High-Performance
Internet Connectivity Solution

W5300
Version 1.3.1

© 2008-2014 WIZnet Co.,Ltd. All Rights Reserved.

For more information, visit our website at http://www.wiznet.co.kr
### High-performance Internet Connectivity Solution

#### W5300

**Document History Information**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver. 1.0.0</td>
<td>Mar. 11, 2008</td>
<td>Release with W5300 launching</td>
</tr>
</tbody>
</table>
| Ver. 1.1.0 | May. 15, 2008 | ◦ Correct a number of typing errors  
◦ 4.4 SOCKET Register >> Sn_DPORTR  
R/W → WO, Modify the description, Refer to P.77  
◦ 4.4 SOCKET Register >> Sn_MSSR  
In the MSS Table, Modified the PPPoE MSS value of MACRAW(1502 → 1514), Refer to P.79  
◦ 5.2.1.1 TCP SERVER >> • ESTABLISHED : Receiving process  
At the <Notice> phase, Modified the example code  
Replace ‘SEND’ with ‘SEND_KEEP’. Refer to P.93~94  
◦ 5.2.4 MACRAW >> • Receiving process  
At the <NOTICE> phase, Modified the free size and CRC  
Free size 1526 → 1528, CRC(2) → CRC(4), Refer to P.111 |
| Ver. 1.1.1 | July 4, 2008   | ◦ Correct a number of typing errors  
◦ Add PIN “BRDYn” description to “1.3 Host Interface signal”  
◦ 5.2.1.1 TCP SERVER >> • ESTABLISHED : Receiving process  
At the <Notice> phase, Modified the example code  
Replace ‘SEND_KEEP’ with ‘SEND’. Refer to P.93~94 |
| Ver. 1.2  | Dec. 30, 2008 | ◦ 1. PIN Description  
Add to ‘8’ Symbol  
◦ 1.2 Configuration Signals  
Modify ADDR type (ID → I), No Internal Pulled-down  
Modify DATA[15:0] type (IO → IO8)  
◦ 6.2. Indirect Address Mode  
ADDR[9:0] has no internal pulled-down resister. So, ADDR[9:3] should be connected to ground for using indirect address mode.  
Modify the description & figures. |
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
</table>
| Ver 1.2.1 | Jan. 22, 2009 | ◦ Modify the Figure 2.  
Ferrite Bead 0.1uF → 1uH |
| Ver 1.2.2 | Feb. 16, 2009 | ◦ 1.7 Clock Signals.  
Delete XTLP/XTLN Pin Type  
◦ 7. Electrical Specifications  
- DC Characteristics  
  : Modify the Test Condition of \( V_{OH}, V_{OL} \)  
  : \( V_{OH} \) - Min (2.0(2.4), Delete Typical and Max value  
  : \( V_{OL} \) – Delete Min and Typical value |
| V1.2.3   | Feb. 11, 2010 | ◦ Change Figure 2  
- Change W5300 Power Supply Signal schematic |
| V1.2.4   | Aug. 19, 2010 | - Change Temperature condition (p. 119) |
| V1.2.5   | Sep. 29, 2010 | ◦ Modify Table 1.8 Power Supply Signal (p. 21)  
- 1V8O: 1.8V regulator output voltage  
  capacitor value : 0.1uF → 10uF  
- Modified Figure 2 Power Design (p. 21) |
| V1.2.6   | Sep. 17, 2012 | ◦ Modified Figure 3  
- Modified W5300 Indirect Address Mode MR (p. 23) |
| V1.2.7   | Mar. 27, 2013 | ◦ correct a number of typing error  
- Modified Socket number of TMSR6,7 (p. 28)  
  Socket7 -> 6, Socket-8 -> 7  
- Modified range of Sn_PORTR (p. 79)  
  0x20A+0x40A -> 0x20A+0x40n  
- 7. Electrical Specifications  
- Modified Read register, Write register timing (P. 123~124) |
| V1.2.8   | JUN. 28, 2013 | ◦ correct a number of typing error  
- modified operating temperature \( T_{op} \) (p. 130)  
  -40 to 80 → -40 to 85 |
| V1.2.9   | FEB. 7, 2014 | ◦ Modified source code  
- recv_size calculation |
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.3.0</td>
<td>JUL. 11. 2014</td>
<td>Add &quot;&lt;notice&gt;&quot; information of LINKLED. (p.20)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify the “tDATA” of Register READ Timing (MIN -&gt; MAX) (p.124)</td>
</tr>
<tr>
<td>V1.3.1</td>
<td>MAR. 19. 2015</td>
<td>Modify the “frequency Tolerance” of Crystal Characteristics (p.125)</td>
</tr>
</tbody>
</table>
WIZnet’s online Technical Support

If you have something to ask about WIZnet Products, write down your question on Q&A Board of ‘Support’ menu in WIZnet website (www.wiznet.co.kr). WIZnet Engineer will give an answer as soon as possible.
W5300

W5300 is a 0.18 µm CMOS technology single chip into which 10/100 Ethernet controller, MAC, and TCP/IP are integrated. W5300 is designed for Internet embedded applications where easy implementation, stability, high performance, and effective cost are required. W5300’s target application is the embedded internet solution requiring high performance such as multi-media streaming service. Comparing to existing WIZnet chip solution, W5300 has been improved in memory and data process. W5300 is the most appropriate to the products of IPTV, IP-STB and DTV transferring multi-media data with high-capacity. The Internet connectivity can be implemented easily and quickly only with single chip having TCP/IP protocol and 10/100 Ethernet MAC & PHY.

High-Performance Hardware TCP/IP single chip solutions
WIZnet retains the technology of full hardware logic of communication protocols such as TCP, UDP, IPv4, ICMP, IGMP, ARP and PPPoE. In order to provide high-performing data communication, the data communication memory is extended to 128Kbyte and 16bit bus interface is supported in W5300. Users can utilize independent 8 hardware SOCKETs for high-speed data communication.

More flexible memory allocation for various applications
The memory for data communication can be allocated to each SOCKET in the range of 0~64Kbytes. It is more flexible for users to utilize the memory according to their application. Users can develop more efficient system by concentrating on the application of high performance.

Easy to implements for beginners
W5300 supports BUS interface as the host interface. By using direct and indirect access methods, W5300 can easily interfaced to the host as like SRAM memory. The data communication memory of W5300 can be accessed through TX/RX FIFO registers that exist in each SOCKET. With these features, even beginners can implement Internet connectivity by using W5300.
Target Applications

The W5300 is well-suited for many embedded applications, including:

- Home Network Devices: Set-Top Boxes, PVRs, Digital Media Adapters
- Serial-to-Ethernet: Access Controls, LED displays, etc.
- Parallel-to-Ethernet: POS / Mini Printers, Copiers
- USB-to-Ethernet: Storage Devices, Network Printers
- GPIO-to-Ethernet: Home Network Sensors
- Security Systems: DVRs, Network Cameras, Kiosks
- Factory and Building Automation
- Medical Monitoring Equipment
- Embedded Servers

Features

- Supports hardwired TCP/IP protocols: TCP, UDP, ICMP, IPv4, ARP, IGMPv2, PPPoE, Ethernet
- Supports 8 independent SOCKETs simultaneously
- High network performance: Up to 50Mbps
- Supports hybrid TCP/IP stack (software and hardware TCP/IP stack)
- Supports PPPoE connection (with PAP/CHAP Authentication mode)
- IP Fragmentation is not supported
- Internal 128Kbytes memory for data communication (Internal TX/RX memory)
- More flexible allocation internal TX/RX memory according to application throughput
- Supports memory-to-memory DMA (only 16bit Data bus width & slave mode)
- Embedded 10BaseT/100BaseTX Ethernet PHY
- Supports auto negotiation (Full-duplex and half duplex)
- Supports auto MDI/MDIX (Crossover)
- Supports network Indicator LEDs (TX, RX, Full/Half duplex, Collision, Link, Speed)
- Supports a external PHY instead of the internal PHY
- Supports 16/8 bit data bus width
- Supports 2 host interface mode (Direct address mode & Indirect address mode)
- External 25MHz operation frequency (For internal PLL logic, period=40ns)
- Internal 150MHz core operation frequency (PLL_CLK, period about 6.67ns)
- Network operation frequency (NIC_CLK: 25MHz (100BaseTX) or 2.5MHz (10BaseT))
- 3.3V operation with 5V I/O signal tolerance
- Embedded power regulator for 1.8V core operation
- 0.18 µm CMOS technology
- 100LQFP 14X14 Lead-Free Package
Block Diagram

