STM32 DMA MEM2MEM.

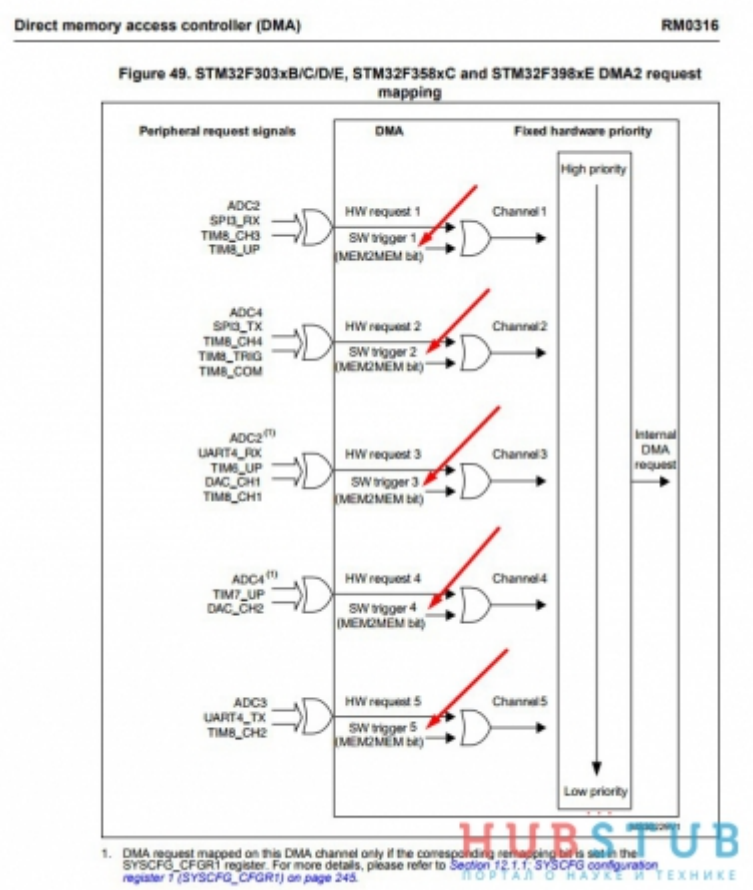
В одной из прошлых статей уже описывал, что такое **DMA** и приводил пример его работы в связке с АЦП. В том примере, результаты преобразования АЦП записывались в буфер без участия ядра. В этот раз мне надо было *перенести буфер из одного участка памяти в другой* и я точно знал, что это можно сделать с помощью DMA, используя режим **MEM2MEM**.

Тот кто работал с **DMA** или читал статью, которую упомянул выше, знает, что при настройке DMA надо:

- выбрать канал DMA
- указать направление передачи данных(из памяти в периферию или из периферию в память)

На самом деле настройка **DMA** содержит в себе больше пунктов, а указал только эти потому, что при их настройке у меня возникли вопросы.

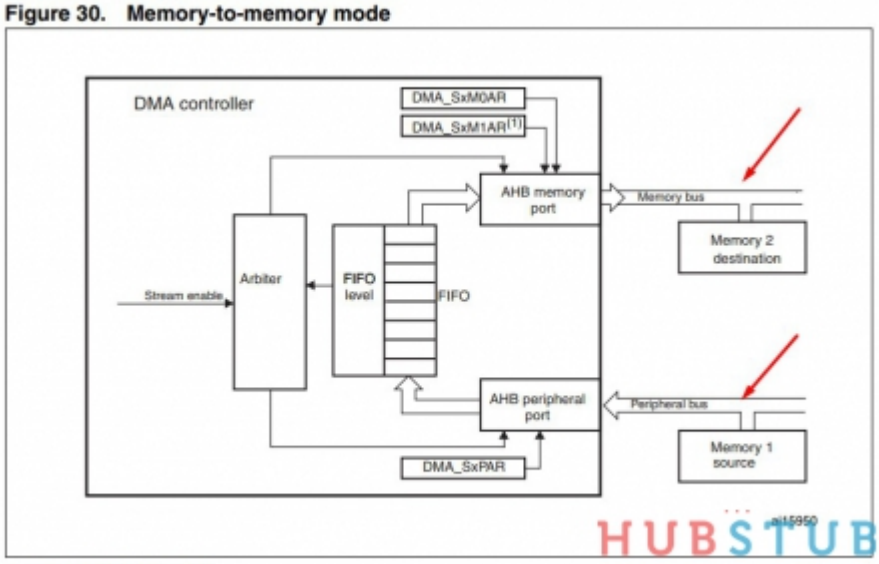
Что касается выбора канала, ответ на этот вопрос отпал при просмотре картинки из **RM**.



