

# How to use STM32 DMA & SPI for WIZnet

Version 1.0.0



© 2018 WIZnet Co., Ltd. All Rights Reserved.

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

## Table of Contents

1	<b>STM32F10x DMA</b>	3
2	<b>SPI Frame</b>	5
2.1	<b>DMA 및 BUS/SPI Setting</b>	5
2.1.1	<b>Clock Configuration</b>	5
2.1.2	<b>DMA 및 BUS/SPI 순서도</b>	7
2.1.3	<b>SPI DMA</b>	9
2.1.3.1	<b>Callback 함수</b>	9
2.1.3.2	<b>SPI Initialization 함수</b>	9
2.1.3.3	<b>DMA Initialization 함수</b>	10
2.1.3.4	<b>SPI ReadBurst 함수</b>	10
2.1.4	<b>Indirect Bus</b>	12
2.1.4.1	<b>Callback 함수</b>	12
2.1.4.2	<b>BUS Initialization 함수</b>	13
2.1.4.3	<b>DMA Initialization 함수</b>	14
2.1.4.4	<b>BUS ReadBurst 함수</b>	14
3	<b>테스트 결과</b>	16
4	<b>Document History Information</b>	1

## 1 STM32F10x DMA

STM32F10x의 DMA는 아래의 그림과 같이 연결되어 있으며, 이는 STM32F10x의 Reference manual에서 확인할 수 있다.

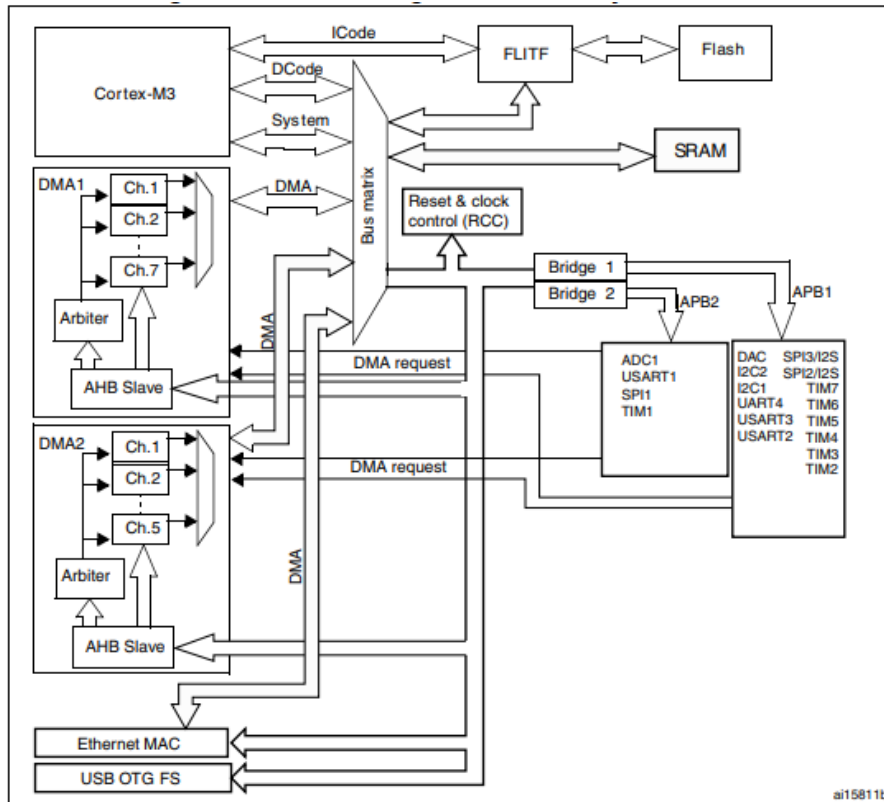


Figure 1 DMA block diagram in connectivity line devices

SPI DMA모드를 사용했을 때, 블록단위의 데이터 송수신이 연속적으로 이루어지기 때문에, 개선된 속도를 확인할 수 있다.

