How to use STM32 DMA & SPI for WIZnet

Version 1.0.0

For more information, visit our website at http://www.wiznet.io

Table of Contents

1 STM32F10x DMA

The DMA of the STM32F10x is connected as shown in the figure below, which can confirm in the Reference manual of the STM32F10x.

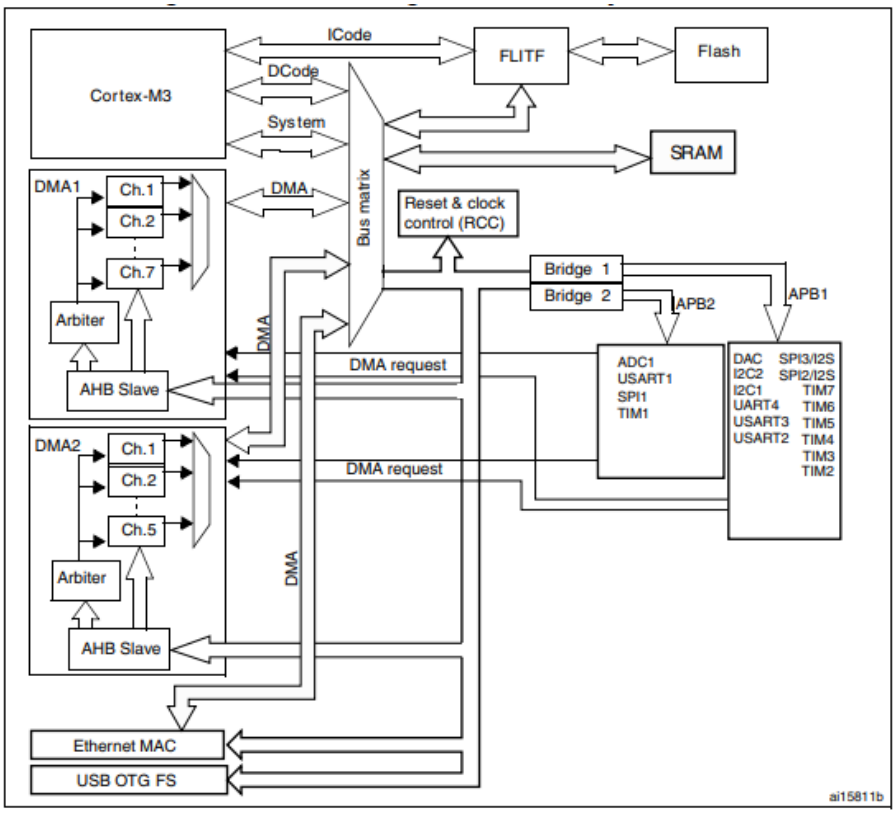


Figure 1 DMA block diagram in connectivity line devices

When SPI DMA mode use, the improved speed can be confirmed because the data transmission and reception in units of blocks are continuously performed.

The SPI1 and SPI2 to be used are allocated to DMA1.

A SPI1 uses Channel2 and Channel 3 of DMA1 and A SPI2 uses channel4 and channel 5 of DMA2.

Peripheral	Channel2	Channel3	Channel4	Channel5
SPI	**SPI1_RX**	**SPI1_TX**	**SPI2_RX**	**SPI2_TX**

Figure 2 Summary of DMA1 requests for each channel

In case of W5100S EVB, it is connected to SPI2. If you want to use SPI1, it can be tested by separately connecting modules that can connect Ethernet Chip such as Ethernet Shield.

The pin of the SPI1 and SPI2 is shown in Figure 3 SPI1,SPI2 Pin configuration.

The pin of the BUS is shown in Figure 4 Bus pin configuration.

