How to Test Internal TX/RX Memory in W5300

Version 1.0

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

Document History Information

Version	Date	Descriptions
Ver. 0.9	Mar. 11, 2008	Release with W5300 launching
Ver. 1.0	May, 15, 2008	Replace W5300 image on front page.

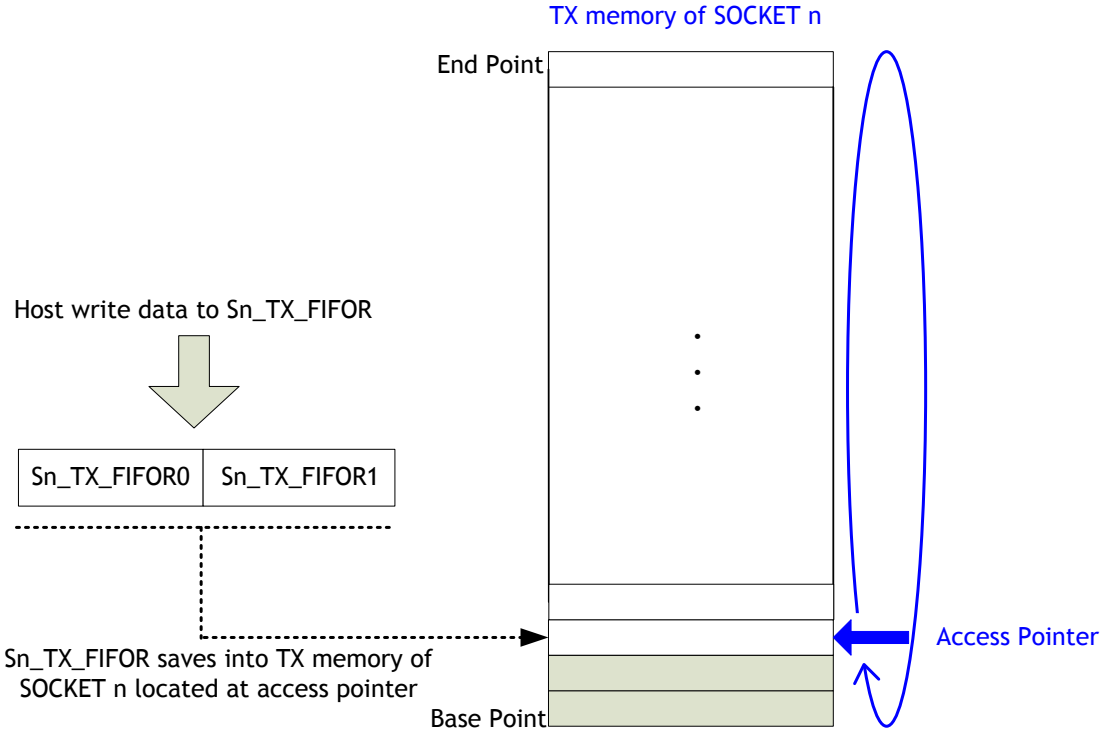
Internal TX/RX Memory

W5300 has 128KBytes memory for data communication. This memory is divided into TX and RX memory according to TMSR and RMSR Register setting, and re-divided into TX and RX memory in each socket.

Direct access from the host to the internal TX/RX memory is not possible, but only indirect access is allowed through FIFO Registers in each socket. By using Sn_TX_FIFOR and Sn_RX_FIFOR, indirect access to TX and RX memory are possible.

In normal operation of W5300, TX memory allows only Host-Write through Sn_TX_FIFOR, and RX memory does only Host-Read through Sn_RX_FIFOR. In order to check if Host-Write data is appropriately written in TX memory and Host-Read data is correctly read from RX memory, let the W5300 operate in memory test mode.