DMA1, 2가 있으며, 사용하고자 하는 SPI1, SPI2는 DMA1에 할당되어 있다.

SPI1은 DMA1의 Channel2, Channel3을 SPI2는 DMA2의 Channel4,5를 사용한다.

Peripheral	Channel2	Channel3	Channel4	Channel5
SPI	SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX

Figure 2 Summary of DMA1 requests for each channel

W5100S EVB의 경우는 SPI2에 연결 되어있으며, SPI1을 사용하고자 한다면, Ethernet Shield와 같은 별도의 Ethernet Chip을 연결할 수 있는 모듈 등을 별도로 연결해서 테스트 가능하다.

SPI1, SPI2의 Pin의 Figure 3 SPI1,SPI2 Pin configuration 를 참고하면 된다.

BUS 의 Pin은 Figure 4 Bus pin configuration를 참고하면 된다.

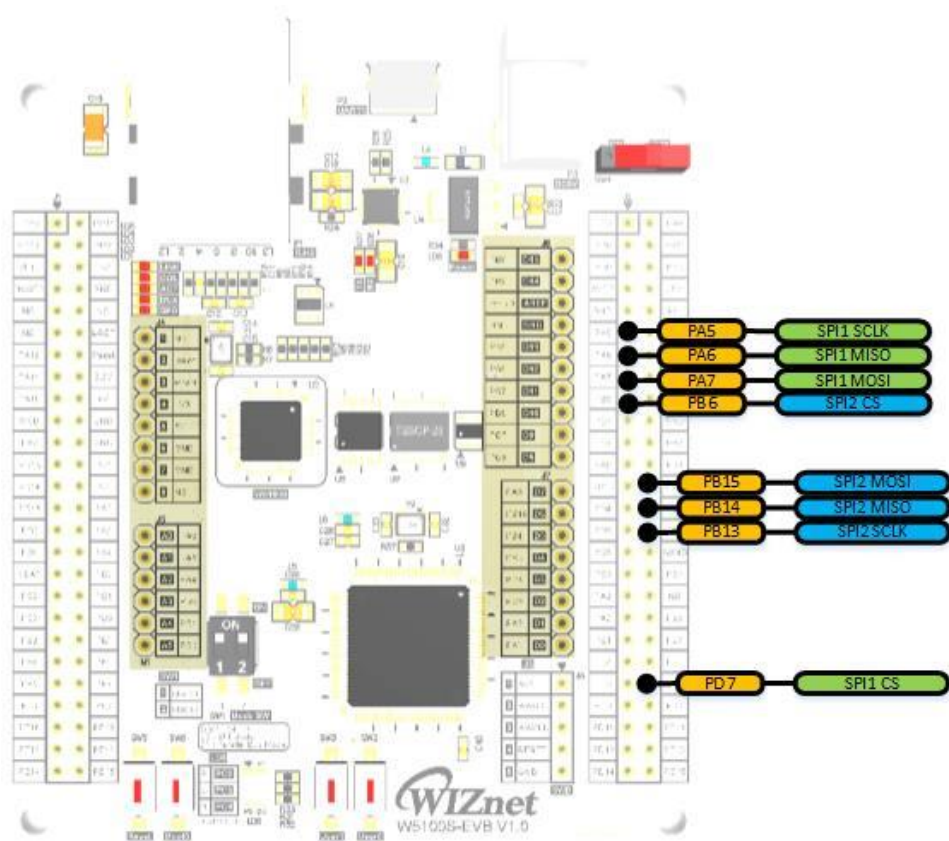


Figure 3 SPI1,SPI2 Pin configuration