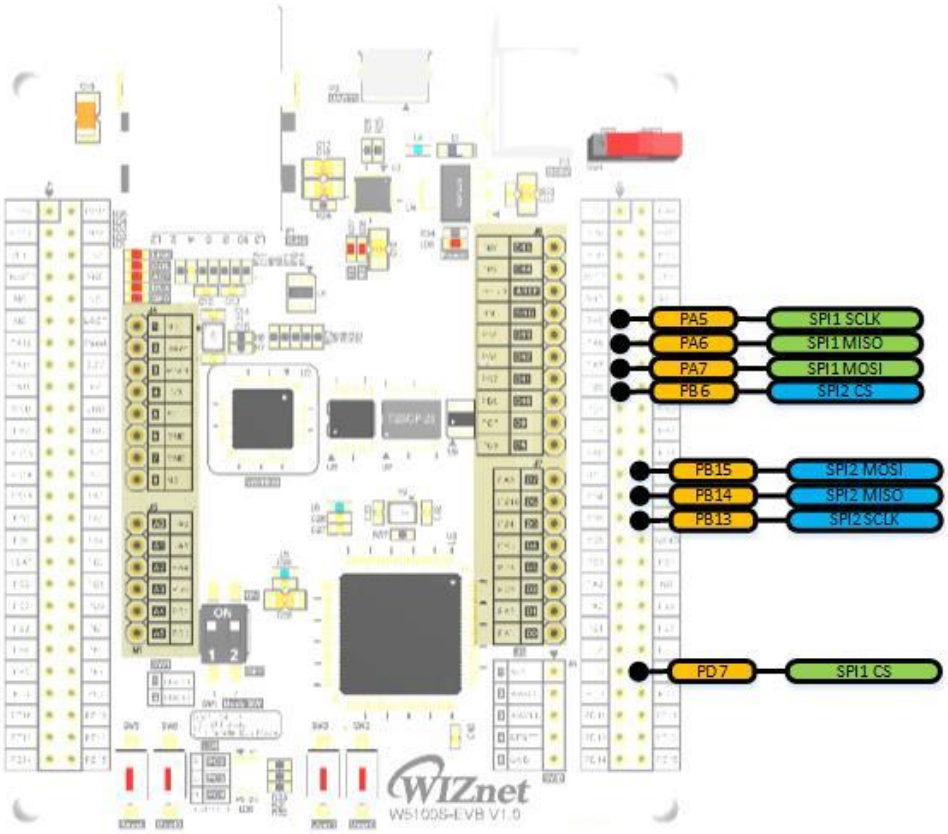


Figure 3 SPI1,SPI2 Pin configuration

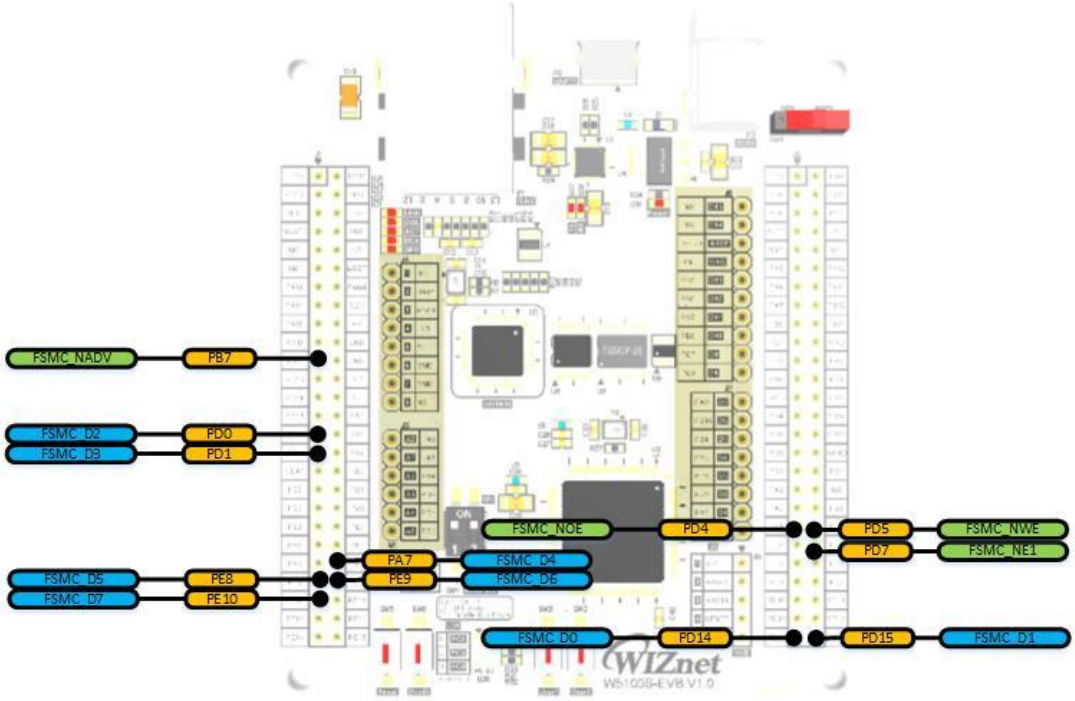


Figure 4 Bus pin configuration

2 SPI Frame

W5100S has two kind of SPI Frame. It can transmit the data using the frame of W5100 SPI or W5500 SPI and it can modify the SPI frame by value of MOD[0].

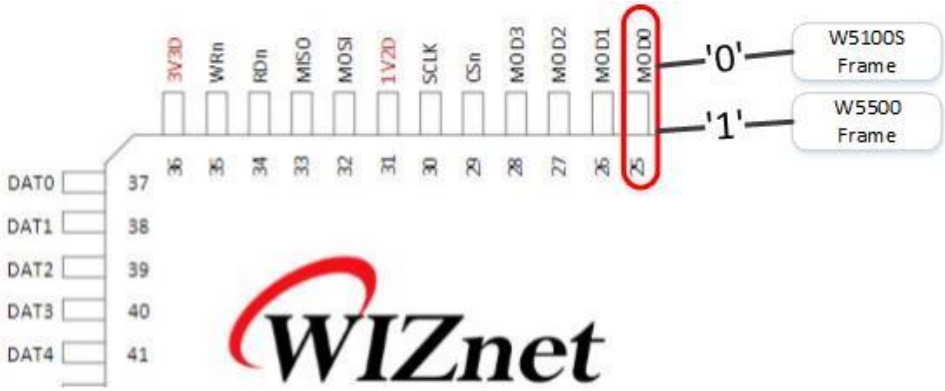


Figure 5 Pin Layout

The difference between the W5100S and the W5500 SPI Frame is that the position of the Control Phase changes as shown in the figure below.

