Hi All

HAL_SPI_Receive function has a strange problem, a simple workaround can be implemented as a temporal solution but I need reason of the this behaviour.

```
    address = 0x138001;
//  Ext_Flash_Write_Byte(address, 'A');
//  Ext_Flash_Write_Byte(address+1, 'B');
//  Ext_Flash_Write_Byte(address+2, 'C');
//  Ext_Flash_Write_Byte(address+3, 'D');
//  Ext_Flash_Write_Byte(address+4, 'E');
//  Ext_Flash_Write_Byte(address+5, 'F');
//  Ext_Flash_Write_Byte(address+6, 'G');
//  Ext_Flash_Write_Byte(address+7, 'H');
//  Ext_Flash_Write_Byte(address+8, 'I');
//  Ext_Flash_Write_Byte(address+9, 'J');

    data = Ext_Flash_Read_Byte(address+3);

    Ext_Flash_Read(address, (uint8_t*)SPI_Rx_Buf, 10);
```

10 capital letter is written to External SPI flash memory and tried to read **one byte** and **ten bytes** to show the behaviour. Here is the read functions;

```
uint8_t Ext_Flash_Read_Byte(unsigned int address)
{
    EXT_FLASH_CS_Low();
    SPI_Tx_Buf[0] = CMD_READ;
    SPI_Tx_Buf[1] = ((address >> 16) & 0xFF);
    SPI_Tx_Buf[2] = ((address >> 8) & 0xFF);
    SPI_Tx_Buf[3] = (address & 0xFF);
    HAL_SPI_Transmit(&hspi1, (uint8_t*)SPI_Tx_Buf, 4, 100);
//HAL_SPI_Receive(&hspi1, (uint8_t*)SPI_Rx_Buf, 3, 100);
    HAL_SPI_Receive(&hspi1, (uint8_t*)SPI_Rx_Buf, 1, 100);
    EXT_FLASH_CS_High();

    return SPI_Rx_Buf[0];
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

void Ext_Flash_Read(unsigned int address, uint8_t* pData, uint16_t n)
{
    EXT_FLASH_CS_Low();
    SPI_Tx_Buf[0] = CMD_READ;
    SPI_Tx_Buf[1] = ((address >> 16) & 0xFF);
    SPI_Tx_Buf[2] = ((address >> 8) & 0xFF);
    SPI_Tx_Buf[3] = (address & 0xFF);
    HAL_SPI_Transmit(&hspi1, (uint8_t*)SPI_Tx_Buf, 4, 100);
//HAL_SPI_Receive(&hspi1, (uint8_t*)SPI_Rx_Buf, 3, 100);
    HAL_SPI_Receive(&hspi1, (uint8_t*)pData, n, 100);
    EXT_FLASH_CS_High();
}
```

Here is the result of these functions;