Sn_TX_FIFOR & TX Memory

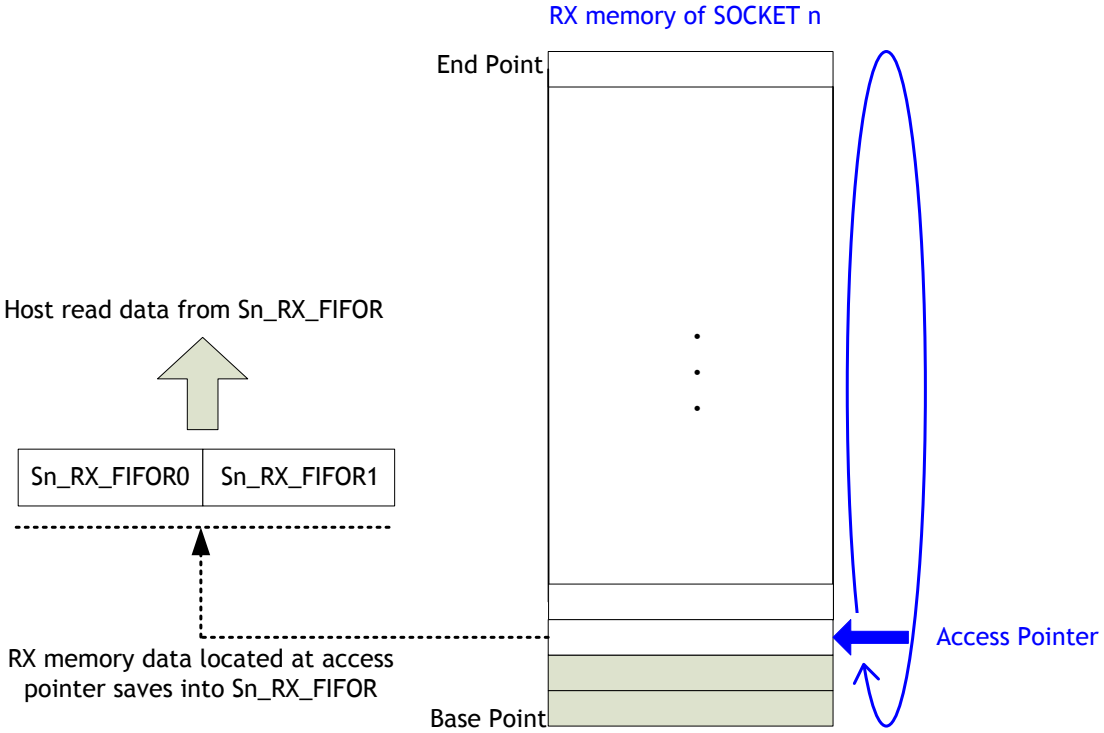


TX memory has internal Access Pointer. Whenever the Host writes the Sn_TX_FIFOR, Access Pointer internally increases from base to end point of TX memory by 2byte offset after saving Sn_TX_FIFOR into TX memory of SOCKET n.

In case that the Host keeps writing Sn_TX_FIFOR at the Access Pointer which is increased to the end point of TX Memory, the Access Pointer rolls back to base point from the end point. Access Pointer returns to the beginning position when Sn_TX_FIFOR is written as much as assigned memory size. Generally, Access Pointer increasing and roll-back function is supported

in case of the Host-Write of Sn_TX_FIFOR. However, if 'MT' bit of MR is set as '1', the function is supported even in case of Host-Read of Sn_TX_FIFOR. Therefore, by using the function, the Host-Write operation of TX memory can be verified.

Sn_RX_FIFOR와 RX Memory



RX memory also has internal Access Pointer. Whenever the Host reads Sn_RX_FIFOR, Access Pointer internally increases from the base to end point of RX memory of SOCKET n by 2Byte offset.

When the Host keeps reading Sn_RX_FIFOR at the Access Pointer which is increased to end point, the Access Pointer rolls back to the base point. Access Pointer goes back to the beginning point when Sn_RX_FIFOR is read as much as assigned memory size. Generally, Access Pointer increasing and roll-back function is supported only in case of Host-Read of Sn_RX_FIFOR. However, if 'MT' bit of MR is set as '1', the function is supported even in case of Host-Write of Sn_RX_FIFOR. By using this function, it is possible to verify the Host-Read operation of RX memory.

How to Test SOCKET memory

For the testing, follow below.

- W5300 Initialization

- SOCKET Memory Allocation
- Set the MT bit of MR
- Open the SOCKET n
- SOCKET n Memory Test
- Close the SOCKET n
- Un-set the MT bit of MR