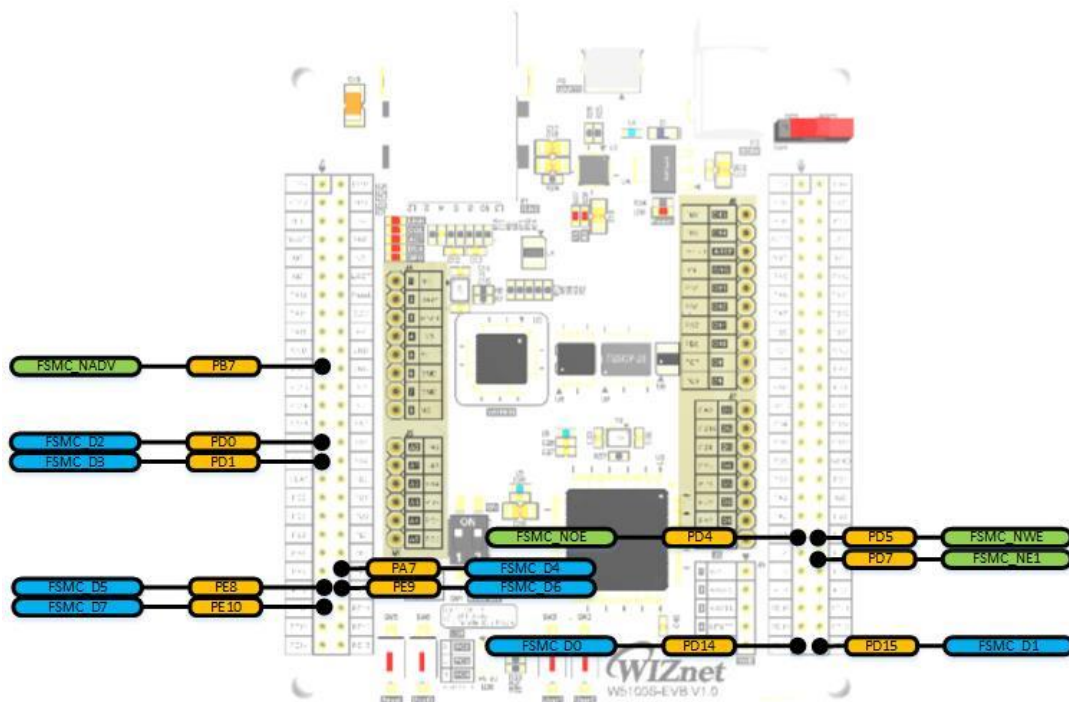


Figure 4 Bus pin configuration

## 2 SPI Frame

W5100S 는 2 개의 SPI Frame 을 가지고 있다.

W5100 SPI, W5500 SPI 의 Frame 으로 데이터를 전송할 수 있으며, SPI Frame 의 선택은 MOD[0]에 따라 변경 가능하다.



Figure 5 Pin Layout

W5100S 와 W5500 SPI Frame 의 차이는 아래의 그림과 같이 Control Phase 의 위치에 따른 차이이다.

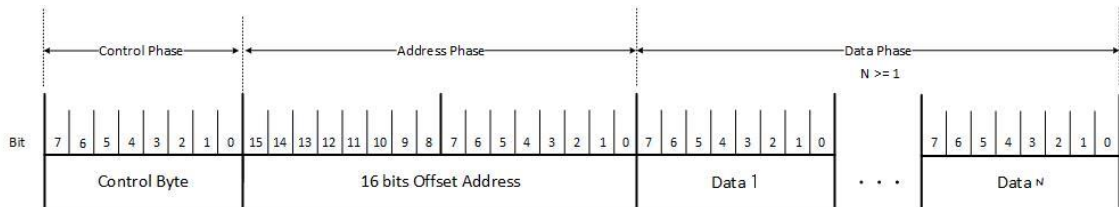


Figure 6 W5100S SPI Frame

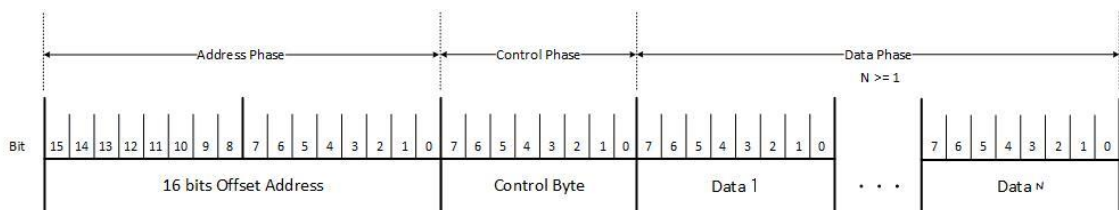


Figure 7 W5500 SPI Frame

## 2.1 DMA 및 BUS/SPI Setting

### 2.1.1 Clock Configuration

W5100S EVB에 사용되는 MCU는 system clock의 Max clock 은 72MHz이며, AHB는 system clock과 동일

값으로 사용된다. APB1와 APB2의 Max clock은 각각 36MHz, 72MHz이다.