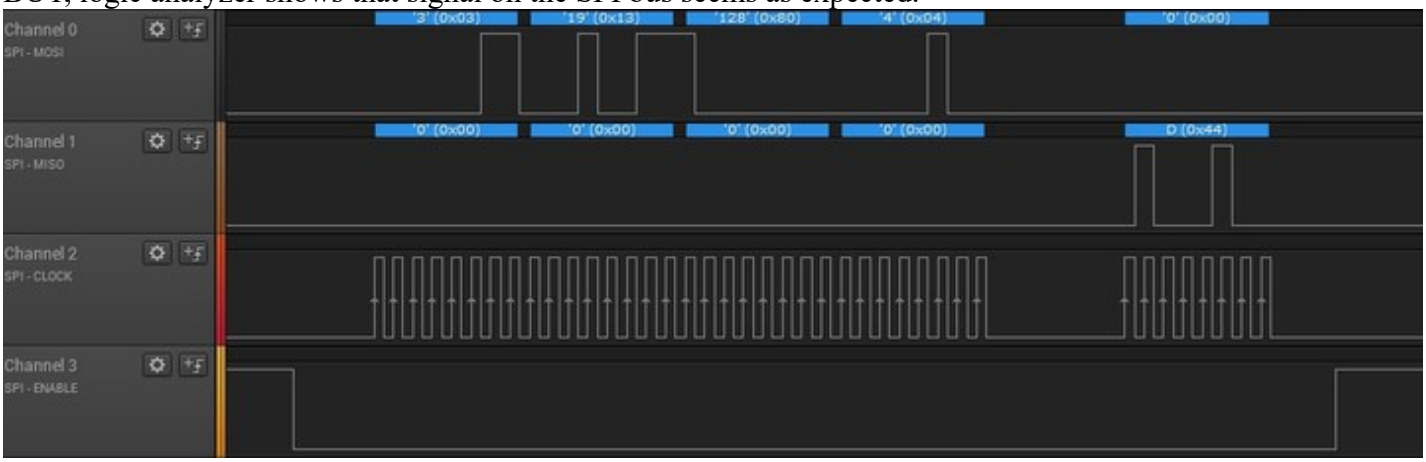
Watch 2

Name	Value	Type
data	0x00	unsigned char
SPI_Rx_Buf	0x20000C20 SPI_Rx_Buf[] "D"	unsigned char[100]
[0]	0x44 'D'	unsigned char
[1]	0x00	unsigned char
[2]	0x00	unsigned char
[3]	0x41 'A'	unsigned char
[4]	0x42 'B'	unsigned char
[5]	0x43 'C'	unsigned char
[6]	0x44 'D'	unsigned char
[7]	0x45 'E'	unsigned char
[8]	0x46 'F'	unsigned char
[9]	0x47 'G'	unsigned char

Call Stack + Locals | Watch 1 | Watch 2 | Memory 1

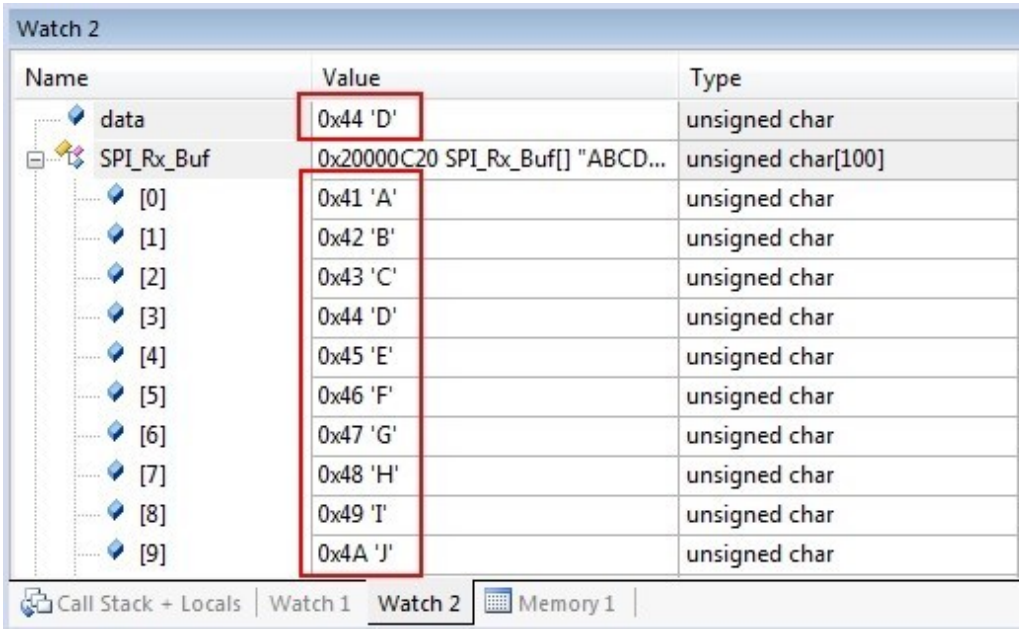
As you can see, single byte read has returned **0x00** instead of **'D'** and multibyte array result has **shifted 3 bytes** and 1st byte is corrupted.

BUT, logic analyzer shows that signal on the SPI bus seems as expected.