Ex1) SOCKET 1 TX Memory Test for 16 bit data bus width & direct address mode.

```
{
  /* W5300 Initialize */
  MR = MR | 0x0080;                  // W5300 Soft Reset
  Wait for PLL lock-in time (about 10ms)

  /* SOCKET Memory Allocation - Instead of using the default size, you can use to your size
                                in each SOCKET TX/RX memory*/
  TMS01R = 0x0808;
  TMS23R = 0x0808;
  TMS45R = 0x0808;
  TMS67R = 0x0808;

  RMS01R = 0x0808;
  RMS23R = 0x0808;
  RMS45R = 0x0808;
  RMS67R = 0x0808;

  MTYPER = 0x00FF;

  /* Set MR(MT) to '1' */
  MR = MR | 0x0020;

  /* SOCKET n Open */
  S1_MR = 0x0001;                    // sets TCP mode. You can set Sn_MR to other mode
  S1_CR = 0x0001;                    // sets OPEN command
  while(S1_CR != 0x0000);            // wait until the command is cleared by W5300
  // wait until Sn_SSR is changed to SOCK_INIT
  while((S1_SSR & 0x00FF) != SOCK_INIT);
```

```
  /* Test TX Memory of SOCKET n */
  // As TX memory size half of SOCKET 1 Host write the test data to S1_TX_FIFOR.
  test_data = 0x0000;
  for (i = 0; i < 8192/2 ; i++)
  {
      S1_TX_FIFOR = test_data;
      test_data = test_data + 0x0101;
  }
  // For Verification of SOCKET 1 TX memory, As size half of its Host read the read_data from
  S1_TX_FIFOR. And then it compares the test_data and the read_data.
  test_data = 0x0000;
  for(i = 0; i < 8192/2 ; i++)
  {
      read_data = S1_TX_FIFOR;
      if(test_data != read_data) FAIL to verify TX memory of SOCKET 1.
      test_data = test_data + 0x0101;
  }

  /* Close SOCKET n */
  S1_CR = 0x0010;                    //sets CLOSE command
  while(S1_CR != 0x0000);            //wait unit the command is cleared by W5300
  // wait until Sn_SSR is changed to SOCK_CLOSED
  while((S1_SSR & 0x00FF) != SOCK_CLOSED);

  /* Clear the 'MT' bit of MR */
  MR = MR & 0xFFDF;                  // Clear MT bit of MR
}
```

Ex2) SOCKET 1 RX Memory Test for 8 bit data bus width & direct address mode.

```
{
  /* W5300 Initialize */
  MR1 = MR1 | 0x80;                  // W5300 Soft Reset
  Wait for PLL lock-in time (about 10ms)

  /* SOCKET Memory Allocation - Instead of using the default size, you can use to your
                                memory size in each SOCKET TX/RX memory*/
  TMSR0 = 0x08;
```

```
TMSR1 = 0x08;
TMSR2 = 0x08;
TMSR4 = 0x08;
TMSR5 = 0x08;
TMSR6 = 0x08;
TMSR7 = 0x08;

RMSR0 = 0x08;
RMSR1 = 0x08;
RMSR2 = 0x08;
RMSR4 = 0x08;
RMSR5 = 0x08;
RMSR6 = 0x08;
RMSR7 = 0x08;

MTYPER0 = 0x00;
MTYPER1 = 0xFF;

/* Set MR(MT) to '1' */
MR1 = MR1 | 0x20;

/* SOCKET n Open */
S1_MR1 = 0x01;                  // sets TCP mode. You can set Sn_MR to other mode
S1_CR1 = 0x01;                  // sets OPEN command
while(S1_CR1 != 0x00);          // wait until the command is cleared by W5300
// wait until Sn_SSR is changed to SOCK_INIT
while(S1_SSR1 != SOCK_INIT);

/* Test RX Memory of SOCKET n */
// As RX memory size half of SOCKET 1 Host write the test data to S1_RX_FIFOR.
test_data[0] = 0x00;
test_data[1] = 0x00;
for (i = 0; i < 8192/2 ; i++)
{
    S1_RX_FIFOR0 = test_data[0];
    S1_RX_FIFOR1 = test_data[1];
    test_data[0] = test_data[0] + 0x01;
```

```
        test_data[0] = test_data[0] + 0x01;
    }
    // For Verification of SOCKET 1 RX memory, As size half of its Host read the read_data from
    S1_RX_FIFOR. And then it compares the test_data and the read_data.
    test_data[0] = 0x00;
    test_data[1] = 0x00;
    for(i = 0; i < 8192/2 ; i++)
    {
        read_data[0] = S1_RX_FIFOR0;
        read_data[1] = S1_RX_FIFOR1;
        if( (test_data[0] != read_data[0]) || (test_data[1] != read_data[1]) )
            FAIL to verify RX memory of SOCKET 1.
        test_data[0] = test_data[0] + 0x01;
        test_data[1] = test_data[1] + 0x01;
    }

    /* Close SOCKET n */
    S1_CR1 = 0x10;                  //sets CLOSE command
    while(S1_CR1 != 0x00);          //wait unit the command is cleared by W5300
    // wait until Sn_SSR is changed to SOCK_CLOSED
    while(S1_SSR1 != SOCK_CLOSED);

    /* Clear the 'MT' bit of MR */
    MR1 = MR1 & 0xDF;               // Clear MT bit of MR
}
```