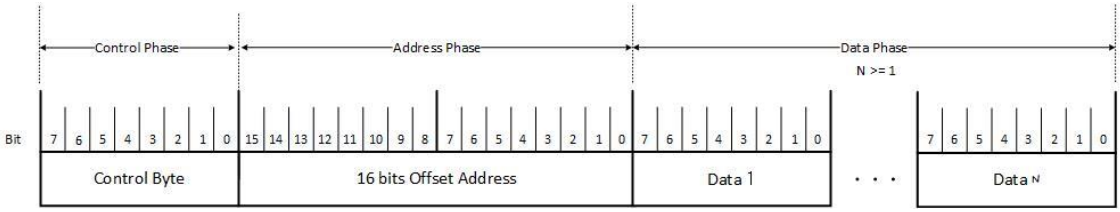


Figure 6 W5100S SPI Frame

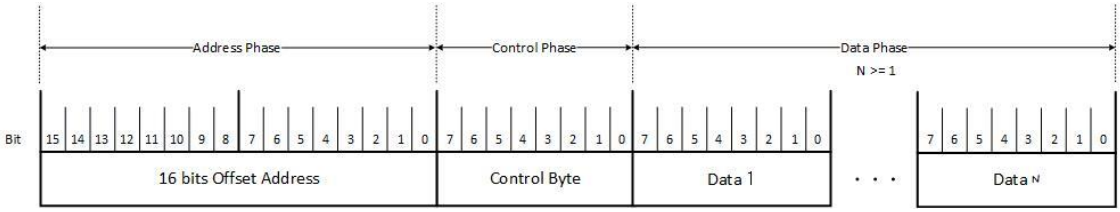


Figure 7 W5500 SPI Frame

2.1 BUS/SPI Setting for DMA

2.1.1 Clock Configuration

The maximum clock of System used in the W5100S EVB is 72MHz and the clock of AHB same the system clock. The maximum clock of APB1 and APB2 is 36MHz and 72MHz, respectively.

Table 1 STM32103 Max Clock Configuration

	Max Clock(MHz)	etc
SYSCLK	72	
AHB CLK	72	FSMC
APB1 CLK	36	SPI2
APB2 CLK	72	SPI1

The clock configuration is shown in the figure below.

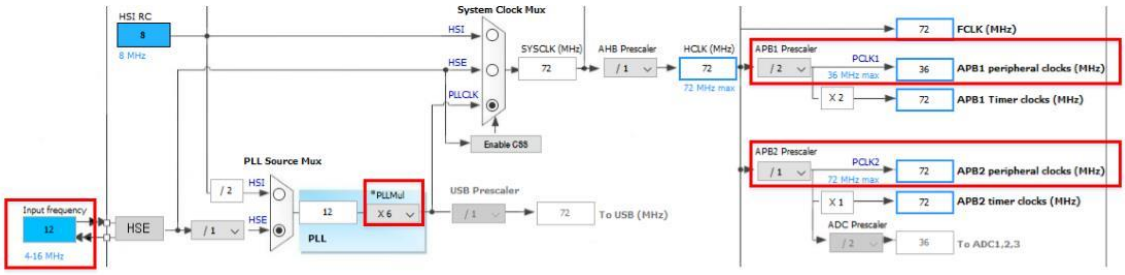


Figure 8 Clock Configuration

The default HSE_VALUE is 12MHz or 8MHz. If you use another external oscillator, you must change the stm32f10x.h and system_stm32f10x.c. SYSCLK is calculated as HSE_VALUE * PLL_MUL. For the W5100S EVB, change the HSE_VALUE value to 12MHz and the PLL_MUL value to 6 because the external oscillator is 12MHz.

```
#if !defined  HSE_VALUE
 #ifdef STM32F10X_CL
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Value of the External oscillator in Hz */
 #else
  #define HSE_VALUE    ((uint32_t)12000000) /*!< Value of the External oscillator in Hz */
 #endif /* STM32F10X_CL */
#endif /* HSE_VALUE */
```

Figure 9 stm32f10x.h

In System_stm32f103x.c, it changes the PLLMULL value of RCC's CFGR to RCC_CFGR_PLLMULL6.

```
    /*  PLL configuration: PLLCLK = HSE * RCC_CFGR_PLLMULL = 72 MHz */
    RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE |
                                        RCC_CFGR_PLLMULL));

    RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSE | RCC_CFGR_PLLMULL6);
```

Figure 10 system_stm32f10x.c

APB CLOCK is basically set as below.

```
/* HCLK = SYSCLK */
RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;

/* PCLK2 = HCLK */
RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV1;

/* PCLK1 = HCLK */
RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE1_DIV2;
```

Figure 11 Clock Prescaler in system_stm32f10x.c

2.1.2 BUS/SPI flow for DMA

In the case of DMA, DMA RX and TX should be executed simultaneously even when one of Read and Write operations is performed. Also if DMA_Cmd is used, RX of DMA channel must be enabled first. After DMA_Cmd, wait until RX, TX FLAG is enables and DMA should be disabled. A simple explanation is shown the flowchart below.