Table 1 STM32103 Max Clock Configuration

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

Clock Configuration은 아래 그림에서 확인하면 된다.

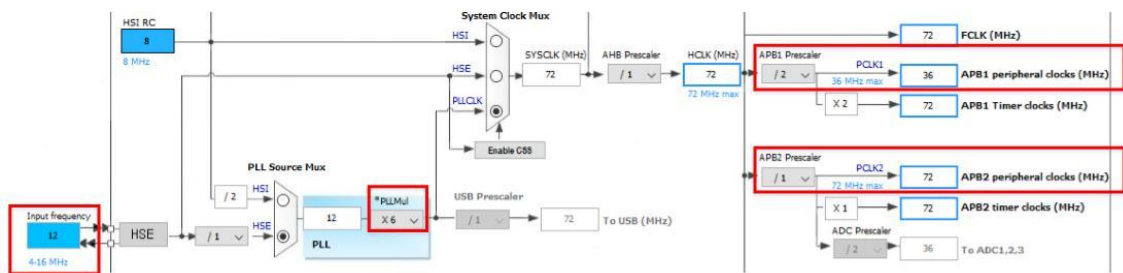


Figure 8 Clock Configuration

기본 HSE\_VALUE는 25MHz or 8MHz로 되어있으며, 다른 External oscillator의 값을 사용한다면 기본으로 주어지는 stm32f10x.h 와 system\_stm32f10x.c를 변경해야 한다.

SYSCCLK는 HSE\_VALUE \* PLL\_MUL로 계산되어진다.

W5100S EVB의 경우 External oscillator가 12MHz이기 때문에 HSE\_VALUE 값은 12MHz, PLL\_MUL값은 6으로 변경한다.

```
#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

System\_stm32f103x.c에서 RCC의 CFGR의 PLLMULL값을 RCC\_CFGR\_PLLMULL6로 변경한다.

