Hi, I'm trying to get an STM32F0 read incoming data packets as a SPI slave using the DMA. I've been reading the DataExchangeDMA peripheral example code and trying to convert some of it to an interrupt driven model, but I have trouble getting the DMA to work correctly. I've boiled down my code to a simple program which should just move incoming SPI data to a buffer in RAM.

To debug, I'm running the code on an STM32F051 discovery board, and using the LED (PC8) to turn on between when the DMA completes a transfer and whenever the next byte comes in over SPI.

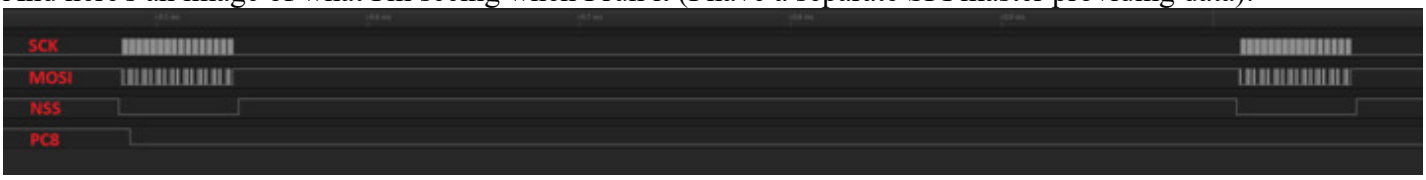
Here's the code I'm working with:

```
001. #include "stm32f0xx.h"
002.
003. #define BUFFER_LEN 16
004.
005. GPIO_InitTypeDef    GPIO_InitStruct;
006. SPI_InitTypeDef     SPI_InitStruct;
007. NVIC_InitTypeDef    NVIC_InitStruct;
008. DMA_InitTypeDef     DMA_InitStruct;
009.
010. void init(void);
011.
012. uint8_t DataBuffer[BUFFER_LEN];
013.
014. int main(void)
015. {
016.     init();
017.
018.     for(;;)
019.     {
020.     }
021. }
022.
023. void init()
024. {
025.     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA | RCC_AHBPeriph_DMA1, ENABLE);
026.     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
027.     RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);
028.
029.     // Discovery LED
030.     GPIO_InitStruct.GPIO_Pin = GPIO_Pin_8;
031.     GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
032.     GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
033.     GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
034.     GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
035.     GPIO_Init(GPIOC, &GPIO_InitStruct);
036.
037.     GPIO_SetBits(GPIOC, GPIO_Pin_8);
038.
039.     // SPI pins
040.     GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
041.     GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF;
042.     GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
043.     GPIO_Init(GPIOA, &GPIO_InitStruct);
044.
045.     GPIO_PinAFConfig(GPIOA, GPIO_PinSource4, GPIO_AF_0);
046.     GPIO_PinAFConfig(GPIOA, GPIO_PinSource5, GPIO_AF_0);
047.     GPIO_PinAFConfig(GPIOA, GPIO_PinSource6, GPIO_AF_0);
048.     GPIO_PinAFConfig(GPIOA, GPIO_PinSource7, GPIO_AF_0);
049.
050.     // SPI peripheral setup
051.     SPI_InitStruct.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
052.     SPI_InitStruct.SPI_Mode = SPI_Mode_Slave;
053.     SPI_InitStruct.SPI_DataSize = SPI_DataSize_8b;
054.     SPI_InitStruct.SPI_CPOL = SPI_CPOL_Low;
055.     SPI_InitStruct.SPI_CPHA = SPI_CPHA_1Edge;
056.     SPI_InitStruct.SPI_NSS = SPI_NSS_Hard;
```

```
057.     SPI_InitStruct.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_2;
058.     SPI_InitStruct.SPI_FirstBit = SPI_FirstBit_MSB;
059.     SPI_InitStruct.SPI_CRCPolynomial = 7;
060.     SPI_Init(SPI1, &SPI_InitStruct);
061.
062.     SPI_RxFIFOThresholdConfig(SPI1, SPI_RxFIFOThreshold_QF);
063.     SPI_Cmd(SPI1, ENABLE);
064.
065.     // SPI interrupt config
066.     NVIC_InitStruct.NVIC_IRQChannel = SPI1_IRQn;
067.     NVIC_InitStruct.NVIC_IRQChannelPriority = 1;
