**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](http://www.wiznet.io/)

**Table of Contents**

[1 STM32F10x DMA 4](#_Toc530056886)

[2 SPI Frame 5](#_Toc530056887)

[2.1 DMA 및 SPI Setting 6](#_Toc530056888)

[2.1.1 Clock Configuration 6](#_Toc530056889)

[2.1.2 DMA, SPI 순서도 7](#_Toc530056890)

[2.1.2.1 Callback 함수 9](#_Toc530056891)

[2.1.2.2 SPI Initialization 함수 9](#_Toc530056892)

[2.1.2.3 DMA Initialization 함수 10](#_Toc530056893)

[2.1.2.4 SPI ReadBurst 함수 10](#_Toc530056894)

[3 테스트 결과 12](#_Toc530056895)

[4 Document History Information 13](#_Toc530056896)

1. STM32F10x DMA

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

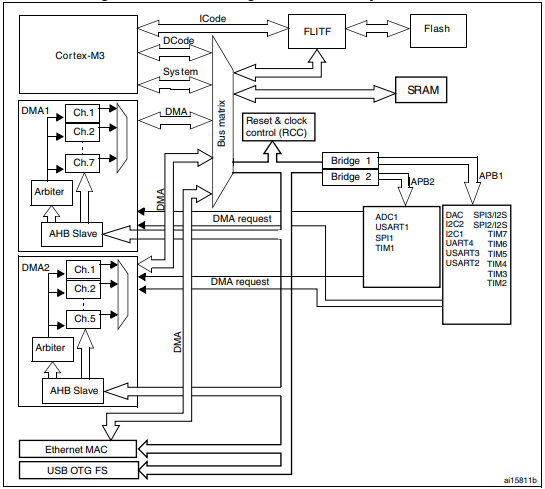


Figure 1 DMA block diagram in connectivity line devices

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 configration 를 참고하면 된다.

BUS 의 Pin은 Figure 4 Bus pin configration를 참고하면 된다.

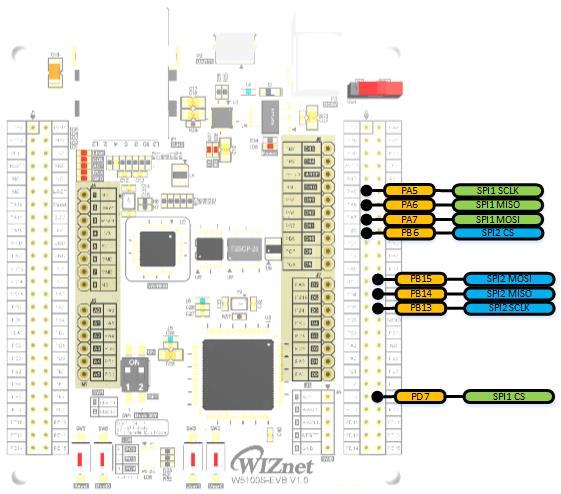


Figure 3 SPI1,SPI2 Pin configration

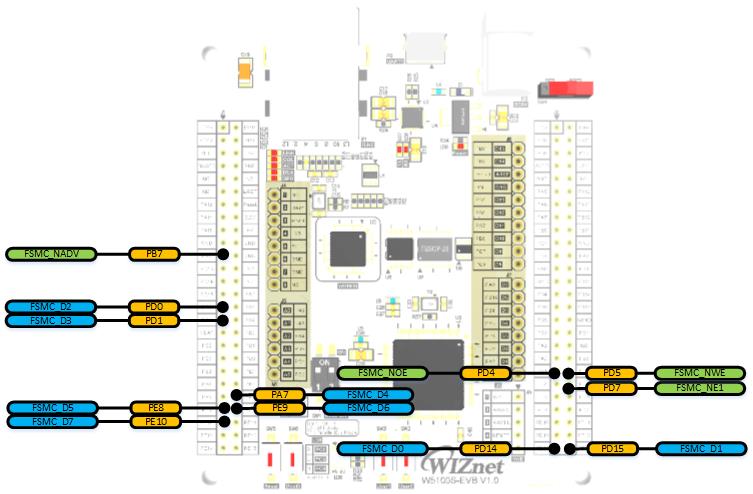


Figure Bus pin configration

1. SPI Frame

W5100S는 2개의 SPI Frame을 가지고 있다.

W5100 SPI, W5500 SPI의 Frame으로 데이터를 전송할 수 있으며, SPI Frame의 선택은 MOD[0]에 따라 변경 가능하다.