```
/* 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 CLOCK은 기본으로 아래와 같이 설정되어 있다.

```
/* 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 DMA 및 BUS/SPI 순서도

DMA 경우 Read 나 Write 하나의 동작 할 경우라도 DMA RX, TX 동시에 수행된다.  
또한, DMA\_Cmd 를 사용할 경우 DMA Channel 중 RX 를 먼저 Enable 시켜줘야 한다.  
DMA\_Cmd 이후 RX, TX FLAG 가 뜰때까지 기다린 후 DMA 를 Disable 해야 한다.  
간단한 설명은 아래의 순서도를 이해하면 된다.

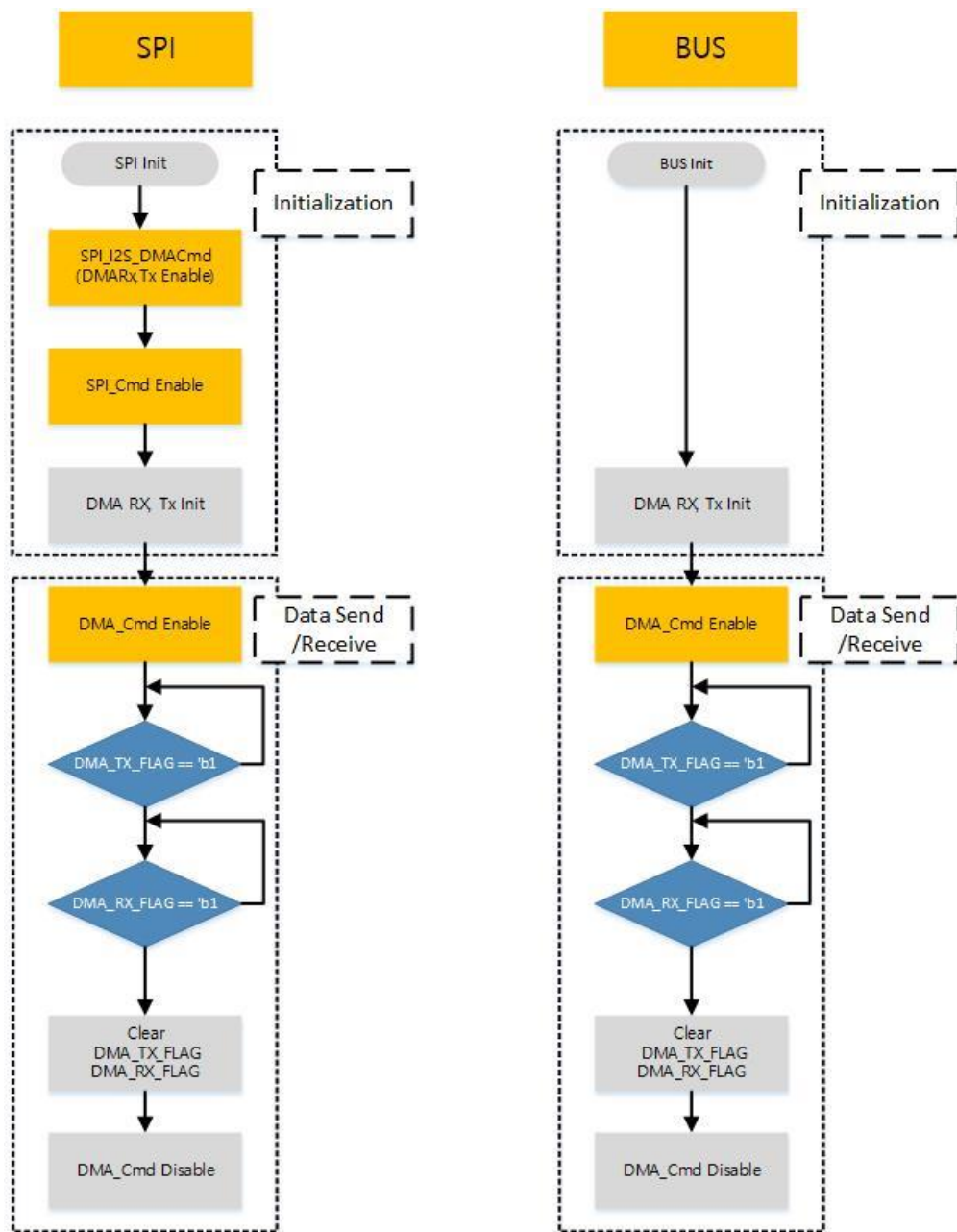


Figure 12 DMA, SPI 및 버스 동작



## 2.1.3 SPI DMA

### 2.1.3.1 Callback 함수

Callback 함수에 SPI 로 사용하고 싶은 함수를 등록하여 사용한다.

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

### 2.1.3.2 SPI Initialization 함수

SPI Max Clock 은 APB Clock 의 1/2 이며, 앞서 설명할 것처럼 SPI1 Clock 는 36MHz,  
SPI2 Clock 은 18MHz 가 Max Clock 이 된다.

```
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 함수

```

/* 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;
DMA_TX_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_TX_InitStructure.DMA_MemoryBaseAddr = 0;
DMA_TX_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_TX_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_TX_InitStructure.DMA_Mode = DMA_Mode_Normal;
DMA_TX_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(SPI->DR);
DMA_TX_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_TX_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
DMA_TX_InitStructure.DMA_Priority = DMA_Priority_High;

DMA_Init(DMA_CHANNEL_TX, &DMA_TX_InitStructure);

```

### 2.1.3.4 SPI ReadBurst 함수

SPI ReadBurst 와 SPI WriteBurst는 동일한 구조를 가지고 있다.

```
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 함수

Callback 함수에 Bus 로 사용하고 싶은 함수를 등록하여 사용한다.

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

## 2.1.4.2 BUS Initialization 함수

BUS 는 AHB Clock 을 사용한다.