- **Host**
- **Host Interface Manager**
- **Register Manager**
- **TCP/IP Core**
  - **TCP**
  - **UDP**
  - **ICMP**
  - **IP**
  - **PPPoE**
  - **ARP**
  - **802.3 Ethernet MAC**
- **MII Manager (CSMA/CD)**
- **128KB TX/RX DPRAM**
- **Memory Manager**
- **PLL**
- **Power Regulator**
- **150MHz**
- **25MHz**
- **3.3V**
- **1.8V**
- **Transformer**
- **3rd Party Ethernet PHY**
- **External MII**
- **Media Interface**
- **Transformer**
- **RJ45**
PLL (Phase-Locked Loop)
It creates a 150MHz clock signal by multiplying 25MHz clock source by six. The 150MHz clock is used for operating internal blocks such as TCP/IP core block, ‘Host Interface Manager’ and ‘Register Manager’. PLL is locked-in after reset and it supplies a stable clock.

Power Regulator
With 3.3V power input, the power regulator creates 1.8V/150mA power. This power regulator supplies the power for core operation of W5300. It is not required to add other power regulators, but recommended to add a capacitor for more stable 1.8V power supplying.

Host Interface Manager
It detects host bus signal, and manages read/write operations of the host according to data bus width or host interface mode.

Register Manager
It manages Mode register, COMMON Register, and SOCKET Register.

Memory Manager
It manages internal data memory of 128KBytes – TX/RX memory allocated in each SOCKET by the host. The host can access the memory only through TX/RX FIFO Register of each SOCKET.

128KB TX/RX DPRAM
It is the 128KByte memory for data communication and composed of 16 DPRAM (Dual-Port RAM) of 8KBytes. It is allocated flexibly to each SOCKET by the host.

MII (Media Independent Interface) Manager
It manages MII interface. MII interface can be switched to internal PHY or external PHY (3rd party PHY) according to the configuration of TEST_MODE[3:0].

Internal Ethernet PHY
W5300 includes 10BaseT/100BaseTX Ethernet PHY. Internal PHY supports half-duplex/full duplex, auto-negotiation and auto MDI/MDIX. It also supports 6 network indicator LED output such as Link status, speed and duplex.

TCP/IP Core
TCP/IP Core is the fully hardwired logic based on network protocol processing technology of WIZnet.
High-performance Internet Connectivity Solution  W5300

- **802.3 Ethernet MAC(Media Access Control)**
  It controls Ethernet access of CSMA/CD(Carrier Sense Multiple Access with Collision Detect). It is the protocol technology based on a 48-bit source/destination MAC address. It also allows the host to control MAC layer through its 0th SOCKET. So, it is possible to implement software TCP/IP stack together with hardware TCP/IP stack.

- **PPPoE(Point-To-Point Protocol over Ethernet)**
  It is the protocol technology to use PPP service at the Ethernet. It encapsulates the payload(data) part of Ethernet frame as the PPP frame and transmits it. When receiving, it de-capsulates the PPP frame. PPPoE supports PPP communication with PPPoE server and PAP/CHAP authentication methods.

- **ARP(Address Resolution Protocol)**
  ARP is the MAC address resolution protocol by using IP address. It transmits the ARP-reply to the ARP-request from the peer. It also sends ARP-request to find the MAC address of the peer and processes the ARP-reply to the request.

- **IP(Internet Protocol)**
  IP is the protocol technology to support data communication at the IP layer. IP fragmentation is not supported. It is not possible to receive the fragmented packets. Except for TCP or UDP, all protocol number is supported. In case of TCP or UDP, use the hardwired stack.

- **ICMP(Internet Control Message Protocol)**
  It receives the ICMP packets such as the fragment MTU, unreachable destination, and notifies the host. After receiving Ping-request ICMP packet, it transmits Ping-reply ICMP packet. It supports maximum 119 Byte as Ping-request size. If the size is over 119Bytes, it is not supported.

- **IGMPv1/v2(Internet Group Management Protocol version 1/2)**
  It processes IGMP such as IGMP Join/Leave, Report at the UDP multicasting mode. Only version 1 and 2 of IGMP logic is supported. When using upper version of IGMP, it should be manually implemented by using IP layer.

- **UDP(User Datagram Protocol)**
  It is the protocol technology to support data communication at the UDP layer. It supports user datagram such as unicast, multicast, and broadcast.

- **TCP(Transmission Control Protocol)**
  It is the protocol technology to support data communication at the TCP layer. It supports “TCP SERVER” and “TCP CLIENT” communication.

W5300 internally processes all protocol communication without intervention of the host. W5300 is based on TOE(TCP/IP Offload Engine) that can maximize the host performance by reducing the host overhead in processing TCP/IP stack.
Table of Contents

Table of Contents .................................................................................................................. 11
List of Figures .......................................................................................................................... 12
1. PIN Description .................................................................................................................. 13
   1.1 PIN Layout .................................................................................................................. 13
   1.2 Configuration Signals ................................................................................................. 14
   1.3 Host Interface Signals ............................................................................................... 15
   1.4 Media Interface Signals ............................................................................................. 17
   1.5 MII interface signal for external PHY ..................................................................... 18
   1.6 Network Indicator LED Signals ............................................................................... 20
   1.7 Clock Signals ............................................................................................................ 21
   1.8 Power Supply Signals ............................................................................................... 21
2. System Memory Map .......................................................................................................... 24
3. W5300 Registers ............................................................................................................... 26
   3.1 Mode Register .......................................................................................................... 27
   3.2 Indirect Mode Registers .......................................................................................... 27
   3.3 COMMON registers ................................................................................................. 27
   3.4 SOCKET registers ................................................................................................... 31
4. Register Description .......................................................................................................... 47
   4.1 Mode Register .......................................................................................................... 48
   4.2 Indirect Mode Registers .......................................................................................... 51
   4.3 COMMON Registers ............................................................................................... 52
   4.4 SOCKET Registers ................................................................................................. 68
5. Functional Description ....................................................................................................... 90
   5.1 Initialization ............................................................................................................... 90
   5.2 Data Communication ............................................................................................... 93
      5.2.1 TCP ................................................................................................................... 93
      5.2.2 UDP .................................................................................................................. 103
      5.2.3 IPRAW ............................................................................................................. 110
      5.2.4 MACRAW ....................................................................................................... 112
6. External Interface .............................................................................................................. 118
   6.1 Direct Address Mode ............................................................................................... 118
      6.1.1 16 Bit Data Bus Width ..................................................................................... 118
      6.1.2 8 Bit Data Bus Width ....................................................................................... 118
   6.2 Indirect Address Mode ............................................................................................ 119
      6.2.1 16 Bit Data Bus Width ..................................................................................... 119
List of Figures

Fig 1. PIN Layout ................................................................. 13
Fig 2. Power Design ........................................................... 22
Fig 3. Memory Map ............................................................. 25
Fig 4. ‘BRDYn’ Timing ........................................................... 67
Fig 5. SOCKETn Status Transition .......................................... 79
Fig 6. Access to Internal TX Memory ....................................... 88
Fig 7. Access to Internal RX Memory ....................................... 90
Fig 8. Allocation Internal TX/RX memory of SOCKETn ................. 92
Fig 9. "TCP SERVER" & "TCP CLIENT" ....................................... 93
Fig 10. "TCP SERVER" Operation Flow ...................................... 94
Fig 11. The received TCP data format ..................................... 96
Fig 12. "TCP CLIENT" Operation Flow ...................................... 102
Fig 13. UDP Operation Flow .................................................. 103
Fig 14. The received UDP data format ..................................... 104
Fig 15. IPRAW Operation Flow .............................................. 110
Fig 16. The received IPRAW data format .................................. 111
Fig 17. MACRAW Operation Flow .......................................... 112
Fig 18. The received MACRAW data format ............................. 113
Fig 19. Internal PHY & LED Signals ....................................... 120
Fig 20. External PHY Interface with MII ................................... 121
1. PIN Description

