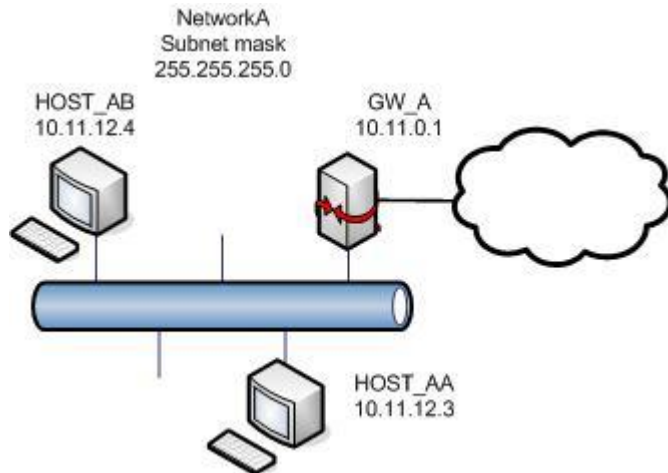


W3150A+/W5100 Errata Sheet

Document History

Ver. 1.0.0 (OCT. 27, 2007)	First release (erratum 1)
Ver. 2.0.0 (SEP. 10, 2008)	Add W5100 solution for erratum 1 Remove Recommendation for erratum 1
Ver. 2.1 (APR. 5, 2010)	Remove the erratum solution for W5100 in v2.0 and bring back the recommendation of v1.0
Ver. 2.2 (FEB. 17, 2012)	Add erratum 2, 3
Ver. 2.3 (MAR. 5, 2012)	Add a solution for erratum 2, 3
Ver. 2.4 (OCT. 28, 2013)	Add a description of solution for erratum 2
Ver. 2.5 (JUL. 7, 2014)	Modify a description of solution for erratum 2
Ver. 2.6 (OCT. 5, 2015)	Not support SPI mode 3

© 2012 WIZnet Co., Ltd. All Rights Reserved.
 For more information, visit our website at <http://www.wiznet.co.kr>

Erratum 1	
Phenomenon	<p>Unable to complete SEND/SENDMAC command in UDP/IP-Raw mode. That is, do not assert SEND_OK interrupt and do not equal the values of Socket n Tx Read Pointer Register (Sn_TX_RD) and Socket n Tx Write Pointer Register (Sn_TX_WR).</p>
Condition	<p>Generally, the Sn_TX_RD value is equal to the Sn_TX_WR value and asserts SEND_OK interrupt when a transmission process was completed. But the case of errata which the Sn_TX_RD and Sn_TX_WR values are different and do not assert SEND_OK interrupt in UDP/IP-Raw mode. This phenomenon is occurred when our chip receive data in opened socket (i.e. assert RECV interrupt) and simultaneously the application program (Host MCU) executes SEND/SENDMAC command on that socket.</p> <div style="text-align: center;">  </div> <p>Figure 1. General network</p> <p>In Figure 1 network,</p> <ol style="list-style-type: none"> 1. Open UDP socket in AA. Port number is 1000. 2. Send UDP data packet(destination port number 1000) from AB to AA. 3. AA processes the UDP data packet from AB. 4. Host MCU in AA runs SEND command to send. <p>When completion of 3 in AA (i.e. AA assert RECV interrupt), AA also runs 4 (i.e. run SEND command). In that case, do not complete process 4 due to errata.</p>