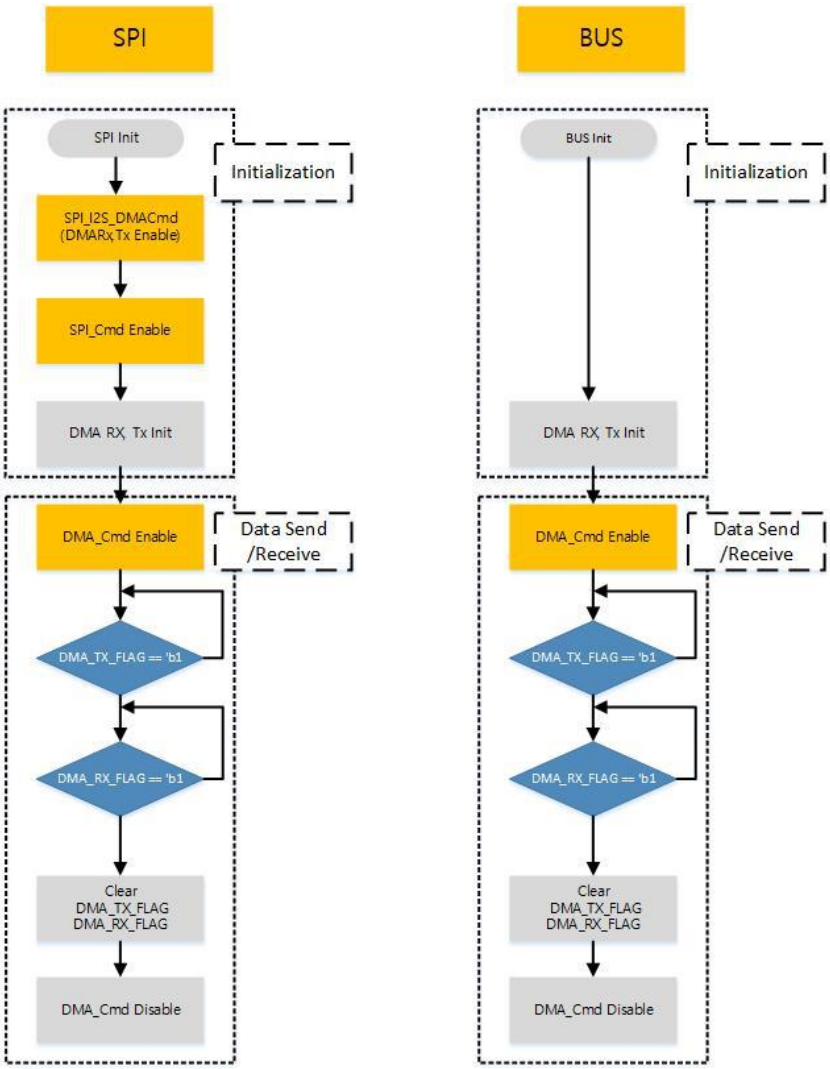


Figure 12 SPI or BUS using the DMA

2.1.3 SPI DMA

2.1.3.1 Callback function

Register the function that you want to use as SPI in the callback function.

```
reg_wizchip_spi_cbfunc(spiReadByte, spiWriteByte);
reg_wizchip_spiburst_cbfunc(spiReadBurst, spiWriteBurst);
```

2.1.3.2 SPI Initialization function

SPI Maximum Clock is 1/2 of APB Clock. As described above, SPI1 Clock is 36MHz and SPI2 Clock is 18MHz is Maximum Clock.

```
SPI_InitTypeDef SPI_InitStructure;
SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_High;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;
SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_2;
SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
SPI_InitStructure.SPI_CRCPolynomial = 7;
/* Initializes the SPI communication */
SPI_InitStructure.SPI_Mode = SPI_Mode_Master;

SPI_Init(SPI2, &SPI_InitStructure);
SPI_I2S_DMACmd(SPI2, SPI_I2S_DMAReq_Rx | SPI_I2S_DMAReq_Tx , ENABLE);
SPI_Cmd(SPI2,ENABLE);
```

2.1.3.3 DMA Initialization function

```
/* DMA SPI RX Channel */
DMA_RX_InitStructure.DMA_BufferSize = 0; //default
DMA_RX_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
DMA_RX_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_RX_InitStructure.DMA_MemoryBaseAddr = 0;
DMA_RX_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_RX_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_RX_InitStructure.DMA_Mode = DMA_Mode_Normal;

DMA_RX_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)(&(SPI->DR));
DMA_RX_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_RX_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
DMA_RX_InitStructure.DMA_Priority = DMA_Priority_High;

DMA_Init(DMA_CHANNEL_RX, &DMA_RX_InitStructure);



/* DMA SPI TX Channel */
DMA_TX_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
                          ⋮
Same like DMA_RX          ⋮

DMA_Init(DMA_CHANNEL_TX, &DMA_TX_InitStructure);
```

2.1.3.4 SPI ReadBurst function

SPI ReadBurst and SPI WriteBurst have the same structure.

```
uint8_t spiReadBurst(uint8_t* pBuf, uint16_t len)
{
        DMA_TX_InitStructure.DMA_BufferSize = len;
        DMA_TX_InitStructure.DMA_MemoryBaseAddr = pBuf;
        DMA_Init(W5100S_DMA_CHANNEL_TX, &DMA_TX_InitStructure);

        DMA_RX_InitStructure.DMA_BufferSize = len;
        DMA_RX_InitStructure.DMA_MemoryBaseAddr = pBuf;
        DMA_Init(DMA_CHANNEL_RX, &DMA_RX_InitStructure);
        /* Enable SPI Rx/Tx DMA Request*/
        DMA_Cmd(DMA_CHANNEL_RX, ENABLE);
        DMA_Cmd(DMA_CHANNEL_TX, ENABLE);
        /* Waiting for the end of Data Transfer */
        while(DMA_GetFlagStatus(DMA_TX_FLAG) == RESET);
        while(DMA_GetFlagStatus(DMA_RX_FLAG) == RESET);

        DMA_ClearFlag(DMA_TX_FLAG | DMA_RX_FLAG);

        DMA_Cmd(DMA_CHANNEL_TX, DISABLE);
        DMA_Cmd(DMA_CHANNEL_RX, DISABLE);

}
```

2.1.4 Indirect Bus

2.1.4.1 Callback function

Register the function that you want to use as Bus in Callback function.

```
reg_wizchip_bus_cbfunc(busReadByte, busWriteByte);
reg_wizchip_busburst_cbfunc(busReadBurst, busWriteBurst);
```