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I I</td>
<td>Input</td>
<td>D D</td>
<td>Internal pulled-down with 75KΩ resistor</td>
</tr>
<tr>
<td>O O</td>
<td>Output with driving current 2mA</td>
<td>M M</td>
<td>Multi-function</td>
</tr>
<tr>
<td>IO IO</td>
<td>Input/Output (Bidirectional)</td>
<td>H H</td>
<td>Active high</td>
</tr>
<tr>
<td>U U</td>
<td>Internal pulled-up with 75KΩ resistor</td>
<td>L L</td>
<td>Active low</td>
</tr>
<tr>
<td>O8 O8</td>
<td>Output with driving current 8mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<Notation> IUL : Input PIN with 75KΩ pull-up resistor. Active low
OM : Multi-functional Output PIN

1.1 PIN Layout

![PIN Layout Diagram](image)

Fig 1. PIN Layout
1.2 Configuration Signals

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST_MODE[3:0]</td>
<td>ID</td>
<td><strong>W5300 mode select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>It configures PHY mode and factory test mode of W5300.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST_MODE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>Internal PHY Mode (Normal Operation)</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>External PHY Mode with Crystal clock</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>External PHY Mode with Oscillator clock</td>
</tr>
<tr>
<td>Others</td>
<td>Reserved (Factory Test Mode)</td>
</tr>
</tbody>
</table>

At the external PHY mode, Clock input pin is changed by clock source. Refer to “1.7 Clock Signals”.

<table>
<thead>
<tr>
<th>OP_MODE[2:0]</th>
<th>ID</th>
<th><strong>Internal PHY operation control mode</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It configures the operation mode of internal PHY.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OP_MODE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 1 0</td>
<td>Normal Operation Mode, Recommended</td>
</tr>
<tr>
<td>0 0 0</td>
<td>Auto-negotiation enable with all capabilities</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Auto-negotiation with 100 BASE-TX FDX/HDX ability</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Auto-negotiation with 10 BASE-T FDX/HDX ability</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Manual selection of 100 BASE-TX FDX</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Manual selection of 100 BASE-TX HDX</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Manual selection of 10 BASE-T FDX</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Manual selection of 10 BASE-T HDX</td>
</tr>
</tbody>
</table>

cf> FDX : Full-duplex, HDX : Half-duplex

The setting value is latched after hardware reset.
### 1.3 Host Interface Signals

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/RESET</td>
<td>IL</td>
<td><strong>RESET</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hardware Reset Signal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It initializes W5300. RESET should be held at least 2us after low assert, and wait for at least 10ms after high de-assert in order for PLL logic to be stable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to RESET timing of “7 Electrical Specification”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W5300 does not support Power-On-Reset. Therefore, it should be manually designed in the target system.</td>
</tr>
<tr>
<td>BIT16EN</td>
<td>IU</td>
<td><strong>16/8 BIT DATA BUS SELECT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>High : 16 bit data bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low : 8 bit data bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It determinates data bus width of W5300.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At reset time, it is latched in 15th Bit(‘BW’)of Mode register(MR).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After reset, its change is ignored. It means data bus width can’t be changed after reset. When using 8 bit data bus, it should be connected to ground.</td>
</tr>
<tr>
<td>ADDR9-0</td>
<td>I</td>
<td><strong>ADDRESS</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>System address bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These are selected by host interface mode and data bus width of W5300. When using 16 bit data bus, ADDR0 is internally ignored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to “6.External Interface”.</td>
</tr>
<tr>
<td>DATA[15:8]</td>
<td>IO</td>
<td><strong>DATA</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>System high data bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These are used for read/write operation of W5300 register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In case of using 8 bit data bus, These are driven as High-Z.</td>
</tr>
<tr>
<td>DATA[7:0]</td>
<td>IO8</td>
<td><strong>DATA</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>System low data bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These are used for read/write operation of W5300 register.</td>
</tr>
<tr>
<td>/CS</td>
<td>IL</td>
<td><strong>CHIP SELECT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chip select signal.</td>
</tr>
<tr>
<td>Pin</td>
<td>State</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/CS</td>
<td>IL</td>
<td>Host selects W5300 at the W5300 read/write operation. When /CS is de-asserted high, DATA[15:0] are driven as High-Z.</td>
</tr>
<tr>
<td>/RD</td>
<td>IL</td>
<td><strong>READ ENABLE</strong>&lt;br&gt;Read enable signal. Host reads W5300 register addressed by ADDR[9:0] through DATA[15:0].</td>
</tr>
<tr>
<td>/INT</td>
<td>OL</td>
<td><strong>INTERRUPT</strong>&lt;br&gt;Interrupt Request Signal. It is asserted low when interrupt (connected, disconnected, data received, data sent or timeout) occurs on operating. When interrupt service is completed by host and Interrupt register(IR) is cleared by host, it is de-asserted high. Refer to IR, Interrupt Mask Register(IMR), SOCKETn Interrupt Register(Sn_IR), SOCKETn Interrupt Mask Register(Sn_IMR).</td>
</tr>
<tr>
<td>BRDY[3:0]</td>
<td>O</td>
<td><strong>Buffer Ready Indicator</strong>&lt;br&gt;These PIN are configured with SOCKET number, memory Type, and buffer depth by user. When TX free or RX received size of the specified SOCKET is same or greater than the configured buffer depth, these PIN signals asserts high or low. Refer to Pn_BRDYR &amp; Pn_DPTHR in “4.3 COMMON Registers”.</td>
</tr>
</tbody>
</table>
1.4 Media Interface Signals

Media (10Mbps/100Mbps) interface signals are used in internal PHY mode (TEST_Mode[3:0] = “0000”). Refer to “1.2 Configuration Signals”.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXIP</td>
<td>I</td>
<td>RXIP/RXIN Signal Pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential receive Input signal pair.</td>
</tr>
<tr>
<td>RXIN</td>
<td>I</td>
<td>Receive data from the media. This signal pair needs 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>termination resistors 50Ω(±1%) and 1 capacitor 0.1uF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for better impedance matching, and this resistor/capacitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pair is located near magnetic(transformer). If not used,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>connect to ground.</td>
</tr>
<tr>
<td>TXOP</td>
<td>O</td>
<td>TXOP/TXON Signal Pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential transmit output signal pair.</td>
</tr>
<tr>
<td>TXON</td>
<td>O</td>
<td>Transmits data to the media. This signal pair needs 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>termination resistors 50Ω(±1%) and 1 capacitor 0.1uF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for better impedance matching, and this resistor/capacitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pair should be located near W5300. If not used, just</td>
</tr>
<tr>
<td></td>
<td></td>
<td>let them float.</td>
</tr>
<tr>
<td>RSET_BG</td>
<td>O</td>
<td>Off-chip Resistor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This pin should be pulled-down with 12.3 kΩ±1% resistor.</td>
</tr>
</tbody>
</table>

For the better performance,

1. Make the length of RXIP/RXIN signal pair (RX) same if possible.
2. Make the length of TXOP/TXON signal pair (TX) same if possible.
3. Locate the RXIP and RXIN signal as near as possible.
4. Locate the TXOP and TXON signal as near as possible.
5. Locate the RX and TX signal pairs far from noisy signals such as bias resistor or crystal.