```
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 함수

```

/* 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 */
DMA_TX와 동일

```

### 2.1.4.4 BUS ReadBurst 함수

아래의 구문의 제외하고는 BUS ReadBurst 와 BUS WriteBurst 는 동일한 구조를 가지고 있으며, 예는 BUS Read Burst 이다.

```

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 테스트 결과

#### 환경구성

- Socket 구성 : Tx/RX 8k , No delay Ack
- 테스트 보드 : W5100S EVB v1.0
- 컴파일러 : Atollic TrueSTUDIO v9.0.1

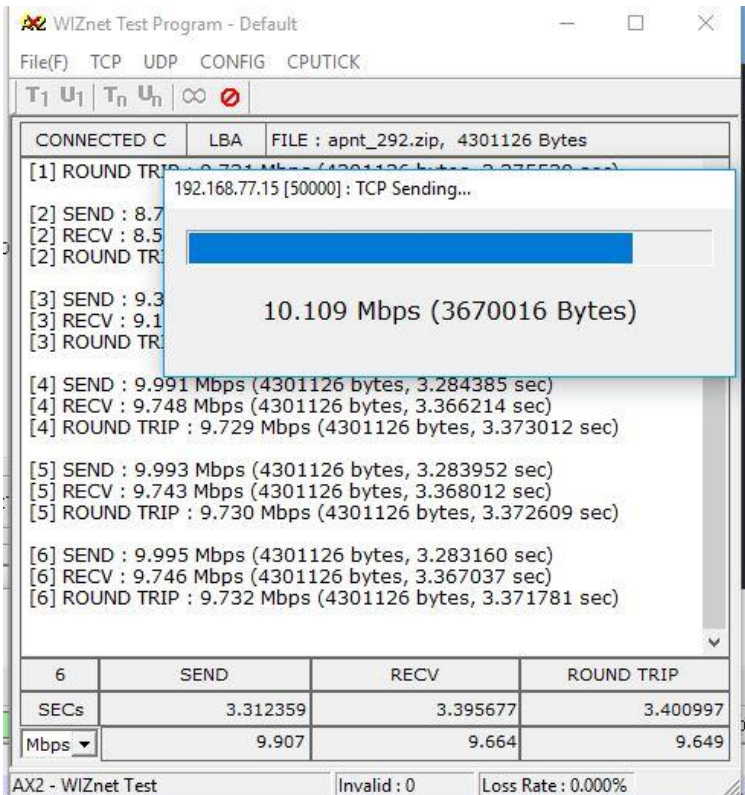
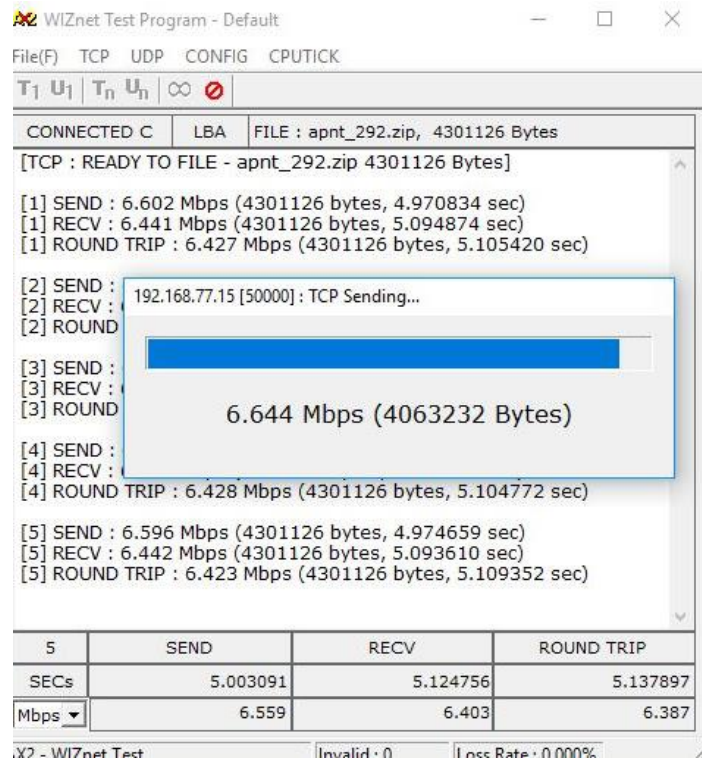
Table 2 Test Result

SYSCLK	APB Clock	Peripheral	SPI Clock	DMA	Frame Format	TX(Max)	RX(Max)
72MHz	APB2 72MHz	SPI1	36MHz	No DMA	W5100S Frame	1.6Mbps	1.5Mbps
					W5500 Frame	1.6Mbps	1.5Mbps
				DMA	W5100S Frame	5.3Mbps	4.8Mbps
					W5500 Frame	5.3Mbps	4.8Mbps
	APB1 36MHz	SPI2	18MHz	No DMA	W5100S Frame	1.5Mbps	1.4Mbps
					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