2.1.4.2 BUS Initialization function

BUS used the AHB clock.

```
FSMC_NORSRAMInitTypeDef  FSMC_NORSRAMInitStructure;
FSMC_NORSRAMTimingInitTypeDef  p;

FSMC_NORSRAMCmd(FSMC_Bank1_NORSRAM1, DISABLE);
p.FSMC_AddressSetupTime = 0x03;
p.FSMC_AddressHoldTime = 0x01;
p.FSMC_DataSetupTime = 0x08;
p.FSMC_BusTurnAroundDuration = 0;
p.FSMC_CLKDivision = 0x00;
p.FSMC_DataLatency = 0;
p.FSMC_AccessMode = FSMC_AccessMode_B;
FSMC_NORSRAMInitStructure.FSMC_Bank = FSMC_Bank1_NORSRAM1;
FSMC_NORSRAMInitStructure.FSMC_DataAddressMux = FSMC_DataAddressMux_Enable;
FSMC_NORSRAMInitStructure.FSMC_MemoryType = FSMC_MemoryType_NOR;
FSMC_NORSRAMInitStructure.FSMC_MemoryDataWidth = FSMC_MemoryDataWidth_8b;
FSMC_NORSRAMInitStructure.FSMC_BurstAccessMode = FSMC_BurstAccessMode_Disable;
FSMC_NORSRAMInitStructure.FSMC_WaitSignalPolarity = FSMC_WaitSignalPolarity_Low;
FSMC_NORSRAMInitStructure.FSMC_WrapMode = FSMC_WrapMode_Disable;
FSMC_NORSRAMInitStructure.FSMC_WaitSignalActive =
FSMC_WaitSignalActive_BeforeWaitState;
FSMC_NORSRAMInitStructure.FSMC_WriteOperation = FSMC_WriteOperation_Enable;
FSMC_NORSRAMInitStructure.FSMC_WaitSignal = FSMC_WaitSignal_Disable;
FSMC_NORSRAMInitStructure.FSMC_ExtendedMode = FSMC_ExtendedMode_Disable;
FSMC_NORSRAMInitStructure.FSMC_WriteBurst = FSMC_WriteBurst_Disable;
FSMC_NORSRAMInitStructure.FSMC_ReadWriteTimingStruct = &p;
FSMC_NORSRAMInitStructure.FSMC_WriteTimingStruct = &p;
FSMC_NORSRAMInitStructure.FSMC_AsynchronousWait = FSMC_AsynchronousWait_Disable;
FSMC_NORSRAMInit(&FSMC_NORSRAMInitStructure);
        /*!< Enable FSMC Bank1_SRAM1 Bank */
FSMC_NORSRAMCmd(FSMC_Bank1_NORSRAM1, ENABLE);
```

2.1.4.3 DMA Initialization function

```
/* DMA MEM TX Channel */
DMA_TX_InitStructure.DMA_BufferSize = 0; //default
DMA_TX_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
DMA_TX_InitStructure.DMA_M2M = DMA_M2M_Enable;
DMA_TX_InitStructure.DMA_MemoryBaseAddr = 0;
DMA_TX_InitStructure.DMA_PeripheralBaseAddr =0;
DMA_TX_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_TX_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_TX_InitStructure.DMA_PeripheralInc  = DMA_PeripheralInc_Enable;
DMA_TX_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
DMA_TX_InitStructure.DMA_Mode = DMA_Mode_Normal;
DMA_Init(W5100S_DMA_CHANNEL_TX, &DMA_TX_InitStructure);

/* DMA MEM RX Channel */
Same like DMA_TX
```

2.1.4.4 BUS ReadBurst function

Except for the below syntax, BUS ReadBurst and BUS WriteBurst have the same structure.

The BUS Read Burst example is shown below.

```
busReadBurst
        DMA_RX_InitStructure.DMA_MemoryBaseAddr =pBuf;
        DMA_RX_InitStructure.DMA_PeripheralBaseAddr =addr;
busWriteBurst
        DMA_RX_InitStructure.DMA_MemoryBaseAddr = addr;
        DMA_RX_InitStructure.DMA_PeripheralBaseAddr = pBuf;
```

```
void busReadBurst(uint32_t addr,uint8_t* pBuf, uint32_t len)
{
        DMA_RX_InitStructure.DMA_BufferSize = len;
        DMA_RX_InitStructure.DMA_MemoryBaseAddr =pBuf;
        DMA_RX_InitStructure.DMA_PeripheralBaseAddr =addr;
        DMA_Init(W5100S_DMA_CHANNEL_RX, &DMA_RX_InitStructure);
        DMA_Cmd(W5100S_DMA_CHANNEL_RX, ENABLE);
        /* Waiting for the end of Data Transfer */
        while(DMA_GetFlagStatus(DMA_RX_FLAG) == RESET);
        DMA_ClearFlag(DMA_RX_FLAG);
        DMA_Cmd(W5100S_DMA_CHANNEL_RX, DISABLE);
}
```