For the detailed information refer to “W5100 Layout Guide.pdf”
1.5 MII interface signal for external PHY
MII interface signals are for interfacing to external PHY instead of the internal PHY of W5300. These signals can be used at the external PHY mode (TEST_Mode[3:0] = “0001” or “0010”). Refer to “1.2 Configuration Signals”.
At the internal PHY mode, just let them float because the pins except for multi-function pins are internal pulled-down.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/TXLED(MII_TXEN)</td>
<td>OMH</td>
<td>Transmit Act LED / Transmit Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This signal indicates the presence of transmit packet on the MII_TXD[3:0]. It is asserted high when the first nibble data of transmit packet is valid on MII_TXD[3:0] and is de-asserted low after the last nibble data of transmit packet is clocked out on MII_TXD[3:0].</td>
</tr>
<tr>
<td>/RXLED(MII_TXD3)</td>
<td>OM</td>
<td>/RXLED./COLLED./LEDFDX./SPDLED / Transmit data output</td>
</tr>
<tr>
<td>/COLLED(MII_TXD2)</td>
<td></td>
<td>The transmit packet is synchronized with MII_TXC clock and output to external PHY in nibble unit.</td>
</tr>
<tr>
<td>/FDXLED(MII_TXD1)</td>
<td></td>
<td>MII_TXD3 is the Most Significant Bit (MSB).</td>
</tr>
<tr>
<td>/SPDLED(MII_TXD0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MII_TXC</td>
<td>ID</td>
<td>Transmit Clock Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is a continuous transmit clock from the external PHY. It is 25MHz at the 100BaseTX and 2.5MHz at the 10 BaseT. Transmit clock is used as timing reference of MII_TXD[3:0] and used for network operation clock (NIC_CLK). Rising Edge Sensitive.</td>
</tr>
<tr>
<td>MII_CRS</td>
<td>IDH</td>
<td>Carrier Sense</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is signal to notify the link traffic of the media. If carrier of media is not idle (carrier present), it is asserted high.</td>
</tr>
<tr>
<td>MII_COL</td>
<td>IDH</td>
<td>Collision Detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When collision is detected on the media, it is asserted high. It is valid at the half-duplex and ignored at the full-duplex. Asynchronous signal.</td>
</tr>
<tr>
<td>MII_RXD3</td>
<td>ID</td>
<td>Receive Data Input</td>
</tr>
<tr>
<td>---------</td>
<td>----</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When MII_RXDV is high, the received packet is synchronized with MII_RXC and inputs in nibble unit. MII_RXD3 is MSB.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MII_RXD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MII_RXD1</td>
</tr>
<tr>
<td>MII_RXD0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MII_RXDV</th>
<th>ID</th>
<th>Receive Data Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This signal indicates the presence of received packet from MII_RXD[3:0]. It is asserted high when the first nibble data of the received packet is valid on MII_RXD[3:0] and is de-asserted low after the last nibble data of receive packet clocked in on MII_RXD[3:0]. It is valid when MII_RXC is at rising edge.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MII_RXC</th>
<th>ID</th>
<th>Receive Clock Input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It is continuous receive clock from the external PHY. It is 25MHz at the 100Base TX and 2.5MHz at the 10BaseT. Receive clock is used for timing reference of MII_RXD[3:0] and MII_RXDV. Rising Edge Sensitive.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/FDX</th>
<th>IDL</th>
<th>Full-Duplex Select</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0 : Full-duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : Half-duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is input signal from PHY that indicates link status of external PHY. Most of PHYs support auto-negotiation and notifies the result to network indicator LED or other signals. It can be connected to those signals and also it can be configurable manually by connecting high or low.</td>
</tr>
</tbody>
</table>

Recommend for the better performance.

1. MII interface signal line length should not be more than 25cm if possible.
2. The length of MII_TXD[3:0] should be same if possible.
3. The length of MII_RXD[3:0] should be same if possible.
4. The length of MII_TXC should not be longer than MII_TXD[3:0] signal line by 2.5cm.
5. The length of MII_RXC should not be longer than MII_RXD[3:0] signal line by 2.5cm.
### 1.6 Network Indicator LED Signals

The signals except for LINKLED, are used as multi-function PIN according to the configuration of TEST_MODE[3:0]. When using those signals as network indicator signals, internal PHY mode (TEST_MODE[3:0]=“0000”) should be configured.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINKLED</td>
<td>OL</td>
<td>Link LED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It indicates the link status of media(10/100M).</td>
</tr>
</tbody>
</table>

<Notice> When /RESET is low, LINKLED goes to low (i.e. turn off). But in case that the “Internal PHY operation control mode” is configured as “Manual selection of 100 BASE-TX FDX/HDX” (OP_MODE[2:0] = “1xx”), LINKLED stays at the previous link status before /RESET is low.

After /RESET goes to high, then it indicates the link status of media(10/100M) properly.

<table>
<thead>
<tr>
<th>/TXLED(MII_TXEN)</th>
<th>OML</th>
<th>Transmit activity LED/Transmit Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It notifies the output of transmit data through TXOP/TXON (Transmit Activity).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/RXLED(MII_TXD3)</th>
<th>OML</th>
<th>Receive activity LED/Transmit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It notifies the input of receive data from RXIP/RXIN (Receive Activity)</td>
</tr>
</tbody>
</table>

cf> By binding /TXLED and /RXLED signals with ‘AND’ gate, it can be used for network activity LED.

<table>
<thead>
<tr>
<th>/COLLED(MII_TXD2)</th>
<th>OML</th>
<th>Collision LED/Transmit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It notifies when collisions occur.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is valid at half-duplex, and is ignored at full-duplex.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/FDXLED(MII_TXD1)</th>
<th>OML</th>
<th>Full duplex LED/Transmit Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It outputs low at the full-duplex and outputs high at the half-duplex according to auto-negotiation or manual configuration of OP_MODE[2:0].</td>
</tr>
</tbody>
</table>
1.7 Clock Signals
For the clock source of W5300, either a crystal or an oscillator may be used. 25MHz frequency from the clock source is created to 150MHz frequency using internal PLL logic. This 150MHz frequency is used for PLL_CLK(Period 6.67ns) and W5300 core operation clock.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XTLP</td>
<td></td>
<td>25MHz crystal input/output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25MHz parallel-resonant crystal is used with matching capacitor for internal oscillator stabilization.</td>
</tr>
<tr>
<td>XTLN</td>
<td></td>
<td>Refer to “Clock Characteristic” of “7.Electrical Specifications”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These can be used for internal PHY mode(TEST_MODE[3:0]=&quot;0000&quot;) or external PHY mode with crystal clock (TEST_MODE[3:0]=&quot;0001&quot;).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When using oscillator at the internal PHY mode, be sure to use 1.8V level oscillator and connect only to XTLP. And let be float XTLN.</td>
</tr>
<tr>
<td>OSC25I</td>
<td>I</td>
<td>25MHz Oscillator input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is used only in external PHY mode with oscillator clock (TEST_MODE[3:0]=&quot;0010&quot;). In order to prevent the leakage current, be sure to keep XTLP high and float XTLN, and use 1.8v level oscillator.</td>
</tr>
</tbody>
</table>

1.8 Power Supply Signals

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC3A3</td>
<td>Power</td>
<td>3.3V power supply for Analog part</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Be sure to connect 10uF tantalum capacitor between VCC343 and GNDA in order to prevent power compensation.</td>
</tr>
<tr>
<td>VCC3V3</td>
<td>Power</td>
<td>3.3V power supply for Digital part</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between each VCC and GND, 0.1uF decoupling capacitor can be selectively connected. VCC3V3 can be separated to 1uH ferrite bead and connected to VCC3A3.</td>
</tr>
<tr>
<td>VCC1A8</td>
<td>Power</td>
<td>1.8V power supply for Analog part</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Be sure to connect a 10uF tantalum capacitor and 0.1uF capacitor</td>
</tr>
</tbody>
</table>
between VCC1A8 and GNDA for core power noise filtering.

<table>
<thead>
<tr>
<th>VCC1V8</th>
<th>Power</th>
<th>1.8V power supply for Digital part</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Between each VCC and GND, 0.1uF decoupling capacitor can be selectively connected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GNDA</th>
<th>Ground</th>
<th>Analog ground</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Make analogue ground plane as wide as possible when designing the PCB layout.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GND</th>
<th>Ground</th>
<th>Digital ground</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Make digital ground plane as wide as possible when designing the PCB layout.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1V8O</th>
<th>O</th>
<th>1.8V regulator output voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1.8V/150mA power created by internal power regulator, is used for core operation power (VCC1A8, VCC1V8).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Be sure to connect 3.3uF tantalum capacitor between 1V8O and GND for output frequency compensation, and selectively connect 10uF capacitor for high frequency noise decoupling. 1V8O is connected to VCC1V8, separated to 1uH ferrite bead and connected to VCC1A8.</td>
</tr>
</tbody>
</table>

<Notice> 1V8O is the power for W5300 core operation. It should not be connected to the power of other devices.