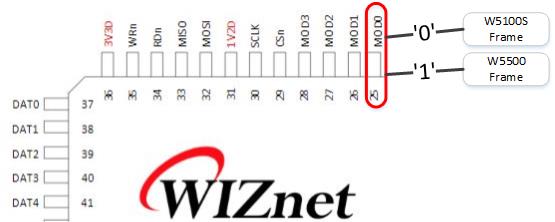


Figure 5 Pin Layout

W5100S와 W5500 SPI Frame의 차이는 아래의 그림과 같이 Control Phase의 위치에 따른 차이다.

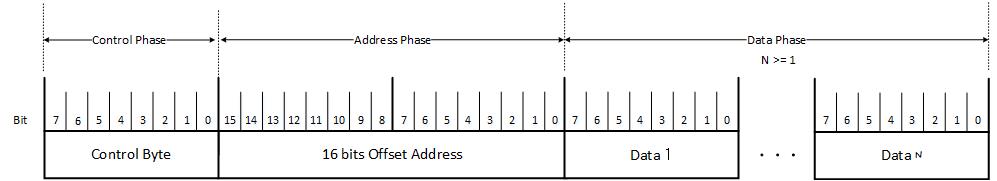


Figure 6 W5100S SPI Frame

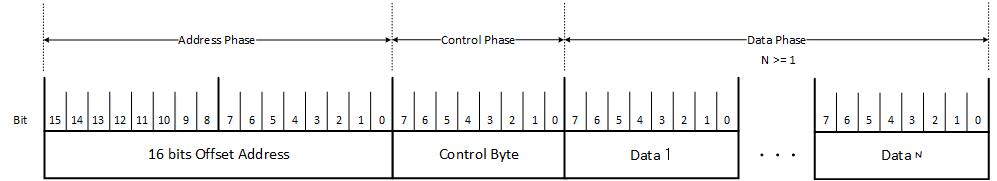


Figure 7 W5500 SPI Frame

* 1. DMA 및 BUS/SPI Setting
     1. Clock Configuration

W5100S EVB에 사용되는 MCU는 system clock의 Max clock 은 72MHz이며, AHB는 system clock과 동일값으로 사용된다. APB1와 APB2의 Max clock은 각각 36MHz, 72MHz이다.

Table 1 STM32103 Max Clock Configuration

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

Clock Configuration은 아래 그림에서 확인하면 된다.

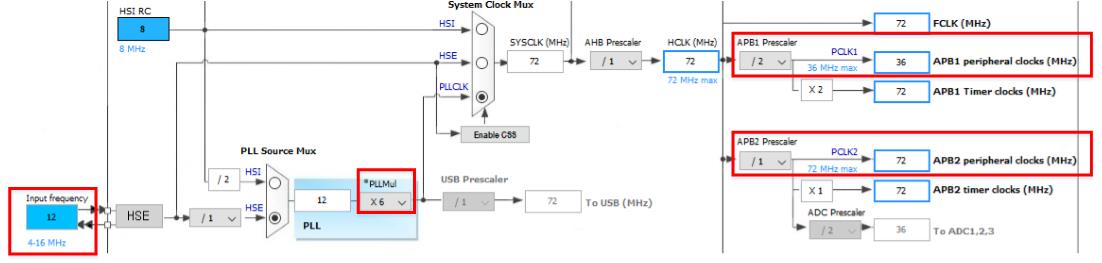


Figure 8 Clock Configuration

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

SYSCLK는 HSE\_VALUE \* PLL\_MUL로 계산되어진다.

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

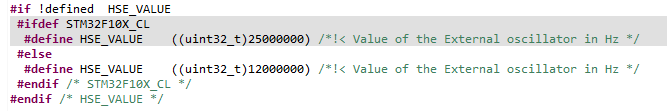


Figure 9 stm32f10x.h

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

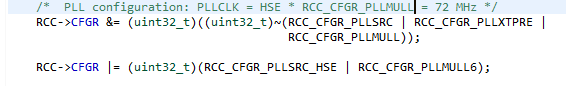


Figure 10 system\_stm32f10x.c

APB CLOCK은 기본으로 아래와 같이 설정되어 있다.

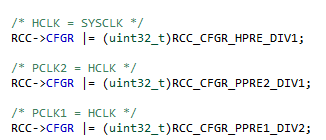


Figure 11 Clock Prescaler in system\_stm32f10x.c

* + 1. DMA 및 BUS/SPI 순서도

DMA경우 Read 나 Write 하나의 동작 할 경우라도 DMA RX, TX 동시에 수행된다.