해당 표는 W5100S\_EVB 를 사용하여 테스트를 진행한 것이며, SPI2 에 W5500 Frame 의 내용 없는 것은 W5100S EVB 에 MOD0 가 '0'으로 고정되어 있기 때문이다. 그러나, SPI1 에서는 두가지의 Frame 을 사용하여 테스트 한 경우 차이 점을 찾지 못했다.

SPI, BUS 모두 DMA 를 사용하였을 때 SPI 보다 BUS 가 약 2 배이상 빠른 것을 볼 수 있다. 자세한 테스트 속도를 확인하려면 참고 자료를 보시면 됩니다.



	Socket Size	BUS	SPI
DMA	8K		

4K

WIZnet Test Program - Default

File(F) TCP UDP CONFIG CPUTICK

T<sub>1</sub> U<sub>1</sub> T<sub>n</sub> U<sub>n</sub> ∞ 0

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)		
[2] SEND : 9.911 Mbps (4128768 Bytes)		
[2] RECV : 9.667 Mbps (4301126 bytes, 3.394443 sec)		
[2] ROUND TRIP : 9.648 Mbps (4301126 bytes, 3.401223 sec)		
[3] SEND : 9.899 Mbps (4301126 bytes, 3.315030 sec)		
[3] RECV : 9.667 Mbps (4301126 bytes, 3.394443 sec)		
[3] ROUND TRIP : 9.648 Mbps (4301126 bytes, 3.401223 sec)		
[4] SEND : 9.200 Mbps (4301126 bytes, 3.613270 sec)		
[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

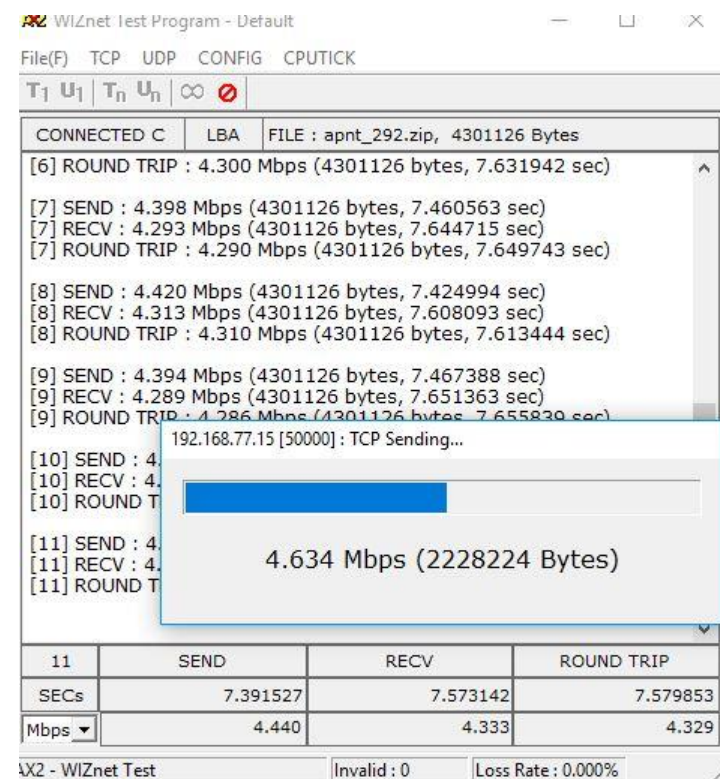
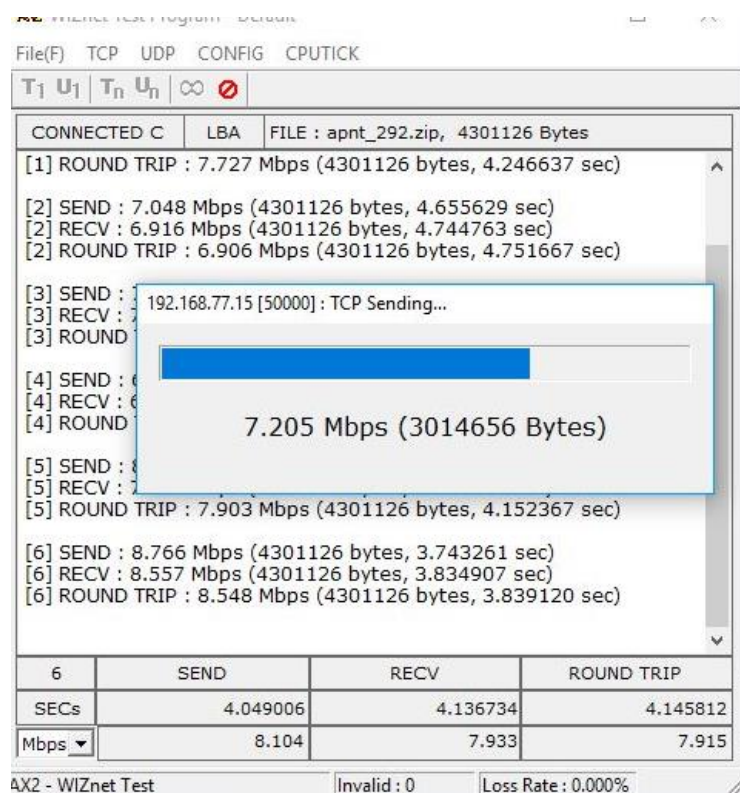
T<sub>1</sub> U<sub>1</sub> T<sub>n</sub> U<sub>n</sub> ∞ 0