---

**Fig. 2. Power Design**
Recommend for power design.

1. Locate decoupling capacitor as close as possible to W5300.
2. Use ground plane as wide as possible.
3. If ground plane width is adequate, having a separate analog ground plane and digital ground plane is good practice.

If ground plane is not wide, design analog and digital ground planes as a single ground plane, rather than separate them.
2. System Memory Map

According to the host interface, W5300 supports direct address mode and indirect address mode.

The direct address mode is that the target host system can directly access W5300 registers after mapping the registers to T.M.S (Target host system Memory-mapped I/O Space).

Direct address mode memory map is composed of Mode register(MR), COMMON registers, and SOCKET registers. Those registers are mapped in T.M.S sequentially increasing by 2 bytes from the BA (Base Address) of T.M.S. Using the mapping address, the target host system can directly access MR, COMMON registers and SOCKET registers. To use the direct address mode, total 0x400 bytes are required for memory space.

In indirect address mode, target host system indirectly accesses COMMON registers and SOCKET registers by using IDM_AR (Indirect Mode Address Register) and IDM_DR (Indirect Mode Data Register) which are just only directly mapped in T.M.S together with MR.

Indirect address mode memory map is composed of direct accessible MR, IDM_AR, IDM_DR and indirect accessible COMMON & SOCKET registers. Only MR, IDM_AR and IDM_DR are mapped in T.M.S sequentially increasing by 2 Bytes from BA of T.M.S, but COMMON & SOCKET registers are not mapped in T.M.S because those register can be accessed indirectly using IDM_AR & IDM_DR. To use the indirect address mode, just 0x06 bytes are required for memory space.

When target host system access Interrupt register(IR) of COMMON registers at the indirect address mode, it is processed as below:

- **Host Write**: Set IDM_AR to 0x0002, IR address (IDM_AR = 0x0002)
  - Set IDM_DR to 0xFFFF (IDM_DR = 0xFFFF)

- **Host Read**: Set IDM_AR to 0x0002, IR address (IDM_AR = 0x0002)
  - Read IDM_DR and save as Value (Value = IDM_DR)

The host interface mode of W5300 is decided according to the value of ‘IND’ bit (0th bit) of MR.

- MR(0) = ‘0’ => Direct address mode
- MR(0) = ‘1’ => Indirect address mode

The memory map of each address mode is as below:
### High-performance Internet Connectivity Solution

**W5300**

### Direct Address Mode (MR(0) = ‘0’) 

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA + 0x000</td>
<td>MR (Mode Reg)</td>
</tr>
<tr>
<td>BA + 0x002</td>
<td>IR (Interrupt Reg)</td>
</tr>
<tr>
<td>BA + 0x004</td>
<td>IMR (Interrupt Mask Reg)</td>
</tr>
<tr>
<td>BA + 0x0FE</td>
<td>IDR (ID Reg)</td>
</tr>
<tr>
<td>BA + 0x100</td>
<td>Reserved</td>
</tr>
<tr>
<td>BA + 0x1FE</td>
<td>Reserved</td>
</tr>
<tr>
<td>BA + 0x200</td>
<td>S0_MR (SOCKET0 Mode Reg)</td>
</tr>
<tr>
<td>BA + 0x240</td>
<td>S1_MR</td>
</tr>
<tr>
<td>BA + 0x280</td>
<td>S2_MR</td>
</tr>
<tr>
<td>BA + 0x2C0</td>
<td>S3_MR</td>
</tr>
<tr>
<td>BA + 0x300</td>
<td>S4_MR</td>
</tr>
<tr>
<td>BA + 0x340</td>
<td>S5_MR</td>
</tr>
<tr>
<td>BA + 0x380</td>
<td>S6_MR</td>
</tr>
<tr>
<td>BA + 0x3C0</td>
<td>S7_MR</td>
</tr>
</tbody>
</table>

### Indirect Address Mode (MR(0) = ‘1’) 

- **T.M.S**: BA + 0x000, BA + 0x002, BA + 0x004
- **W5300**: Internal Memory-Mapped Space (W.M.S)
- **BA + 0x100**
- **BA + 0x1FE**
- **BA + 0x200**
- **BA + 0x240**
- **BA + 0x280**
- **BA + 0x2C0**
- **BA + 0x300**
- **BA + 0x340**
- **BA + 0x380**
- **BA + 0x3C0**
- **BA + 0x3FF**

**Fig 3. Memory Map**
3. W5300 Registers

W5300 register is composed of MR(to decide direct or indirect address mode), IDM_AR & IDM_DR(only used at the indirect address mode) and COMMON registers and SOCKET registers.

MR, IDM_AR, and IDM_DR register are mapped in T.M.S. COMMON & SOCKET registers are mapped in T.M.S or W.M.S (W5300 internal Memory Space) according to address mode.

All W5300 registers are 1Byte, 2Bytes, 4Bytes or 6Bytes. According to data bus width of target host system, the access is processed – 2bytes address offset at the 16bit data bus and 1 byte address offset at the 8bit data bus.

When mapping W5300 registers in T.M.S, the physical T.M.S address of W5300 register is calculated as below.

\[
\text{Physical Address of W5300 Reg} = \text{Base Address of T.M.S} + \text{Address offset of W5300 Reg}
\]

The byte ordering of W5300 registers is big-endian – low address byte is used as the most significant byte.

[Register Notation]

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>MR register</td>
</tr>
<tr>
<td>MR0</td>
<td>Low address register of MR (Address offset - 0x000), Most significant byte</td>
</tr>
<tr>
<td>MR1</td>
<td>High address register of MR (Address offset – 0x001), Least significant byte</td>
</tr>
<tr>
<td>MR(15:5)</td>
<td>11 bit (from 15th bit to 5th bit of MR register)</td>
</tr>
<tr>
<td>MR(0)</td>
<td>0th bit of MR register, 0th bit of MR1</td>
</tr>
<tr>
<td>MR(13)</td>
<td>13th bit of MR register, 5th bit of MR0</td>
</tr>
<tr>
<td>MR0(7)</td>
<td>15th bit of MR register, Most significant bit of MR0</td>
</tr>
<tr>
<td>MR(DWB)</td>
<td>MR[\text{DWB}] DWB bit (DWB : Bit Symbol)</td>
</tr>
</tbody>
</table>

SHAR : Source Hardware Address Register

<table>
<thead>
<tr>
<th>SHAR0</th>
<th>1st address register of SHAR (Address offset – 0x008)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHAR1</td>
<td>2nd address register of SHAR (Address offset – 0x009)</td>
</tr>
<tr>
<td>SHAR2</td>
<td>3rd address register of SHAR (Address offset – 0x00A)</td>
</tr>
<tr>
<td>SHAR3</td>
<td>4th address register of SHAR (Address offset – 0x00B)</td>
</tr>
<tr>
<td>SHAR4</td>
<td>5th Address register of SHAR (Address offset – 0x00C)</td>
</tr>
<tr>
<td>SHAR5</td>
<td>6th address register of SHAR (Address offset – 0x00D)</td>
</tr>
</tbody>
</table>
### 3.1 Mode Register

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>MR</td>
<td>Mode Register</td>
</tr>
<tr>
<td>0x000</td>
<td>MR0</td>
<td></td>
</tr>
<tr>
<td>0x001</td>
<td>MR1</td>
<td></td>
</tr>
</tbody>
</table>

### 3.2 Indirect Mode Registers

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x002</td>
<td>IDM_AR</td>
<td>Indirect Mode Address Register</td>
</tr>
<tr>
<td>0x003</td>
<td>IDM_AR</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>IDM_DR</td>
<td>Indirect Mode Data Register</td>
</tr>
<tr>
<td>0x005</td>
<td>IDM_DR</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3 COMMON registers

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x002</td>
<td>IR</td>
<td>Interrupt Register</td>
</tr>
<tr>
<td>0x003</td>
<td>IR0</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>IMR</td>
<td>Interrupt Mask Register</td>
</tr>
<tr>
<td>0x005</td>
<td>IMR0</td>
<td></td>
</tr>
<tr>
<td>0x006</td>
<td>IRM1</td>
<td></td>
</tr>
<tr>
<td>0x007</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>SHAR</td>
<td>Source Hardware Address Register</td>
</tr>
<tr>
<td>0x009</td>
<td>SHAR0</td>
<td></td>
</tr>
<tr>
<td>0x00A</td>
<td>SHAR2</td>
<td></td>
</tr>
<tr>
<td>0x00B</td>
<td>SHAR3</td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>SHAR4</td>
<td></td>
</tr>
<tr>
<td>0x00D</td>
<td>SHAR5</td>
<td></td>
</tr>
<tr>
<td>0x00E</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x00F</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>GAR</td>
<td>Gateway Address Register</td>
</tr>
<tr>
<td>0x011</td>
<td>GAR0</td>
<td></td>
</tr>
<tr>
<td>0x012</td>
<td>GAR2</td>
<td></td>
</tr>
<tr>
<td>0x013</td>
<td>GAR3</td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>0x014</td>
<td>SUBR</td>
<td>Subnet Mask Register</td>
</tr>
<tr>
<td>0x016</td>
<td>SUBR2</td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>SIPR</td>
<td>Source IP Address Register</td>
</tr>
<tr>
<td>0x01A</td>
<td>SIPR2</td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>RTR</td>
<td>Retransmission Timeout-value Register</td>
</tr>
<tr>
<td>0x01E</td>
<td>RCR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x020</td>
<td>TMS01R</td>
<td>Transmit Memory Size Register of SOCKET0</td>
</tr>
<tr>
<td>0x021</td>
<td>TMS1R</td>
<td>Transmit Memory Size Register of SOCKET1</td>
</tr>
<tr>
<td>0x022</td>
<td>TMS23R</td>
<td>Transmit Memory Size Register of SOCKET2</td>
</tr>
<tr>
<td>0x023</td>
<td>TMS3R</td>
<td>Transmit Memory Size Register of SOCKET3</td>
</tr>
<tr>
<td>0x024</td>
<td>TMS45R</td>
<td>Transmit Memory Size Register of SOCKET4</td>
</tr>
<tr>
<td>0x025</td>
<td>TMS5R</td>
<td>Transmit Memory Size Register of SOCKET5</td>
</tr>
<tr>
<td>0x026</td>
<td>TMS67R</td>
<td>Transmit Memory Size Register of SOCKET6</td>
</tr>
<tr>
<td>0x027</td>
<td>TMS7R</td>
<td>Transmit Memory Size Register of SOCKET7</td>
</tr>
<tr>
<td>0x028</td>
<td>RMS01R</td>
<td>Receive Memory Size Register of SOCKET0</td>
</tr>
<tr>
<td>0x029</td>
<td>RMS1R</td>
<td>Receive Memory Size Register of SOCKET1</td>
</tr>
<tr>
<td>0x02A</td>
<td>RMS23R</td>
<td>Receive Memory Size Register of SOCKET2</td>
</tr>
<tr>
<td>0x02B</td>
<td>RMS3R</td>
<td>Receive Memory Size Register of SOCKET3</td>
</tr>
<tr>
<td>0x02C</td>
<td>RMS45R</td>
<td>Receive Memory Size Register of SOCKET4</td>
</tr>
<tr>
<td>0x02D</td>
<td>RMS5R</td>
<td>Receive Memory Size Register of SOCKET5</td>
</tr>
<tr>
<td>0x02E</td>
<td>RMS67R</td>
<td>Receive Memory Size Register of SOCKET6</td>
</tr>
<tr>
<td>0x02F</td>
<td>RMS7R</td>
<td>Receive Memory Size Register of SOCKET7</td>
</tr>
<tr>
<td>0x030</td>
<td>MTYPER</td>
<td>Memory Block Type Register</td>
</tr>
<tr>
<td>0x031</td>
<td>MTYPER0</td>
<td></td>
</tr>
<tr>
<td>0x032</td>
<td>PATR</td>
<td>PPPoE Authentication Register</td>
</tr>
<tr>
<td>0x033</td>
<td>PATR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PATR1</td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>0x034</td>
<td>0x034</td>
<td>0x035</td>
</tr>
<tr>
<td>0x036</td>
<td>PTIMER</td>
<td>PTIMER0 0x036</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PTIMER1 0x037</td>
</tr>
<tr>
<td>0x038</td>
<td>PMAGICR</td>
<td>PMAGICR0 0x038</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMAGICR1 0x039</td>
</tr>
<tr>
<td>0x03A</td>
<td>0x03A</td>
<td>0x03B</td>
</tr>
<tr>
<td>0x03C</td>
<td>PSIDR</td>
<td>PSIDR0 0x03C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSIDR1 0x03D</td>
</tr>
<tr>
<td>0x03E</td>
<td>0x03E</td>
<td>0x03F</td>
</tr>
<tr>
<td>0x040</td>
<td>PDHAR</td>
<td>PDHAR0 0x040</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDHAR1 0x041</td>
</tr>
<tr>
<td>0x042</td>
<td>PDHAR2</td>
<td>PDHAR2 0x042</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDHAR3 0x043</td>
</tr>
<tr>
<td>0x044</td>
<td>PDHAR4</td>
<td>PDHAR4 0x044</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDHAR5 0x045</td>
</tr>
<tr>
<td>0x046</td>
<td>0x046</td>
<td>0x047</td>
</tr>
<tr>
<td>0x048</td>
<td>UIPR</td>
<td>UIPR0 0x048</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UIPR1 0x049</td>
</tr>
<tr>
<td>0x04A</td>
<td>UIPR2</td>
<td>UIPR2 0x04A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UIPR3 0x04B</td>
</tr>
<tr>
<td>0x04C</td>
<td>UPORTR</td>
<td>UPORT0 0x04C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPORT1 0x04D</td>
</tr>
<tr>
<td>0x04E</td>
<td>FMTUR</td>
<td>FMTUR0 0x04E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FMTUR1 0x04F</td>
</tr>
<tr>
<td>0x050</td>
<td>0x050</td>
<td>0x051</td>
</tr>
<tr>
<td>0x05E</td>
<td>0x05E</td>
<td>0x060</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>16Bit</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>0x060</td>
<td>P0_BRDYR</td>
<td>0x060</td>
</tr>
<tr>
<td></td>
<td>P0_BDPTHBR</td>
<td>0x062</td>
</tr>
<tr>
<td>0x064</td>
<td>P1_BRDYR</td>
<td>0x064</td>
</tr>
<tr>
<td></td>
<td>P1_BDPTHBR</td>
<td>0x066</td>
</tr>
<tr>
<td>0x068</td>
<td>P2_BRDYR</td>
<td>0x068</td>
</tr>
<tr>
<td></td>
<td>P2_BDPTHBR</td>
<td>0x06A</td>
</tr>
<tr>
<td>0x06C</td>
<td>P3_BRDYR</td>
<td>0x06C</td>
</tr>
<tr>
<td></td>
<td>P3_BDPTHBR</td>
<td>0x06E</td>
</tr>
<tr>
<td>0x070</td>
<td>IDR</td>
<td>0x070</td>
</tr>
<tr>
<td>0xFC</td>
<td>IDR</td>
<td>0x0FC</td>
</tr>
<tr>
<td>0xFE</td>
<td>IDR</td>
<td>0x0FE</td>
</tr>
</tbody>
</table>
### 3.4 SOCKET registers

