

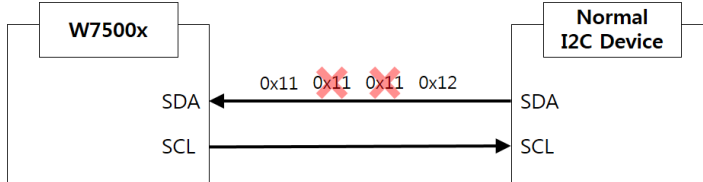
## W7500x Errata Sheet

### Document History

Ver 1.0.0 (July.11, 2016)	First release (erratum 1) - W7500x I2C
Ver 1.1.0 (Jun.18, 2018)	erratum 2 - W7500P Transmission Delay Case

© 2016 WIZnet Co., Inc. All Rights Reserved.

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

Erratum 1	
W7500x I2C	
Phenomenon	I2C에 반복적인 DATA가 들어올 경우 이후에는 멈춘다.
Condition	 <p>W7500x가 I2C통신중에 같은 데이터를 연속적으로 수신하게 되면, 처음 데이터만 수신하고 이후의 같은 데이터들은 인식하지 못하고 데이터를 버리는 문제. 그리고 이로 인한 데이터 손실이 발생한다.</p>
Solution & Recommendation	<p>이 Erratum을 피하기 위해서는 W7500x의 I2C를 GPIO로 변경하여 사용해야 한다. GPIO로 사용할 경우, SCL은 최대 100KHz를 초과할 수 없다.</p> <p>Example pseudo code:</p> <pre> Function Initialize_I2C () {     ...      scl_port_num = I2C_PORT(conf-&gt;scl);     scl_pin_index = I2C_PIN_INDEX(conf-&gt;scl);      sda_port_num = I2C_PORT(conf-&gt;sda);     sda_pin_index = I2C_PIN_INDEX(conf-&gt;sda);      //SCL setting     GPIO_InitDef.GPIO_Pin = scl_pin_index;     GPIO_InitDef.GPIO_Mode = GPIO_Mode_OUT;      if(scl_port_num == 0)     {         GPIO_Init(GPIOA, &amp;GPIO_InitDef);         GPIO_SetBits(GPIOA, scl_pin_index);     }      ...      //SDA setting         </pre>

```

GPIO_InitDef.GPIO_Pin = sda_pin_index;
GPIO_InitDef.GPIO_Mode = GPIO_Mode_IN;
if(sda_port_num == 0)
{
    GPIO_Init(GPIOA, &GPIO_InitDef);
    GPIO_ResetBits(GPIOA, sda_pin_index);
}
...
}

/* SCL function */
Function I2C_SCL()
{
    ...
    if(scl_port_num == 0)
    {
        if(data == 1)
            GPIO_SetBits(GPIOA, scl_pin_index);
        else
            GPIO_ResetBits(GPIOA, scl_pin_index);
    }
    ...
}

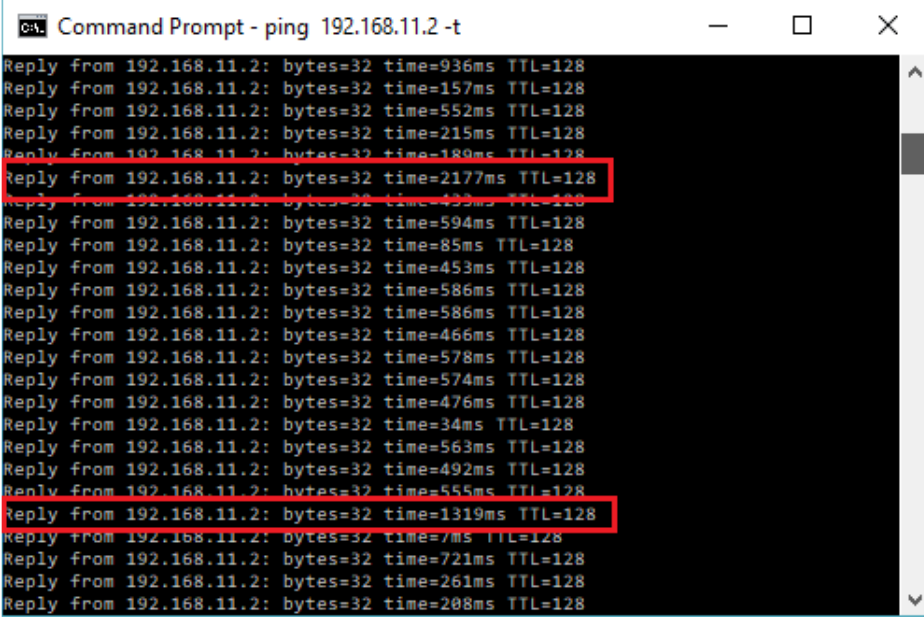
/* SDA function */
Function I2C_SDA()
{
    ...
    if(sda_port_num == 0)
    {
        if(data == 1)
            GPIOA->OUTENCLR = sda_pin_index;
        else
            GPIOA->OUTENSET = sda_pin_index;
    }
    ...
}

```

```
}  
  
/* START function */  
Function I2C_START()  
void I2C_Start(I2C_ConfigStruct* conf)  
{  
    I2C_WriteBitSCL(conf, 1);  
    I2C_WriteBitSDA(conf, 1);  
  
    I2C_WriteBitSDA(conf, 0);  
    I2C_WriteBitSCL(conf, 0);  
}  
  
/* STOP function */  
Function I2C_STOP()  
void I2C_Stop(I2C_ConfigStruct* conf)  
{  
    I2C_WriteBitSCL(conf, 0);  
    I2C_WriteBitSDA(conf, 0);  
  
    I2C_WriteBitSCL(conf, 1);  
    I2C_WriteBitSDA(conf, 1);  
}  
.....
```

## Erratum 2

### W7500P Transmission Delay Case

<p>Phenomenon</p>	<p>W7500P가 일부 스위치 또는 라우터와의 연동 시 Half Duplex로 인식되면서 TX가 늦게 나가는 현상이 발견됨. (아래는 “TP_LINK AC750” 라우터로 테스트한 결과이다.)</p>  <p>위의 결과와 같이 핑 응답이 3 초 이상 지연되는 현상이 불규칙적으로 발생하는 것을 확인할 수 있다.</p>
<p>Condition</p>	<p>이와 같은 현상은 NC(Not Connected) 패드 및 칩 내부의 PHY MII 신호와 관련된 연결 문제로 인해 발생된다. (W7500P는 silicon-in-package 제품이며 내부에 W7500 및 Ethernet PHY를 포함한다.); duplex mode의 잘못된 검출로 인한 Collision 처리에 의해 전송 패킷이 지연된다.</p>
<p>Solution &amp; Recommendation</p>	<p>이 현상을 해결하기 위해서, 사용자는 반드시 아래의 PHY 초기화 코드를 삽입해야 한다.</p> <pre>void PHY_Init(void) { #ifdef __W7500P__ // W7500P only     // PB_12     *(volatile uint32_t *) (0x41003070) = 0x61; // RXDV: set pull down     // PB_05 </pre>

```

*(volatile uint32_t *) (0x41002054) = 0x01;
*(volatile uint32_t *) (0x41003054) = 0x61;

// PB_06
*(volatile uint32_t *) (0x41002058) = 0x01;
*(volatile uint32_t *) (0x41003058) = 0x61;

// PHY reset pin pull-up (PD_06)
*(volatile uint32_t *) (0x410020D8) = 0x01;
*(volatile uint32_t *) (0x410030D8) = 0x02;
*(volatile uint32_t *) (0x45000004) = 0x40;
*(volatile uint32_t *) (0x45000010) = 0x40;

mdio_init(GPIOB, W7500x_MDC, W7500x_MDIO); // MDIO Init
mdio_write(GPIOB, PHYREG_CONTROL, CNTL_RESET); // PHY Reset
#endif
}

```

W7500P의 DUP pin(pin 15)는 스위치 또는 라우터와 연동된 duplex mode를 나타내며 value는 다음과 같다.

- DUP pin = '1' (HIGH) : Full duplex mode
- DUP pin = '0' (LOW) : Half duplex mode