На картинке видно, что каждый из каналов DMA2, используемого мной контроллера, поддерживает режим **MEM2MEM**, по этому выбираем любой свободный.

С направлением передачи данных, все оказалось не так просто. В **RM** на мой МК мне не удалось найти нужную картинку, за то она нашлась в **RM** на другой МК.

Figure 30. Memory-to-memory mode



1. For double-buffer mode.

Из картинки понятно, что источник(*Memory Source*) находится на шине **Peripherial Bus**, а получатель(*Memory Destination*) на шине **Memory Bus**. Значит при настройке направления передачи данных выбираем **из периферии в память**. Ну и самое главное не забываем выставить бит **MEM2MEM**.

```
#define DMA_BUFF_SIZE     100
volatile uint16_t buff[DMA_BUFF_SIZE] = {0};
volatile uint16_t old_buff[DMA_BUFF_SIZE] = {0};

void DMA_Copy_Memory_Init(void)
{
        RCC->AHBENR |= RCC_AHBENR_DMA2EN; //Разрешаем тактирование второго DMA
модуля
        DMA2_Channel4->CCR |= DMA_CCR_MEM2MEM; //Включаем режим MEM2MEM
        DMA2_Channel4->CPAR = (uint32_t)buff; //Указываем адрес периферии
        DMA2_Channel4->CMAR = (uint32_t)old_buff; //Указываем адрес в памяти
        DMA2_Channel4->CCR &= ~DMA_CCR_DIR; //Указываем направление передачи
данных, из периферии в память
        DMA2_Channel4->CNDTR = DMA_BUFF_SIZE; //Количество пересылаемых значений
        DMA2_Channel4->CCR |= DMA_CCR_PINC; //Адрес периферии инкрементируем после
каждой пересылки
        DMA2_Channel4->CCR |= DMA_CCR_MINC; //Адрес памяти инкрементируем после
каждой пересылки.
        DMA2_Channel4->CCR |= DMA_CCR_PSIZE_0; //Размерность данных периферии - 16
бит
        DMA2_Channel4->CCR |= DMA_CCR_MSIZE_0; //Размерность данных памяти - 16
бит
        DMA2_Channel4->CCR |= DMA_CCR_PL; //Приоритет - очень высокий
        DMA2_Channel4->CCR &= ~DMA_CCR_CIRC; //Запрещаем работу DMA в циклическом
режиме
        //DMA2_Channel4->CCR |= DMA_CCR_TCIE;//Разрешаем прерывание по окончанию
передачи
        DMA2_Channel4->CCR |= DMA_CCR_EN; //Разрешаем работу 4-го канала DMA
}
```

Также надо отметить, что DMA в режиме **MEM2MEM**, не может работать в циклическом режиме. А для того, чтобы перезапустить **DMA** модуль надо выполнить следующий код.

```
        DMA2_Channel4->CCR &= ~DMA_CCR_EN;//отключаем DMA
        DMA2_Channel4->CNDTR = DMA_BUFF_SIZE;//указываем количество пересылаемых
значений
        DMA2_Channel4->CCR |= DMA_CCR_EN;//запускаем DMA
```