068.     NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
069.     NVIC_Init(&NVIC_InitStruct);
070.
071.     SPI_I2S_ITConfig(SPI1, SPI_I2S_IT_RXNE, ENABLE);
072.
073.     // DMA setup
074.     DMA_InitStruct.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
075.     DMA_InitStruct.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
076.     DMA_InitStruct.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
077.     DMA_InitStruct.DMA_MemoryInc = DMA_MemoryInc_Enable;
078.     DMA_InitStruct.DMA_Mode = DMA_Mode_Normal;
079.     DMA_InitStruct.DMA_M2M = DMA_M2M_Disable;
080.     DMA_InitStruct.DMA_BufferSize = BUFFER_LEN;
081.     DMA_InitStruct.DMA_MemoryBaseAddr = (uint32_t)DataBuffer;
082.     DMA_InitStruct.DMA_DIR = DMA_DIR_PeripheralSRC;
083.     DMA_InitStruct.DMA_Priority = DMA_Priority_High;
084.     DMA_InitStruct.DMA_PeripheralBaseAddr = 0x4001300C; // SPI1 peripheral DR register
address
085.     DMA_Init(DMA1_Channel2, &DMA_InitStruct);
086.
087.     // DMA interrupt config
088.     NVIC_InitStruct.NVIC_IRQChannel = DMA1_Channel2_3_IRQn;
089.     NVIC_InitStruct.NVIC_IRQChannelPriority = 3;
090.     NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
091.     NVIC_Init(&NVIC_InitStruct);
092.
093.     SPI_I2S_DMACmd(SPI1, SPI_I2S_DMAReq_Rx, ENABLE);
094.     DMA_ITConfig(DMA1_Channel2, DMA_IT_TC, ENABLE);
095. }
096.
097. void DMA1_Channel2_3_IRQHandler()
098. {
099.     GPIO_SetBits(GPIOC, GPIO_Pin_8);
100.     if (DMA_GetFlagStatus(DMA1_FLAG_TC2))
101.     {
102.         SPI_I2S_ITConfig(SPI1, SPI_I2S_IT_RXNE, ENABLE);
103.         DMA_Cmd(DMA1_Channel2, DISABLE);
104.         DMA_SetCurrDataCounter(DMA1_Channel2, BUFFER_LEN);
105.         DMA_ClearFlag(DMA1_FLAG_TC2);
106.     }
107. }
108.
109. void SPI1_IRQHandler()
110. {
111.     GPIO_ResetBits(GPIOC, GPIO_Pin_8);
112.     if (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_RXNE))
113.     {
114.         SPI_I2S_ClearFlag(SPI1, SPI_I2S_FLAG_RXNE);
115.         SPI_I2S_ITConfig(SPI1, SPI_I2S_IT_RXNE, DISABLE);
116.
117.         DMA_Cmd(DMA1_Channel2, ENABLE);
118.     }
119. }
120.
121. void assert_failed(const char* file, const int line)
122. {
```

```
123.     for(;;);
124. }
```

And here's an image of what I'm seeing when I run it (I have a separate SPI master providing data):



As you can see, the SPI interrupt is firing correctly after the first byte comes into the Rx buffer, but the DMA interrupt never fires.

Did I miss something with the DMA interrupt configuration? I have similar code driving DMA interrupts on a different application and it's working with no issues. Also, I'm not sure if I need to make the DMA wait and move one byte at a time instead of the whole buffer to prevent it trying to move data that hasn't yet made it to the SPI data register. This would certainly use more CPU cycles which is unfortunate in my application, I just don't know whether it's necessary or not. Either way, I'd still expect the TC interrupt to fire in the code I gave above.

Thanks for any help, and let me know if any additional details would be helpful.