또한, DMA\_Cmd를 사용할 경우 DMA Channel 중 RX를 먼저 Enable 시켜줘야 한다.

DMA\_Cmd이후 RX, TX FLAG가 뜰때까지 기다린 후 DMA를 Disable 해야 한다.

간단한 설명은 아래의 순서도를 이해하면 된다.

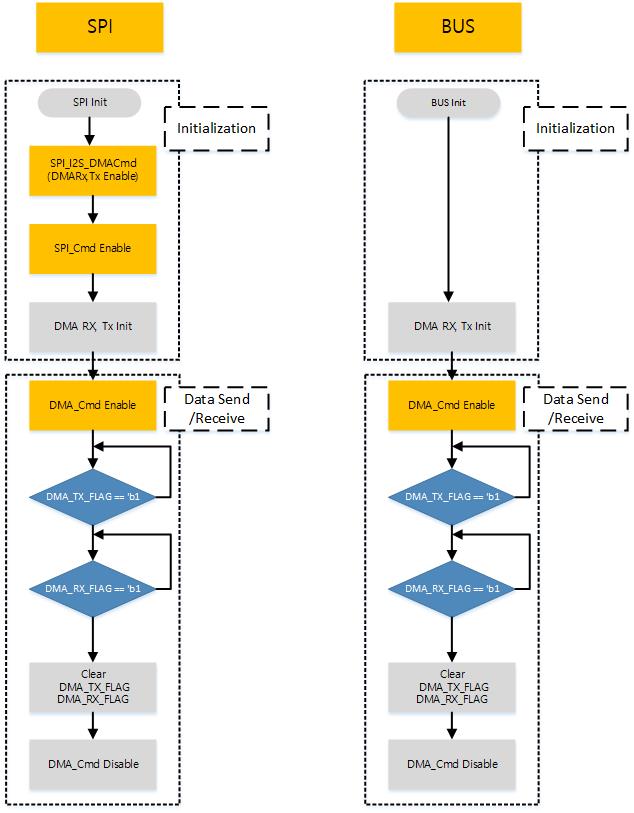


Figure 12 DMA, SPI 및 버스 동작

* + 1. SPI DMA
       1. Callback 함수

Callback 함수에 SPI 로 사용하고 싶은 함수를 등록하여 사용한다.

reg\_wizchip\_spi\_cbfunc(spiReadByte, spiWriteByte);

reg\_wizchip\_spiburst\_cbfunc(spiReadBurst, spiWriteBurst);

* + - 1. SPI Initialization 함수

SPI Max Clock 은 APB Clock의 1/2이며, 앞서 설명할 것처럼 SPI1 Clock는 36MHz, SPI2 Clock 은 18MHz가 Max Clock이 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*);

* + - 1. 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\_RX와 동일**

DMA\_Init(DMA\_CHANNEL\_TX, &DMA\_TX\_InitStructure);

* + - 1. 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*);

}

* + 1. Indirect Bus
       1. Callback 함수

Callback 함수에 Bus 로 사용하고 싶은 함수를 등록하여 사용한다.

reg\_wizchip\_bus\_cbfunc(busReadByte, busWriteByte);

reg\_wizchip\_busburst\_cbfunc(busReadBurst, busWriteBurst);

* + - 1. BUS Initialization 함수

BUS는 AHB Clock을 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*);

* + - 1. 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**

* + - 1. BUS ReadBurst 함수

아래의 구문의 제외하고는 BUS ReadBurst 와 BUS WriteBurst는 동일한 구조를 가지고 있으며, 예는 BUS Read Burst이다.

**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*);

}

1. 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 | 1.5Mpbs |
| 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 |

해당 표는 W5100S\_EVB를 사용하여 테스트를 진행한 것이며, SPI2에 W5500 Frame의 내용 없는 것은 W5100S EVB에 MOD0가 ‘0’으로 고정되어 있기 때문이다. 그러나, SPI1에서는 두가지의 Frame을 사용하여 테스트 한 경우 차이 점을 찾지 못했다.

SPI, BUS 모두 DMA를 사용하였을 때 SPI보다 BUS가 약 2배이상 빠른 것을 볼 수 있다.

자세한 테스트 속도를 확인하려면 참고 자료를 보시면 됩니다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Socket  Size | BUS | SPI |
| DMA | 8K |  |  |
| 4K |  |  |
| 2K |  |  |

1. 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)