Ex3) SOCKET 1 TX Memory Test for 16 bit data bus width & Indirect address mode.

```
{
    /* W5300 Initialize */
    MR = MR | 0x0080;               // W5300 Soft Reset
    Wait for PLL lock-in time (about 10ms)
    MR = MR | 0x0001;               // set IND bit of MR to '1' for indirect address mode

    /* SOCKET Memory Allocation - Instead of using the default size, you can use to your
                                   memory size in each SOCKET TX/RX memory*/
```

```
IDM_AR = Address Offset of TMSR01R;       // 0x0020
IDM_DR = 0x0808;
IDM_AR = Address Offset of TMSR23R;       // 0x0022
IDM_DR = 0x0808;
IDM_AR = Address Offset of TMSR45R;       // 0x0024
IDM_DR = 0x0808;
IDM_AR = Address Offset of TMSR67R;       // 0x0026
IDM_DR = 0x0808;

IDM_AR = Address Offset of RMSR01R;       // 0x0028
IDM_DR = 0x0808;
IDM_AR = Address Offset of RMSR23R;       // 0x002A
IDM_DR = 0x0808;
IDM_AR = Address Offset of RMSR45R;       // 0x002C
IDM_DR = 0x0808;
IDM_AR = Address Offset of RMSR67R;       // 0x002E
IDM_DR = 0x0808;

IDM_AR = Address Offset of MTYPER         // 0x0030
IDM_DR = 0x00FF;

/* Set MR(MT) to '1' */
MR = MR | 0x0020;

/* SOCKET n Open */
IDM_AR = Address Offset of S1_MR;
IDM_DR = 0x0001;                  // sets TCP mode. You can set Sn_MR to other mode
IDM_AR = Address Offset of S1_CR;
IDM_DR = 0x0001;                  // sets OPEN command
while(IDM_DR != 0x0000);          // wait until the command is cleared by W5300
// wait until Sn_SSR is changed to SOCK_INIT
IDM_AR = Address Offset of S1_SSR;
while((IDM_DR & 0x00FF) != SOCK_INIT);

/* Test TX Memory of SOCKET n */
// As TX memory size half of SOCKET 1 Host write the test data to S1_TX_FIFOR.
test_data = 0x0000;
```

```
  IDM_AR = Address offset of S1_TX_FIFOR;
  for (i = 0; i < 8192/2 ; i++)
  {
      IDM_DR = test_data;
      test_data = test_data + 0x0101;
  }
  // For Verification of SOCKET 1 TX memory, As size half of its Host read the read_data from
  S1_TX_FIFOR. And then it compares the test_data and the read_data.
  test_data = 0x0000;
  for(i = 0; i < 8192/2 ; i++)
  {
      read_data = IDM_DR;
      if(test_data != read_data) FAIL to verify TX memory of SOCKET 1.
      test_data = test_data + 0x0101;
  }

  /* Close SOCKET n */
  IDM_AR = Address Offset of S1_CR
  IDM_DR = 0x0010;                  //sets CLOSE command
  while(IDM_DR != 0x0000);          //wait unit the command is cleared by W5300
  // wait until Sn_SSR is changed to SOCK_CLOSED
  IDM_AR = Address Offset of S1_SSR
  while((IDM_DR & 0x00FF) != SOCK_CLOSED);

  /* Clear the 'MT' bit of MR */
  MR = MR & 0xFFDF;                 // Clear MT bit of MR
}
```

