化代码的前后顺序，当他重新调整二者的先后顺序后就一切正常了。他想知道这个顺序是怎么影响 DMA 功能的。

我顺手拿了块 STM32F334 的 Nucleo 板，开启 UART1/UART3 的数据通信功能，使用 DMA 进行数据的循环传输。UART1 发送数据，UART3 接收数据，均配置在异步双工模式。基于 STM32CubeMx 配置后生成初始化代码，添加用户代码。如下图所示：

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_USART1_UART_Init();
MX_USART3_UART_Init();
/* USER CODE BEGIN 2 */
HAL_UART_Receive_DMA(&huart3, Data_RX, Length);
HAL_UART_Transmit_DMA(&huart1, Data_TX, Length);
```

经测试验证，发现基于 UART1/3 的 DMA 传输功能是正常的。

结合客户的反馈，我将 DMA 与 UART 初始化顺序前后调换下，如下图：

```
/* Initialize all configured peripherals */

MX_GPIO_Init();

MX_USART1_UART_Init();
MX_USART3_UART_Init();

MX_DMA_Init();

/* USER CODE BEGIN 2 */

HAL_UART_Receive_DMA(&huart3, Data_RX, Length);

HAL_UART_Transmit_DMA(&huart1, Data_TX, Length);
```

果然发现，DMA 真的不工作了，UART1/UART3 之间没有发生数据通信。通过调试发现，UART1/3 的数据寄存器内容维持 0 值而没有任何变化，尤其作为发送端的 UART1 的数据寄存器也毫无动静。

看来，DMA 和 UART 的初始化代码的顺序的确影响到了二者的功能。也就是说如果代码是基于现有 CubeMX 生成的初始化代码，二者的初始化顺序不能随意调整，那到底怎么回事呢？

首先自然是查看这两个初始化代码内容，试图找到蛛丝马迹。很遗憾，并未很快发现原因。当再次查看 DMA 初始化函数 MX_DMA_Init();的具体内容时，发现其代码其实很简单，就两个动作，几行代码而已：

```
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Channel3_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel3_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel3_IRQn);
  /* DMA1_Channel4_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel4_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel4_IRQn);

}
```

一个开启 DMA 外设的时钟，另一个动作就是使能 DMA 相关的中断矢量控制。

既然这样，我尝试将该 DMA 初始化函数体位置依然放在 UART 初始化代码的后面，但将 DMA 初始化函数里的那句开启 DMA 外设时钟的代码提取出来，并移到 UART 初始化代码之前，据此进行验证。这次，结果又一切正常了。

也就是说，基于现有初始化代码，这个 DMA 时钟的开启要放在 UART 初始化代码之前，那是为什么呢？感觉 UART 的配置跟 DMA 时钟并没有啥关系啊。

再回头细看 UART 的初始化代码，在 UART 初始化函数的一个子函数 HAL_UART_MspInit()那里发现了端倪。

MX_USART1_UART_Init()==》HAL_UART_Init()==》HAL_UART_MspInit();

因为我们开启了跟 UART 传输事件相关的 DMA 功能，在 UART_MspInit();函数里不仅有对与 UART 相关的 GPIO 的复用功能配置，还有跟 UART 事件有关的 DMA 配置。看来 UART 的初始化还是跟 DMA 有关联的。【见如下截图】

```
PB10     ------> USART3_TX
PB11     ------> USART3_RX
*/
GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_11;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* USART3 DMA Init */
/* USART3_RX Init */
hdma_usart3_rx.Instance = DMA1_Channel3;
hdma_usart3_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
hdma_usart3_rx.Init.PeriphInc = DMA_PINC_DISABLE;
hdma_usart3_rx.Init.MemInc = DMA_MINC_ENABLE;
hdma_usart3_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE
hdma_usart3_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
hdma_usart3_rx.Init.Mode = DMA_CIRCULAR;
hdma_usart3_rx.Init.Priority = DMA_PRIORITY_LOW;
if (HAL_DMA_Init(&hdma_usart3_rx) != HAL_OK)
{
  Error_Handler();
}

__HAL_LINKDMA(huart,hdmarx,hdma_usart3_rx);
```

结合前面提到的 DMA 初始化函数里的那句开启 DMA 外设时钟代码，到这里基本明白怎么回事了。

因为我们在 UART 初始化代码里要做跟 DMA 有关的配置，如果不事先将 DMA 外设的时钟开启，【UART 初始化函数里也没有开启 DMA 外设时钟的代码】，那么，在 UART 初始化代码进行有关 DMA 的配置操作就没法保证有效。

到此，开篇中提到的因为 DMA 和 UART 初始化代码顺序影响 DMA 功能的原因应该说揭晓了。

在做嵌入式开发过程中，很多的初始化配置都是基于硬件本身的，有些初始化顺序往往有硬件方面的配置时序要求，关于这些各个芯片手册中一般都会有描述和说明。我们在编写初始代码时须遵循相关规定或约定。当然，有些配置顺序可能还得结合具体应用，实际体会后而做灵活调整。

具体到文中案例，本来 STM32CubeMx 在生成初始化代码时已经考虑到初始化时序这点了，只是用户在整理代码过程中无意调整了二者的初始化顺序而不自知，再加上我们对 CubeMx 生成的初始化代码本身往往缺乏足够的熟悉度，在开发过程中可能会因此陷入一时的困扰。关于这点，相信未来版本的 STM32CubeMx 会做进一步的调整与完善。