I am working on a library for controlling the M95128-W EEPROM from an STM32 device. I have the library writing and reading back data however the first byte of each page it not as expected and seems to be fixed at `0x04` .

For example I write 128 bytes across two pages starting at `0x00` address with value `0x80` . When read back I get:

```
     byte[0] = 0x04;
byte[1] = 0x80;
byte[2] = 0x80;
byte[3] = 0x80;
.......
byte[64] = 0x04;
byte[65] = 0x80;
byte[66] = 0x80;
byte[67] = 0x80;
```

I have debugged the SPI with a logic analyzer and confirmed the correct bytes are being sent. When using the logic analyzer on the read command the mysterios `0x04` is transmitted from the EEPROM.

Here is my code:

```
void FLA::write(constvoid* data,unsignedint dataLength,uint16_t address)

{
int pagePos =0;
int pageCount =(dataLength +64-1)/64;
int bytePos =0;int startAddress = address;

while(pagePos < pageCount)
{

         HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2, GPIO_PIN_SET); // WP High
         chipSelect();
         _spi->transfer(INSTRUCTION_WREN);
         chipUnselect();
         uint8_t status = readRegister(INSTRUCTION_RDSR);
         chipSelect();
         _spi->transfer(INSTRUCTION_WRITE);
         uint8_t xlow = address &0xff;
         uint8_t xhigh =(address >>8);
         _spi->transfer(xhigh);// part 1 address MSB
         _spi->transfer(xlow);// part 2 address LSB

         for(unsignedint i =0; i <64&& bytePos < dataLength; i++)
         {
            uint8_t byte =((uint8_t*)data)[bytePos];
            _spi->transfer(byte);
            printConsole("Wrote byte to ");
            printConsoleInt(startAddress + bytePos);
            printConsole("with value ");
            printConsoleInt(byte);
            printConsole("\n");
            bytePos ++;
          }

         _spi->transfer(INSTRUCTION_WRDI);
         chipUnselect();
         HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2, GPIO_PIN_RESET); //WP LOW

         bool writeComplete =false;
         while(writeComplete ==false)
         {
            uint8_t status = readRegister(INSTRUCTION_RDSR);
            if(status&1<<0)
```

```
                    {
                    printConsole("Waiting for write to complete....\n");
                    }

                    else
                    {
                    writeComplete =true;
                    printConsole("Write complete to page ");
                    printConsoleInt(pagePos);
                    printConsole("@ address ");
                    printConsoleInt(bytePos);
                    printConsole("\n");
                    }
            }
            pagePos++;
            address = address +64;
    }

    printConsole("Finished writing all pages total bytes ");
    printConsoleInt(bytePos);
    printConsole("\n");

    }

    void FLA::read(char* returndata,unsignedint dataLength,uint16_t address)
    {
       chipSelect();
       _spi->transfer(INSTRUCTION_READ);
       uint8_t xlow = address &0xff;
       uint8_t xhigh =(address >>8);
       _spi->transfer(xhigh);// part 1 address
       _spi->transfer(xlow);// part 2 address
       for(unsignedint i =0; i < dataLength; i++)
          {
             returndata[i]= _spi->transfer(0x00);
          }
       chipUnselect();
    }
```

Additionally I have tried writing sequentially 255 bytes increasing data to check for rollover. The results are as follows:

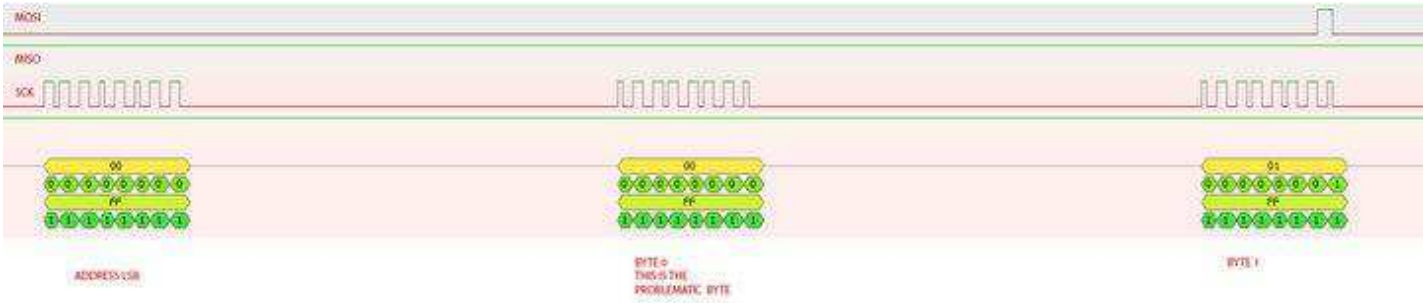
```
      byte[0] = 4; // Incorrect Mystery Byte
byte[1] = 1;  byte[2] = 2;
byte[3] = 3;
.......
byte[63] = 63;
byte[64] = 4; // Incorrect Mystery Byte
byte[65] = 65;
byte[66] = 66;
.......
byte[127] = 127;
byte[128] = 4; // Incorrect Mystery Byte
byte[129} = 129;
```

Pattern continues. I have also tried writing just 8 bytes from address 0x00 and the same problem persists so I think we can rule out rollover.

I have tried removing the debug printConsole and it has had no effect.

Here is a SPI logic trace of the write command:

And a close up of the first byte that is not working correctly:



Code can be viewed on gitlab here:

[Src/flash.cpp · master · Daniel Beyzade / stm32f107vc-home-control-master · GitLab]

Init code of SPI can be seen here in MX_SPI_Init()

[Src/main.cpp · master · Daniel Beyzade / stm32f107vc-home-control-master · GitLab]

I have another device on the SPI bus (RFM69HW RF Module) which works as expected sending and receiving data.

Any suggestion or help appreciated.

Daniel