Ex4) SOCKET 1 RX Memory Test for 8 bit data bus width & Indirect address mode.

```
{
  /* W5300 Initialize */
  MR1 = MR1 | 0x80;                 // W5300 Soft Reset
  Wait for PLL lock-in time (about 10ms)
  MR1 = MR1 | 0x01;                 // set IND bit of MR to '1' for indirect address mode

  /* SOCKET Memory Allocation - Instead of using the default size, you can use to your
                                    memory size in each SOCKET TX/RX memory*/
```

```
IDM_AR0 = Higher Address Offset of TMSR01R;      // 0x00
IDM_AR1 = Lower Address Offset of TMSR01R;       // 0x20
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of TMSR23R;       // 0x22
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of TMSR45R;       // 0x24
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of TMSR67R;       // 0x26
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of RMSR01R;       // 0x28
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of RMSR23R;       // 0x2A
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of RMSR45R;       // 0x2C
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of RMSR67R;       // 0x2E
IDM_DR0 = 0x08;
IDM_DR1 = 0x08;

IDM_AR1 = Lower Address Offset of MTYPER         // 0x30
IDM_DR0 = 0x00;
IDM_DR1 = 0xFF;
```

```
/* Set MR(MT) to '1' */
MR1 = MR1 | 0x20;

/* SOCKET n Open */
IDM_AR0 = Higher Address Offset of S1_MR;          // 0x02
IDM_AR1 = Lower Address Offset of S1_MR;          // 0x40
IDM_DR1 = 0x01;                    // sets TCP mode. You can set Sn_MR to other mode

IDM_AR1 = Lower Address Offset of S1_CR;          // 0x42
IDM_DR1 = 0x01;                    // sets OPEN command
while(IDM_DR1 != 0x00);            // wait until the command is cleared by W5300
// wait until Sn_SSR is changed to SOCK_INIT
IDM_AR1 = Lower Address Offset of S1_SSR;         // 0x48
while(IDM_DR1 != SOCK_INIT);

/* Test RX Memory of SOCKET n */
// As RX memory size half of SOCKET 1 Host write the test data to S1_RX_FIFOR.
test_data[0] = 0x00;
test_data[1] = 0x00;
IDM_AR1 = Lower Address offset of S1_RX_FIFOR;    // 0x70
for (i = 0; i < 8192/2 ; i++)
{
    IDM_DR0 = test_data[0];
    IDM_DR1 = test_data[1];
    test_data[0] = test_data[0] + 0x01;
    test_data[1] = test_data[1] + 0x01;
}
// For Verification of SOCKET 1 RX memory, As size half of its Host read the read_data from
S1_RX_FIFOR. And then it compares the test_data and the read_data.
test_data[0] = 0x00;
test_data[0] = 0x00;
for(i = 0; i < 8192/2 ; i++)
{
    read_data[0] = IDM_DR0;
    read_data[1] = IDM_DR1;
    if( (test_data[0] != read_data[0]) || (test_data[1] != read_data[1]) )
        FAIL to verify RX memory of SOCKET 1.
```

```
        test_data[0] = test_data[0] + 0x01;
        test_data[1] = test_data[1] + 0x01;
    }

    /* Close SOCKET n */
    IDM_AR1 = Lower Address Offset of S1_CR          //0x42
    IDM_DR1 = 0x10;                  //sets CLOSE command
    while(IDM_DR1 != 0x0000);        //wait unit the command is cleared by W5300
    // wait until Sn_SSR is changed to SOCK_CLOSED
    IDM_AR1 = Lower Address Offset of S1_SSR         //0x48
    while(IDM_DR1 != SOCK_CLOSED);

    /* Clear the 'MT' bit of MR */
    MR1 = MR1 & 0xDF;                // Clear MT bit of MR
}
```

Trouble Shooting Guide

When the internal memory test is not normally processed, check below.

- Check VCC3V3, VCC3A3, VCC1V8, VCC1A8 Power
- Check 25MHz operation clock
- Check /RESET signal (should keep minimum 2us as low)
- Check if /CS, /RD, /WR signals are properly low-asserted when access to W5300 Access.
- Check IDR
- Check Read/Write operation of Registers such as SHAR, GAR, SUBR, SIPR
- Check MR Register Value (DBW, MT, IND bit)
- Check if total sum of TMSR and RMSR of each SOCKET is 128
- Check if total sum of TMSR of each SOCKET is multiple of 8
- Check if total sum of RMSR of each SOCKET is multiple of 8
- Check MTYPER setting value

In you still can't process internal testing memory after checking all above, dump all Registers of W5300 as below, and send to support@wiznet.co.kr.

Target Host : Part Name (Datasheet Attachment)

Bus Width : 16bit or 8bit

Address Mode : Direct or Indirect

Base Address : 0x08000

0x0000 - 0x00FF Registers Dump

0x0000 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

0x0010 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

... ...

0x00E0 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

0x00F0 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX 5300

SOCKET 0 Registers Dump

0x0200 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

0x0210 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

... ...

0x0230 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

SOCKET 1 Registers Dump

0x0240 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

... ...

0x0270 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

... ...

... ...

... ...

SOCKET 6 Registers Dump

0x0380 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

... ...

0x03B0 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

SOCKET 7 Registers Dump

0x03C0 : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

... ...

0x03FF : XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX