STM32F2 中 DMA 的 FIFO 模式

问题：

该问题由某客户提出，发生在 STM32F205VET6 器件上。据其工程师讲述：使用 STM32F205 的 DMA 对 USART 的接收进行处理的时候，发现如下现象：如果发送端发送 10 个字节，程序可以正常接收到数据，通过 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)获取的数据长度以及程序中数据接收缓冲区中的数据均是正常的；但是如果发送端只发送 9 个字节，程序就无法正常接收到数据，通过 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)获取的数据长度是正确的，但是在程序中数据接收缓冲区却没有数据，全为 0x00。不解，所以提出帮忙分析。

调研：

检查客户的 DMA 配置程序，配置如下：

```
DMA_InitStructure.DMA_Channel = USARTx_RX_DMA_CHANNEL;
DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)RcvBuffer;
DMA_InitStructure.DMA_PeripheralBaseAddr = USARTx_DR_ADDRESS;
DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralToMemory;
DMA_InitStructure.DMA_BufferSize = BUFFERSIZE;
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_FIFOMode = DMA_FIFOMode_Enable;
DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_Full;
DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
```

其中，BUFFERSIZE = 10。客户的目的就是使用 DMA 从 USARTx 的接收数据寄存器中取值，放入 RcvBuffer 这个数组中，这个数组共有 10 个字节。

uint16_t DMA_GetCurrDataCounter(DMA_Stream_TypeDef* DMAy_Streamx) 函数用来返回当前 DMA 数据流剩下的数据单元个数。用户通过使用 BUFFERSIZE 减去 DMA_GetCurrDataCounter 返回的值来判断已经发送的数据单元个数。

运行程序，验证一下结果，现象如客户所述：如果发送端发送 10 个字节，程序可以正常接收到数据，通过 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)获取的数据长度以及程序中数据接收缓冲区中的数据均是正常的；但是如果发送端只发送 9 个字节，程序就无法正常接收到数据，通过 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)获取的数据长度是正确的，但是在程序中数据接收缓冲区却没有数据，全为 0x00。

仔细观察 DMA 的配置，可以看到这两条配置：

```
DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Enable;
DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_Full;
```

它的意思是打开 DMA 的 FIFO 模式，FIFO 的门限大小为 16 个字节。

修改上述 DMA 配置，如下：

```
DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_Full;
```

重新运行程序进行测试，现象正常如下：如果发送端发送 10 个字节，程序可以正常接收到数据，通过 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)获取的数据长度以及程序中数据接收缓冲区中的数据均是正常的；如果发送端只发送 9 个字节，程序也可以正常接收到数据，通过 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)获取的数据长度是正确的，在程序中数据接收缓冲区的数据也是正确的，前面为接收的 9 个字节的数据，最后一个字节为 0x00。

结论：

由于客户在使用 DMA 的时候打开了 DMA 的 FIFO 模式，并将 FIFO 的门限设置为 16 个字节。当发送端发送 10 个字节时，由于字节数虽然没有达到门限，但是已经达到 DMA 设置的 Buffer Size，DMA 传输完成，FIFO 中的数据被传输到 RcvBuffer；而当发送端发送 9 个字节时，由于这个数值既没有达到 FIFO 的门限，也没有达到 DMA 设置的 Buffer Size，导致 9 个字节的数据仍然停留在 DMA 的 FIFO 中，并没有传输到 RcvBuffer，所以在 RcvBuffer 中就看不到任何接收到的数据。而 DMA_GetCurrDataCounter(USARTx_RX_DMA_STREAM)函数返回的是 USARTx 接收数据流剩下的数据单元个数，因为 9 个字节均已经接收到，自然这个数值就为 1，只不过数据还在 FIFO 中，没有送到 RcvBuffer 罢了。

处理：

不使用 FIFO 模式，直接使用 Direct Mode，问题解决。

建议：

除了循环模式和双缓冲模式，以及流控，STM32F2 系列的 DMA 与 STM32F1 的 DMA 相比，还新增了 FIFO 模式功能。因为 STM32F1 的 DMA 是“Lite”DMA，只支持直接模式，并且未对带宽使用作任何优化。在 STM32F2 上，首先实现双 AHB 主接口，更好地利用总线矩阵和并行传输。另外，为 DMA stream 增加各自的 FIFO，以弥补外设没有 FIFO。每个 stream 拥有各自的 4*32 位的 FIFO。

FIFO 的大小是 4*4 字节，传输的过程为：源地址→AHB 主端口 x→FIFO→AHB 主端口 y→目的地址。是否使用 FIFO 门限，可以区别 FIFO 模式和 Direct 模式。FIFO 常用于 DMA 控制器和 Memories 之间的缓冲。特别是 Memory-to-Memory，是必须使用 FIFO 模式的。

对于 FIFO 模式和 Direct 模式的区别：FIFO 模式——从源地址来的数据先放在 FIFO 中，达到门限后，再根据目的地址的数据宽度送出；Direct 模式——数据放在 FIFO，有 DMA 请求就送走，和门限值无关。默认 DMA 工作在 Direct 模式下，Direct 模式常用于在每次 DMA 请求后都有立即和 Memory 之间单次传输的应用场合。它的缺点是源地址和目的地址的数据宽度必须一致，不能支持突发传输，也不支持 Memory-to-Memory 的传输。而 FIFO 模式中，源地址和目的地址的数据宽度可以不同，分别由 PSIZE 和 MSIZE 指定。当宽度不同时，在 FIFO 中将进行数据的 pack/unpack。另外，可支持突发传输。

使用 FIFO 模式时，会存在剩余“尾巴”数据的问题。什么情况下 FIFO 中会有“尾巴数据”呢？当 NDTR 还没有自减到 0 的情况下，不再有数据进入 FIFO；并且现有数据还未达到 FIFO 门限的时候，将会有“尾巴数据”留在 FIFO 中。此次的问题正是这样的情况。那么尾巴数据怎么处理呢？我们可以通过 FIFO 刷新来解决这一问题，步骤如下：

1. 通过复位 DMA_SxCR 寄存器中的 EN 位来禁止数据流，可以刷新 FIFO。这个时候，DMA 控制器会将剩余的数据继续传输到目的地址中，即使已经有效禁止了数据流。
2. 传送完成后，硬件置位 TCIFx（传输完成标志）。
3. DMA_SxNDTR 中还保持刷新 FIFO 之前的值。
    a. 如果剩余数据少于 MSIZE，DMA 会以 MSIZE 的数量来传送数据。所以，Memory 中会被多写入不需要的数据，因此需要参考 DMA_SxNDTR 来决定哪些才是真正需要的数据。
    b. 如果剩余的数据不足一次 Memory Burst 传输的数据量时，可以使用 Single 传输来完成“尾巴数据”到目的地址的数据传输。