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x200</td>
<td>S0_MR</td>
<td>SOCKET0 Mode Register</td>
</tr>
<tr>
<td>0x201</td>
<td>S0_MR0</td>
<td></td>
</tr>
<tr>
<td>0x202</td>
<td>S0_CR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x203</td>
<td>S0_CR0</td>
<td>SOCKET0 Command Register</td>
</tr>
<tr>
<td>0x204</td>
<td>S0_IMR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x205</td>
<td>S0_IMR0</td>
<td>SOCKET0 Interrupt Mask Register</td>
</tr>
<tr>
<td>0x206</td>
<td>S0_IR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x207</td>
<td>S0_IR0</td>
<td>SOCKET0 Interrupt Register</td>
</tr>
<tr>
<td>0x208</td>
<td>S0_SSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x209</td>
<td>S0_SSR0</td>
<td>SOCKET0 SOCKET Status Register</td>
</tr>
<tr>
<td>0x20A</td>
<td>S0_PORTR</td>
<td>SOCKET0 Source Port Register</td>
</tr>
<tr>
<td>0x20B</td>
<td>S0_PORTR0</td>
<td></td>
</tr>
<tr>
<td>0x20C</td>
<td>S0_DHAR</td>
<td>SOCKET0 Destination Hardware</td>
</tr>
<tr>
<td>0x20D</td>
<td>S0_DHAR0</td>
<td>Address Register</td>
</tr>
<tr>
<td>0x20E</td>
<td>S0_DHAR2</td>
<td></td>
</tr>
<tr>
<td>0x20F</td>
<td>S0_DHAR2</td>
<td></td>
</tr>
<tr>
<td>0x210</td>
<td>S0_DHAR4</td>
<td></td>
</tr>
<tr>
<td>0x211</td>
<td>S0_DHAR5</td>
<td></td>
</tr>
<tr>
<td>0x212</td>
<td>S0_DPORTR</td>
<td>SOCKET0 Destination Port Register</td>
</tr>
<tr>
<td>0x213</td>
<td>S0_DPORTR0</td>
<td></td>
</tr>
<tr>
<td>0x214</td>
<td>S0_DIPR</td>
<td>SOCKET0 Destination IP Address</td>
</tr>
<tr>
<td>0x215</td>
<td>S0_DIPR0</td>
<td>Register</td>
</tr>
<tr>
<td>0x216</td>
<td>S0_DIPR2</td>
<td></td>
</tr>
<tr>
<td>0x217</td>
<td>S0_DIPR3</td>
<td></td>
</tr>
<tr>
<td>0x218</td>
<td>S0_MSSR</td>
<td>SOCKET0 Maximum Segment Size</td>
</tr>
<tr>
<td>0x219</td>
<td>S0_MSSR0</td>
<td>Register</td>
</tr>
<tr>
<td>0x21A</td>
<td>S0_PORTOR</td>
<td>SOCKET0 Keep Alive Time Register</td>
</tr>
<tr>
<td>0x21B</td>
<td>S0_PROTOR</td>
<td>SOCKET0 Protocol Number Register</td>
</tr>
<tr>
<td>0x21C</td>
<td>S0_TOSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x21D</td>
<td>S0_TOSR0</td>
<td>SOCKET0 TOS Register</td>
</tr>
<tr>
<td>0x21E</td>
<td>S0_TTLR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x21F</td>
<td>S0_TTLR0</td>
<td>SOCKET0 TTL Register</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>0x220</td>
<td>0x220</td>
<td>S0_TX_WRSR0</td>
</tr>
<tr>
<td>0x221</td>
<td>0x221</td>
<td>S0_TX_WRSR1</td>
</tr>
<tr>
<td>0x222</td>
<td>0x222</td>
<td>S0_TX_WRSR2</td>
</tr>
<tr>
<td>0x223</td>
<td>0x223</td>
<td>S0_TX_WRSR3</td>
</tr>
<tr>
<td>0x224</td>
<td>0x224</td>
<td>S0_TX_FSR0</td>
</tr>
<tr>
<td>0x225</td>
<td>0x225</td>
<td>S0_TX_FSR1</td>
</tr>
<tr>
<td>0x226</td>
<td>0x226</td>
<td>S0_TX_FSR2</td>
</tr>
<tr>
<td>0x227</td>
<td>0x227</td>
<td>S0_TX_FSR3</td>
</tr>
<tr>
<td>0x228</td>
<td>0x228</td>
<td>S0_RX_RSR0</td>
</tr>
<tr>
<td>0x229</td>
<td>0x229</td>
<td>S0_RX_RSR1</td>
</tr>
<tr>
<td>0x22A</td>
<td>0x22A</td>
<td>S0_RX_RSR2</td>
</tr>
<tr>
<td>0x22B</td>
<td>0x22B</td>
<td>S0_RX_RSR3</td>
</tr>
<tr>
<td>0x22C</td>
<td>0x22C</td>
<td>S0_FRAGR0</td>
</tr>
<tr>
<td>0x22D</td>
<td>0x22D</td>
<td>S0_FRAGR1</td>
</tr>
<tr>
<td>0x22E</td>
<td>0x22E</td>
<td>S0_TX_FIFOR0</td>
</tr>
<tr>
<td>0x22F</td>
<td>0x22F</td>
<td>S0_TX_FIFOR1</td>
</tr>
<tr>
<td>0x230</td>
<td>0x230</td>
<td>S0_RX_FIFOR0</td>
</tr>
<tr>
<td>0x231</td>
<td>0x231</td>
<td>S0_RX_FIFOR1</td>
</tr>
<tr>
<td>0x232</td>
<td>0x232</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x233</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x23E</td>
<td>0x23E</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x23F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>0x240</td>
<td>S1_MR</td>
<td>SOCKET1 Mode Register</td>
</tr>
<tr>
<td></td>
<td>S1_MR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_MR1</td>
<td></td>
</tr>
<tr>
<td>0x242</td>
<td>S1_CR</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_CR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_CR1</td>
<td>SOCKET1 Command Register</td>
</tr>
<tr>
<td>0x244</td>
<td>S1_IMR</td>
<td>SOCKET1 Interrupt Mask Register</td>
</tr>
<tr>
<td></td>
<td>S1_IMR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_IMR1</td>
<td></td>
</tr>
<tr>
<td>0x246</td>
<td>S1_IR</td>
<td>SOCKET1 Interrupt Register</td>
</tr>
<tr>
<td></td>
<td>S1_IR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_IR1</td>
<td></td>
</tr>
<tr>
<td>0x248</td>
<td>S1_SSR</td>
<td>Socket1 Status Register</td>
</tr>
<tr>
<td></td>
<td>S1_SSR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_SSR1</td>
<td></td>
</tr>
<tr>
<td>0x24A</td>
<td>S1_PORTR</td>
<td>SOCKET1 Source Port Register</td>
</tr>
<tr>
<td></td>
<td>S1_PORTR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_PORTR1</td>
<td></td>
</tr>
<tr>
<td>0x24C</td>
<td>S1_DHAR</td>
<td>SOCKET1 Destination Hardware Address Register</td>
</tr>
<tr>
<td></td>
<td>S1_DHAR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_DHAR1</td>
<td></td>
</tr>
<tr>
<td>0x24E</td>
<td>S1_DHAR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_DHAR3</td>
<td></td>
</tr>
<tr>
<td>0x250</td>
<td>S1_DHAR4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_DHAR5</td>
<td></td>
</tr>
<tr>
<td>0x252</td>
<td>S1_DPORTR</td>
<td>SOCKET1 Destination Port Register</td>
</tr>
<tr>
<td></td>
<td>S1_DPORTR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_DPORTR1</td>
<td></td>
</tr>
<tr>
<td>0x254</td>
<td>S1_DIPR</td>
<td>SOCKET1 Destination IP Address Register</td>
</tr>
<tr>
<td></td>
<td>S1_DIPR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_DIPR1</td>
<td></td>
</tr>
<tr>
<td>0x256</td>
<td>S1_DIPR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_DIPR3</td>
<td></td>
</tr>
<tr>
<td>0x258</td>
<td>S1_MSSR</td>
<td>SOCKET1 Maximum Segment Size Register</td>
</tr>
<tr>
<td></td>
<td>S1_MSSR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_MSSR1</td>
<td></td>
</tr>
<tr>
<td>0x25A</td>
<td>S1_PORTOR</td>
<td>SOCKET1 Keep Alive Time Register</td>
</tr>
<tr>
<td></td>
<td>S1_KPALVTR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_PROTOR</td>
<td></td>
</tr>