Solution	<p>In case of W3150A+,</p> <p>If you can monitor the RXDV signal (in MII interface), you can solve this problem.</p> <p>Before you execute SEND/SENDMAC command in UDP/IP-Raw mode, check the value of RXDV signal is '0'. That means there is no received data packet, so you can avoid the errata condition.</p> <p>Refer to following pseudo-code.</p> <pre style="background-color: black; color: yellow; padding: 10px;"> /* Change sendto() function */ Function sendto() { ... While (RXDV == '1') ; SEND command; /* Complete Sending */ } </pre>
Recommendation	<p>In case of W5100,</p> <p>We don't have solution but we can give a recommendation.</p> <p>After complete a transmission process, check whether Sn_TX_RD and Sn_TX_WR have same values or not. If both values are still different, close the socket and reset by force.</p> <pre style="background-color: black; color: yellow; padding: 10px;"> /* Change sendto() function */ Function sendto() { ... /* Complete Sending */ /* wait until Sn_TX_WR and Sn_TX_RD are same */ While (Sn_TX_WR != Sn_TX_RD) { wait some time; loop_cnt++; if (loop_cnt > CONST_BLOCK_CNT) goto RESET } } </pre>

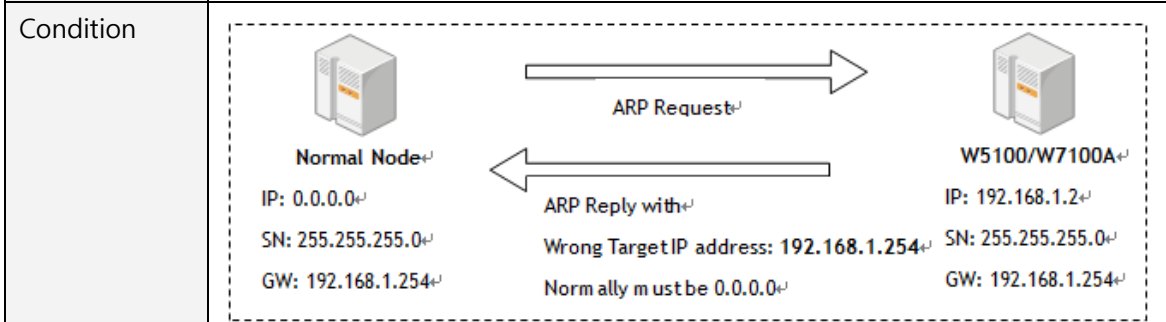
```

}

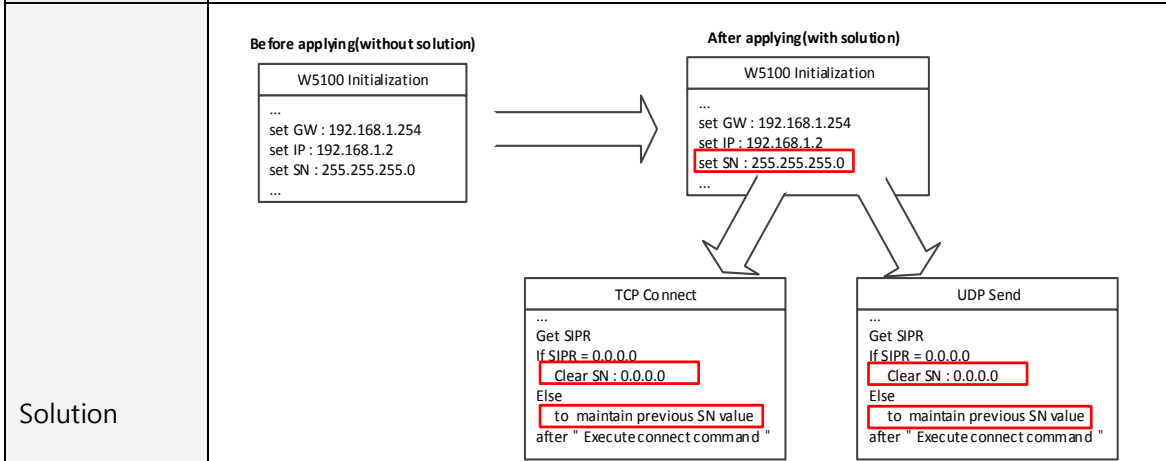
```

Erratum 2

Phenomenon The W5100 replies with gateway IP address for the ARP request from network node which has "0.0.0.0" IP address. But normally the W5100 should replies with target IP address "0.0.0.0" not the gateway IP address.



The main reason of this erratum is subnet calculating logic. The W5100 misunderstands the node locates other sub-network when target has "0.0.0.0" IP address. So the W5100 set the target IP to the gateway IP instead of "0.0.0.0" and sends the ARP reply.



Solution

To avoid this erratum we must keep the subnet mask register value to setting value except two cases which are "CONNECT" command in TCP and "SEND" command in UDP. Because only these two cases are referring the subnet mask register and sending the ARP request.

So set the subnet mask register to "FF.FF.FF.xx" and keeping it but save the right subnet mask value to the global variable when you initialize the W5100. When you use connect command in TCP or send command in UDP, set the subnet mask register to zero using the variable before executing connect or

send command. After done connect or send command, sets the subnet mask register again to keep its value to “FF.FF.FF.xx”.

In the case of applying, you can't use the subnet broadcasting.

Example pseudo code:

```

/* Global variable declaration for subnet mask value */
unsigned char subnet_val[4];
/* W5100 initialization function */
Function Initialize_W5100( )
{
...
/* Clear the subnet mask register */
    IINCHIP_WRITE(SUBR0, 0);
    IINCHIP_WRITE(SUBR1, 0);
    IINCHIP_WRITE(SUBR2, 0);
    IINCHIP_WRITE(SUBR3, 0);
/* Save the right subnet mask value if the subnet is 255.255.255.0 */
    subnet_val[0] = 255;
    subnet_val[1] = 255;
    subnet_val[2] = 255;
    subnet_val[3] = 0;
...
}

/* TCP connect function */
Function TCP_Connect( )
{
...

/* Clear the subnet mask register again and keep it */
    IP_Val[0] = IINCHIP_READ(SIPR0);
    IP_Val[1] = IINCHIP_READ(SIPR0+1);
    IP_Val[2] = IINCHIP_READ(SIPR0+2);
    IP_Val[3] = IINCHIP_READ(SIPR0+3);

    If( IP_Val[0]==0 && IP_Val[1] ==0&& IP_Val[2] ==0&& IP_Val[3] ==0)
    
```

```

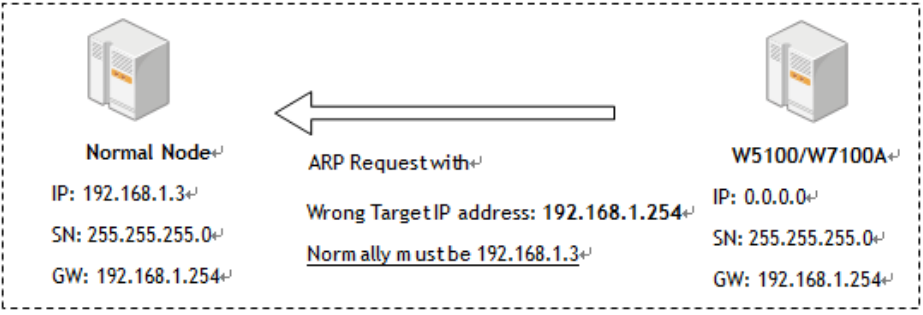
{
    IINCHIP_WRITE(SUBR0, 0);
    IINCHIP_WRITE(SUBR1, 0);
    IINCHIP_WRITE(SUBR2, 0);
    IINCHIP_WRITE(SUBR3, 0);
}
/* Execute TCP connect command */
    IINCHIP_WRITE(Sn_CR(socket), Sn_CR_CONNECT);
/* Wait for command done */
    while(Sn_CR(socket));
/* Set the subnet mask register to the right value using the variable */
    IINCHIP_WRITE(SUBR0, subnet_val[0]);
    IINCHIP_WRITE(SUBR1, subnet_val[1]);
    IINCHIP_WRITE(SUBR2, subnet_val[2]);
    IINCHIP_WRITE(SUBR3, subnet_val[3]);
...
}

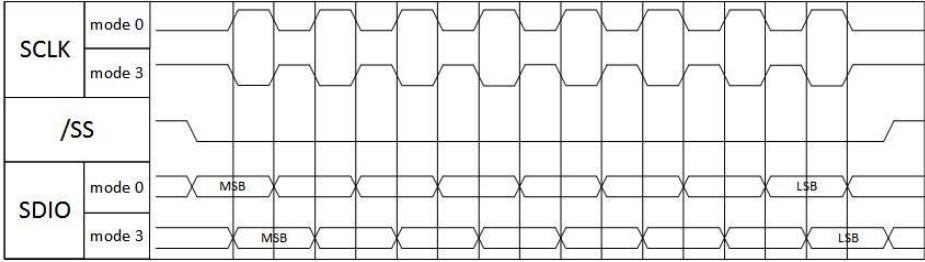
/* UDP sendto function */
Function UDP_Sendto( )
{
...
/* Clear the subnet mask register again and keep it */
    IP_Val[0] = IINCHIP_READ(SIPR0);
    IP_Val[1] = IINCHIP_READ(SIPR0+1);
    IP_Val[2] = IINCHIP_READ(SIPR0+2);
    IP_Val[3] = IINCHIP_READ(SIPR0+3);

    If( IP_Val[0]==0 && IP_Val[1] ==0&& IP_Val[2] ==0&& IP_Val[3] ==0)
    {
        IINCHIP_WRITE(SUBR0, 0);
        IINCHIP_WRITE(SUBR1, 0);
        IINCHIP_WRITE(SUBR2, 0);
        IINCHIP_WRITE(SUBR3, 0);
    }
}
/* Execute UDP send command */

```

```
IINCHIP_WRITE(Sn_CR(socket), Sn_CR_SEND);  
/* Wait for command done */  
while(Sn_CR(socket));  
/* Set the subnet mask register to the right value using the variable */  
IINCHIP_WRITE(SUBR0, subnet_val[0]);  
IINCHIP_WRITE(SUBR1, subnet_val[1]);  
IINCHIP_WRITE(SUBR2, subnet_val[2]);  
IINCHIP_WRITE(SUBR3, subnet_val[3]);  
...  
}
```

Erratum 3	
Phenomenon	Assuming that the IP address of W5100 is “0.0.0.0” and the gateway, subnet mask is valid (not “0.0.0.0”), the W5100 set the target IP address of ARP request to the gateway IP address not the target node IP address when sends ARP request to another node. So the peer node cannot receive the ARP request from the W5100.
Condition	<div style="border: 1px dashed gray; padding: 10px; text-align: center;">  <p> Normal Node IP: 192.168.1.3 SN: 255.255.255.0 GW: 192.168.1.254 </p> <p> W5100/W7100A IP: 0.0.0.0 SN: 255.255.255.0 GW: 192.168.1.254 </p> </div> <p> The W5100 miss calculates the sub-network location when sends the ARP request if its own IP address is “0.0.0.0”. In the same condition, even if the gateway IP address is “0.0.0.0”, the W5100 sends ARP request to “0.0.0.0” IP address because the W5100 sends ARP request to the gateway. </p>
Solution	The reason of this erratum3 is same as erratum2 so the solution is also same with erratum2. Please refer to the solution of erratum2.

Errata 4	
Phenomenon	Not support SPI mode 3 with a bug
Condition	<p>The documents of W5100 and W3150A+ said that it supports two SPI modes (mode 0, 3).</p> <p>The below figure shows the Timing diagram in SPI mode 0 and mode 3.</p>  <p>But now, W5100 and W3150A+ do not support SPI mode 3 with a bug.</p>
Solution	Not support SPI mode 3