3 Test Result

Environment configuration

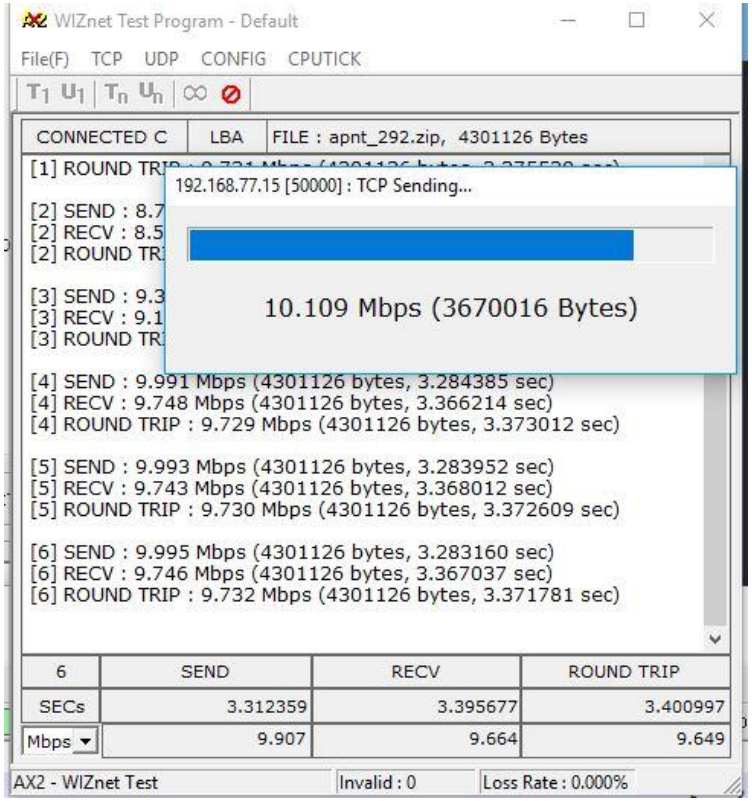
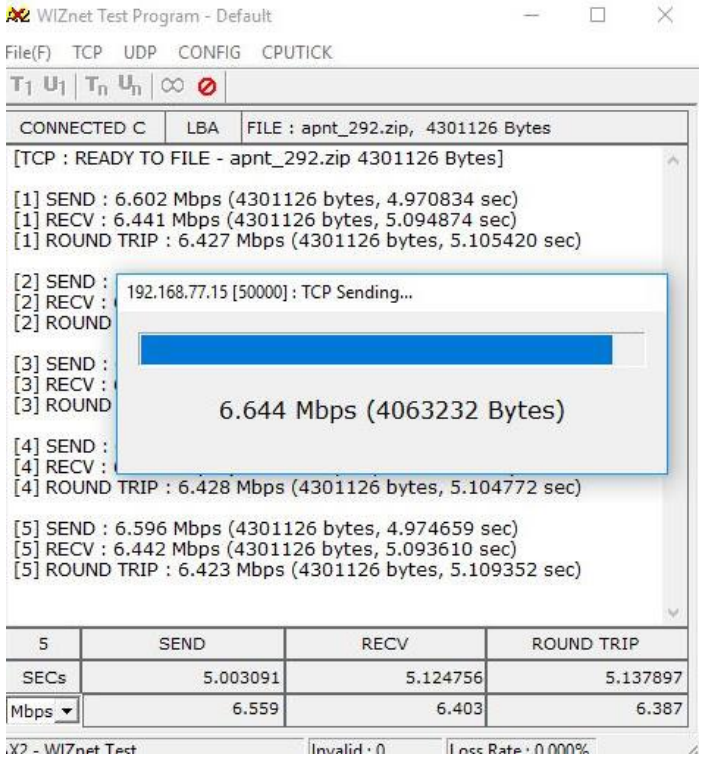
- Socket configuration: Tx/RX 8k , No delay Ack
- Test board : W5100S EVB v1.0
- Compiler : Atollic TrueSTUDIO v9.0.1

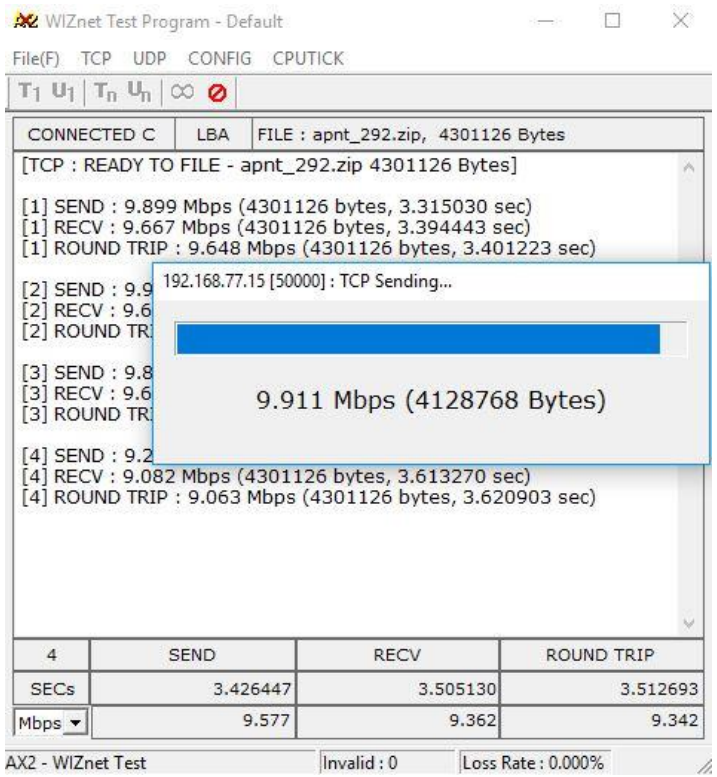
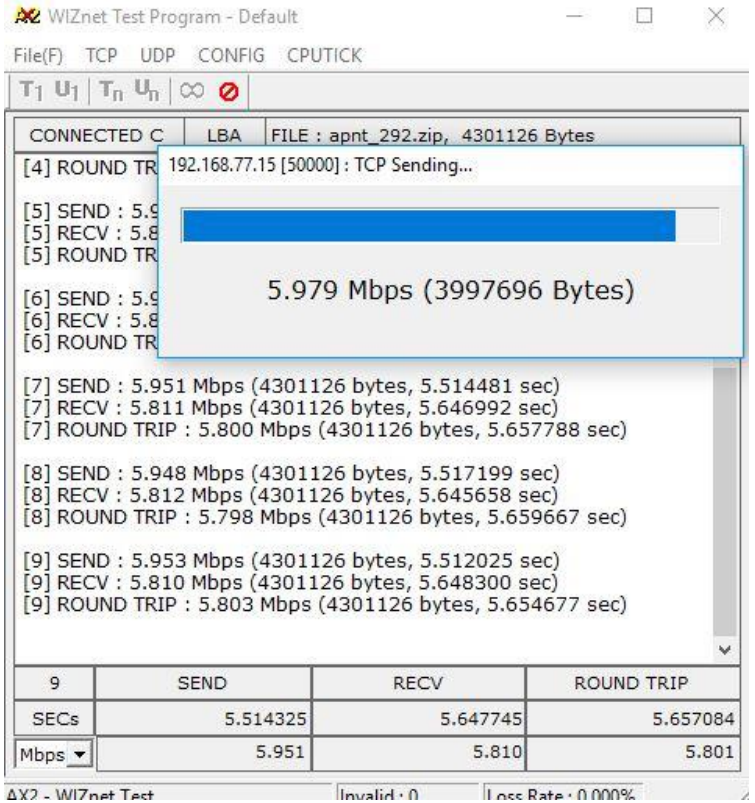
Table 2 Test Result