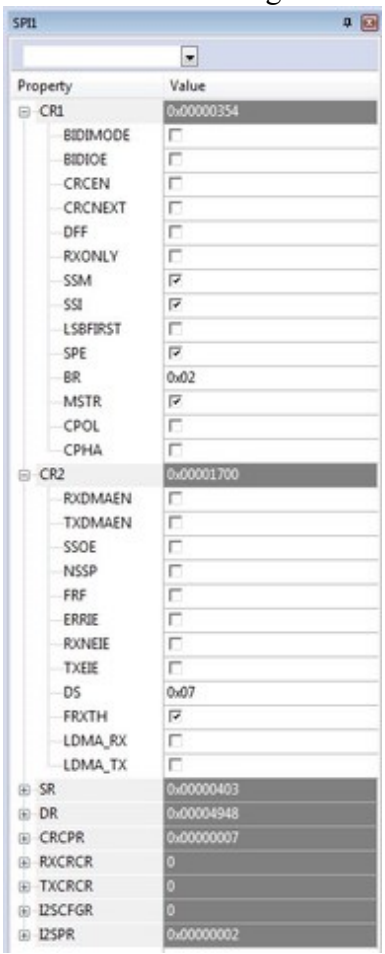
If I removed the **comments** on dummy 3 bytes read everything gets OK !
function HAL_SPI_Receive(&hspi1, (uint8_t*)SPI_Rx_Buf, 3, 100);

Adding 3 extra read sequence clocking bus for extra 3 bytes but it does not a problem for SPI flash, it just continue to out data by increasing its address. There is no any data except the written capital letters, so remaing 3 extra ouput data is 0xFF as expected.



Now it seems OK but why do I need this weird approach ?

Here is the SPI configuration data



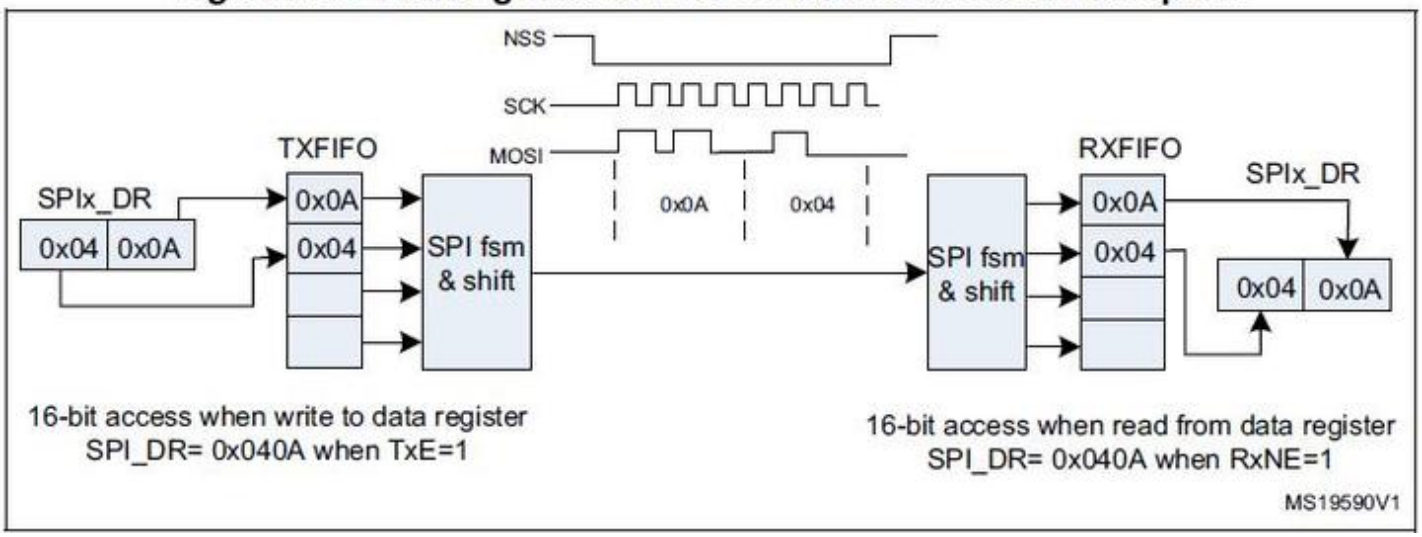
Here is the SPI configuration code;

```
void MX_SPI1_Init(void)
{
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
```

```
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
    Error_Handler();
}

void HAL_SPI_MspInit(SPI_HandleTypeDef* spiHandle)
{
  GPIO_InitTypeDef GPIO_InitStruct;

  if(spiHandle->Instance==SPI1)
  {
    __HAL_RCC_SPI1_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF0_SPI1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
   }
}
```

What is the problem ? Is there a problem in CubeMx function that handling SPI data packaging described below ? Or any point that I am missing ?

Figure 274. Packing data in FIFO for transmission and reception



Best Regards