CONNECTED C	LBA	FILE : apnt_292.zip, 4301126 Bytes
[4] ROUND TRIP : 192.168.77.15 [50000] : TCP Sending...		
[5] SEND : 5.979 Mbps (3997696 Bytes)		
[5] RECV : 5.810 Mbps (4301126 bytes, 5.648300 sec)		
[5] ROUND TRIP : 5.803 Mbps (4301126 bytes, 5.654677 sec)		
[6] SEND : 5.951 Mbps (4301126 bytes, 5.514481 sec)		
[6] RECV : 5.811 Mbps (4301126 bytes, 5.646992 sec)		
[6] ROUND TRIP : 5.800 Mbps (4301126 bytes, 5.657788 sec)		
[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

AX2 - WIZnet Test Invalid : 0 Loss Rate : 0.000%

2K



## 4 Document History Information

Version	Date	Descriptions
Ver. 1.0.0	Sep, 2018	Release

## Copyright Notice

Copyright 2018 WIZnet Co.,Ltd. All Rights Reserved.

Technical Support: [forum.wiznet.io](http://forum.wiznet.io) or [support@wiznet.io](mailto:support@wiznet.io)

Sales & Distribution: [sales@wiznet.io](mailto:sales@wiznet.io)

For more information, visit our website at [www.wiznet.io](http://www.wiznet.io)