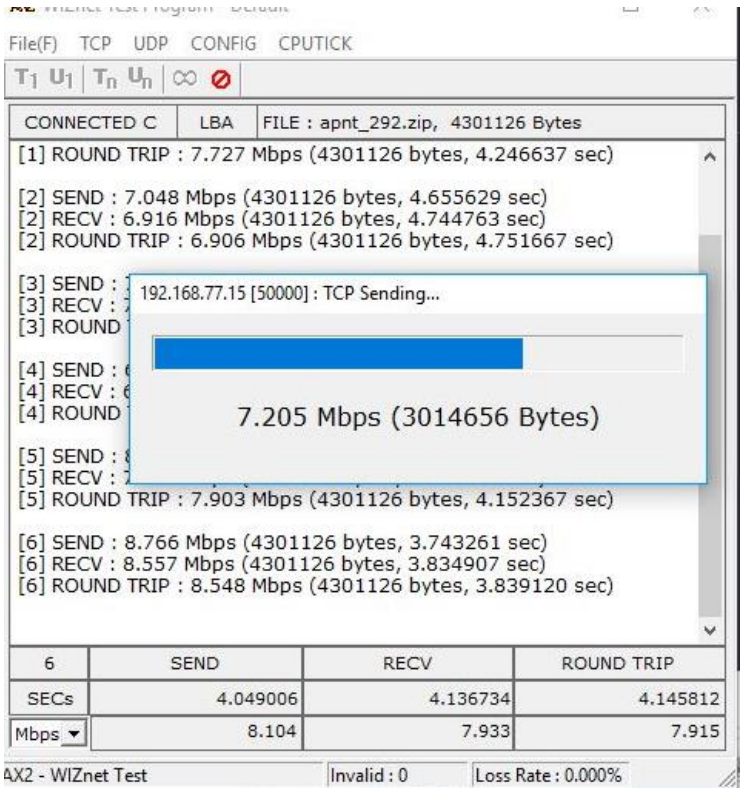
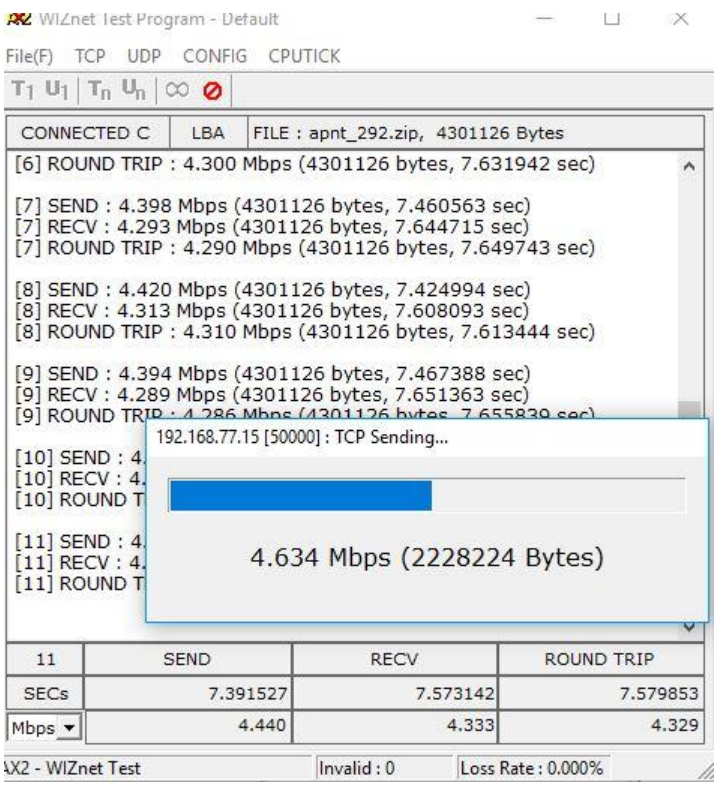
SYSCLK	APB Clock	Peripheral	SPI Clock	DMA	Frame Format	TX(Max)	RX(Max)
72MHz	**APB2 72MHz**	**SPI1**	36MHz	No DMA	W5100S Frame	1.6Mbps	1.5Mpbs
					W5500 Frame	1.6Mbps	1.5Mpbs
				DMA	W5100S Frame	5.3Mbps	4.8Mbps
					W5500 Frame	5.3Mbps	4.8Mbps
	APB1 36MHz	**SPI2**	18MHz	No DMA	W5100S Frame	1.5Mbps	1.4Mpbs
					W5500 Frame		
				DMA	W5100S Frame	4.4Mbps	4.3Mbps
					W5500 Frame		
SYCLK	**AHB Clock**	**Peripheral**		**DMA**		**Tx(Max)**	**RX(Max)**
72MHz	**72MHz**	**FSMC**		No DMA		3.8Mbps	3.7Mbps
				DMA		9.9Mbps	9.6Mbps

This table shows the test using W5100S_EVB, and the absence of W5500 Frame in SPI2 is because MOD0 is fixed to '0' in W5100S EVB. However, it didn't find any difference when tested with two frames using the SPI1.

In case of SPI and BUS the test using the DMA, BUS is about twice faster than SPI.
See Resources for more details on the speed of testing.

	Socket Size	BUS	SPI
DMA	8K		

	4K	 WIZnet Test Program - Default File(F) TCP UDP CONFIG CPUTICK CONNECTED C LBA FILE : apnt_292.zip, 4301126 Bytes [TCP : READY TO FILE - apnt_292.zip 4301126 Bytes] [1] SEND : 9.899 Mbps (4301126 bytes, 3.315030 sec) [1] RECV : 9.667 Mbps (4301126 bytes, 3.394443 sec) [1] ROUND TRIP : 9.648 Mbps (4301126 bytes, 3.401223 sec) 192.168.77.15 [50000] : TCP Sending... 9.911 Mbps (4128768 Bytes) [4] RECV : 9.082 Mbps (4301126 bytes, 3.613270 sec) [4] ROUND TRIP : 9.063 Mbps (4301126 bytes, 3.620903 sec) 4 SEND RECV ROUND TRIP SECs 3.426447 3.505130 3.512693 Mbps 9.577 9.362 9.342 AX2 - WIZnet Test Invalid : 0 Loss Rate : 0.000%	 WIZnet Test Program - Default File(F) TCP UDP CONFIG CPUTICK CONNECTED C LBA FILE : apnt_292.zip, 4301126 Bytes 192.168.77.15 [50000] : TCP Sending... 5.979 Mbps (3997696 Bytes) [7] SEND : 5.951 Mbps (4301126 bytes, 5.514481 sec) [7] RECV : 5.811 Mbps (4301126 bytes, 5.646992 sec) [7] ROUND TRIP : 5.800 Mbps (4301126 bytes, 5.657788 sec) [8] SEND : 5.948 Mbps (4301126 bytes, 5.517199 sec) [8] RECV : 5.812 Mbps (4301126 bytes, 5.645658 sec) [8] ROUND TRIP : 5.798 Mbps (4301126 bytes, 5.659667 sec) [9] SEND : 5.953 Mbps (4301126 bytes, 5.512025 sec) [9] RECV : 5.810 Mbps (4301126 bytes, 5.648300 sec) [9] ROUND TRIP : 5.803 Mbps (4301126 bytes, 5.654677 sec) 9 SEND RECV ROUND TRIP SECs 5.514325 5.647745 5.657084 Mbps 5.951 5.810 5.801

	2K	 File(F) TCP UDP CONFIG CPUTICK CONNECTED C LBA FILE : apnt_292.zip, 4301126 Bytes [1] ROUND TRIP : 7.727 Mbps (4301126 bytes, 4.246637 sec) [2] SEND : 7.048 Mbps (4301126 bytes, 4.655629 sec) [2] RECV : 6.916 Mbps (4301126 bytes, 4.744763 sec) [2] ROUND TRIP : 6.906 Mbps (4301126 bytes, 4.751667 sec) 192.168.77.15 [50000] : TCP Sending... 7.205 Mbps (3014656 Bytes) [5] ROUND TRIP : 7.903 Mbps (4301126 bytes, 4.152367 sec) [6] SEND : 8.766 Mbps (4301126 bytes, 3.743261 sec) [6] RECV : 8.557 Mbps (4301126 bytes, 3.834907 sec) [6] ROUND TRIP : 8.548 Mbps (4301126 bytes, 3.839120 sec) 6 SEND RECV ROUND TRIP SECs 4.049006 4.136734 4.145812 Mbps 8.104 7.933 7.915 AX2 - WIZnet Test Invalid : 0 Loss Rate : 0.000%	 WIZnet Test Program - Default File(F) TCP UDP CONFIG CPUTICK CONNECTED C LBA FILE : apnt_292.zip, 4301126 Bytes [6] ROUND TRIP : 4.300 Mbps (4301126 bytes, 7.631942 sec) [7] SEND : 4.398 Mbps (4301126 bytes, 7.460563 sec) [7] RECV : 4.293 Mbps (4301126 bytes, 7.644715 sec) [7] ROUND TRIP : 4.290 Mbps (4301126 bytes, 7.649743 sec) [8] SEND : 4.420 Mbps (4301126 bytes, 7.424994 sec) [8] RECV : 4.313 Mbps (4301126 bytes, 7.608093 sec) [8] ROUND TRIP : 4.310 Mbps (4301126 bytes, 7.613444 sec) [9] SEND : 4.394 Mbps (4301126 bytes, 7.467388 sec) [9] RECV : 4.289 Mbps (4301126 bytes, 7.651363 sec) 192.168.77.15 [50000] : TCP Sending... 4.634 Mbps (2228224 Bytes) 11 SEND RECV ROUND TRIP SECs 7.391527 7.573142 7.579853 Mbps 4.440 4.333 4.329 AX2 - WIZnet Test Invalid : 0 Loss Rate : 0.000%

4 Document History Information

Version	Date	Descriptions
Ver. 1.0.0	Apr, 2019	Release

Copyright Notice

Technical Support: forum.wiznet.io or support@wiznet.io
Sales & Distribution: sales@wiznet.io

For more information, visit our website at www.wiznet.io