<tr>
<td>0x25C</td>
<td>S1_TOSR</td>
<td>SOCKET1 TOS Register</td>
</tr>
<tr>
<td></td>
<td>S1_TOSR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_TOSR1</td>
<td></td>
</tr>
<tr>
<td>0x25E</td>
<td>S1_TTLR</td>
<td>SOCKET1 TTL Register</td>
</tr>
<tr>
<td></td>
<td>S1_TTLR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S1_TTLR1</td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>0x260</td>
<td>S1_TX_WRSR</td>
<td>S1_TX_WRSR0  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_TX_WRSR1</td>
<td>SOCKET1 TX Write Size Register</td>
</tr>
<tr>
<td>0x262</td>
<td>S1_TX_WRSR2</td>
<td>S1_TX_WRSR2  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_TX_WRSR3</td>
<td>SOCKET1 TX Write Size Register</td>
</tr>
<tr>
<td>0x264</td>
<td>S1_TX_FSR</td>
<td>S1_TX_FSR0  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_TX_FSR1</td>
<td>SOCKET1 TX Free Size Register</td>
</tr>
<tr>
<td>0x266</td>
<td>S1_TX_FSR2</td>
<td>S1_TX_FSR2  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_TX_FSR3</td>
<td>SOCKET1 TX Free Size Register</td>
</tr>
<tr>
<td>0x268</td>
<td>S1_RX_RSR</td>
<td>S1_RX_RSR0  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_RX_RSR1</td>
<td>SOCKET1 RX Receive Size Register</td>
</tr>
<tr>
<td>0x26A</td>
<td>S1_RX_RSR2</td>
<td>S1_RX_RSR2  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_RX_RSR3</td>
<td>SOCKET1 RX Receive Size Register</td>
</tr>
<tr>
<td>0x26C</td>
<td>S1_FRAGR</td>
<td>S1_FRAGR0  Reserved</td>
</tr>
<tr>
<td></td>
<td>S1_FRAGR1</td>
<td>SOCKET1 IP FLAG Field Register</td>
</tr>
<tr>
<td>0x26E</td>
<td>S1_TX_FIFOR</td>
<td>S1_TX_FIFOR0  SOCKET1 TX FIFO Register</td>
</tr>
<tr>
<td></td>
<td>S1_TX_FIFOR1</td>
<td>SOCKET1 TX FIFO Register</td>
</tr>
<tr>
<td>0x270</td>
<td>S1_RX_FIFOR</td>
<td>S1_RX_FIFOR0  SOCKET1 RX FIFO Register</td>
</tr>
<tr>
<td></td>
<td>S1_RX_FIFOR1</td>
<td>SOCKET1 RX FIFO Register</td>
</tr>
<tr>
<td>0x272</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x27E</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>0x280</td>
<td>0x280</td>
<td>S2_MR</td>
</tr>
<tr>
<td>0x281</td>
<td></td>
<td>S2_MR0</td>
</tr>
<tr>
<td>0x282</td>
<td>0x282</td>
<td>S2_CR</td>
</tr>
<tr>
<td>0x283</td>
<td></td>
<td>S2_CR0</td>
</tr>
<tr>
<td>0x284</td>
<td>0x284</td>
<td>S2_IMR</td>
</tr>
<tr>
<td>0x285</td>
<td></td>
<td>S2_IMR0</td>
</tr>
<tr>
<td>0x286</td>
<td>0x286</td>
<td>S2_IR</td>
</tr>
<tr>
<td>0x287</td>
<td></td>
<td>S2_IR0</td>
</tr>
<tr>
<td>0x288</td>
<td>0x288</td>
<td>S2_SSR</td>
</tr>
<tr>
<td>0x289</td>
<td></td>
<td>S2_SSR0</td>
</tr>
<tr>
<td>0x28A</td>
<td>0x28A</td>
<td>S2_PORTR</td>
</tr>
<tr>
<td>0x28B</td>
<td></td>
<td>S2_PORTR0</td>
</tr>
<tr>
<td>0x28C</td>
<td>0x28C</td>
<td>S2_DHAR</td>
</tr>
<tr>
<td>0x28D</td>
<td></td>
<td>S2_DHAR0</td>
</tr>
<tr>
<td>0x28E</td>
<td>0x28E</td>
<td>S2_DHAR2</td>
</tr>
<tr>
<td>0x28F</td>
<td></td>
<td>S2_DHAR3</td>
</tr>
<tr>
<td>0x290</td>
<td>0x290</td>
<td>S2_DHAR4</td>
</tr>
<tr>
<td>0x291</td>
<td></td>
<td>S2_DHAR5</td>
</tr>
<tr>
<td>0x292</td>
<td>0x292</td>
<td>S2_DPORTR</td>
</tr>
<tr>
<td>0x293</td>
<td></td>
<td>S2_DPORTR0</td>
</tr>
<tr>
<td>0x294</td>
<td>0x294</td>
<td>S2_DIPR</td>
</tr>
<tr>
<td>0x295</td>
<td></td>
<td>S2_DIPR0</td>
</tr>
<tr>
<td>0x296</td>
<td>0x296</td>
<td>S2_DIPR2</td>
</tr>
<tr>
<td>0x297</td>
<td></td>
<td>S2_DIPR3</td>
</tr>
<tr>
<td>0x298</td>
<td>0x298</td>
<td>S2_MSSR</td>
</tr>
<tr>
<td>0x299</td>
<td></td>
<td>S2_MSSR0</td>
</tr>
<tr>
<td>0x29A</td>
<td>0x29A</td>
<td>S2_PORTR</td>
</tr>
<tr>
<td>0x29B</td>
<td></td>
<td>S2_PORTR</td>
</tr>
<tr>
<td>0x29C</td>
<td>0x29C</td>
<td>S2_TOSR</td>
</tr>
<tr>
<td>0x29D</td>
<td></td>
<td>S2_TOSR0</td>
</tr>
<tr>
<td>0x29E</td>
<td>0x29E</td>
<td>S2_TTLR</td>
</tr>
<tr>
<td>0x29F</td>
<td></td>
<td>S2_TTLR0</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>0x2A0</td>
<td>S2_TX_WRSR0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2A1</td>
<td>S2_TX_WRSR1</td>
<td>SOCKET2 TX Write Size Register</td>
</tr>
<tr>
<td>0x2A2</td>
<td>S2_TX_WRSR2</td>
<td>SOCKET2 TX Free Size Register</td>
</tr>
<tr>
<td>0x2A3</td>
<td>S2_TX_WRSR3</td>
<td>SOCKET2 RX Receive Size Register</td>
</tr>
<tr>
<td>0x2A4</td>
<td>S2_TX_FSR0</td>
<td>SOCKET2 IP FLAG Field Register</td>
</tr>
<tr>
<td>0x2A5</td>
<td>S2_TX_FSR1</td>
<td>SOCKET2 TX FIFO Register</td>
</tr>
<tr>
<td>0x2A6</td>
<td>S2_TX_FSR2</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2A7</td>
<td>S2_TX_FSR3</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2A8</td>
<td>S2_RX_RSR0</td>
<td>SOCKET2 RX Receive Size Register</td>
</tr>
<tr>
<td>0x2A9</td>
<td>S2_RX_RSR1</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2AA</td>
<td>S2_RX_RSR2</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2AB</td>
<td>S2_RX_RSR3</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2AC</td>
<td>S2_FRAGR0</td>
<td>SOCKET2 TX FIFO Register</td>
</tr>
<tr>
<td>0x2AD</td>
<td>S2_FRAGR1</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2AE</td>
<td>S2_TX_FIFOR0</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2AF</td>
<td>S2_TX_FIFOR1</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2B0</td>
<td>S2_RX_FIFOR0</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2B1</td>
<td>S2_RX_FIFOR1</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2B2</td>
<td>S2_RX_FIFOR2</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2B3</td>
<td>S2_RX_FIFOR3</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2BE</td>
<td>S2_RX_FIFOR4</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>0x2BF</td>
<td>S2_RX_FIFOR5</td>
<td>SOCKET2 RX FIFO Register</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>0x2C0</td>
<td>0x2C0</td>
<td>S3_MR</td>
</tr>
<tr>
<td></td>
<td>0x2C1</td>
<td>S3_MR</td>
</tr>
<tr>
<td>0x2C2</td>
<td>0x2C2</td>
<td>S3_CR</td>
</tr>
<tr>
<td></td>
<td>0x2C3</td>
<td>S3_CR</td>
</tr>
<tr>
<td>0x2C4</td>
<td>0x2C4</td>
<td>S3_IMR</td>
</tr>
<tr>
<td></td>
<td>0x2C5</td>
<td>S3_IMR</td>
</tr>
<tr>
<td>0x2C6</td>
<td>0x2C6</td>
<td>S3_IR</td>
</tr>
<tr>
<td></td>
<td>0x2C7</td>
<td>S3_IR</td>
</tr>
<tr>
<td>0x2C8</td>
<td>0x2C8</td>
<td>S3_SSR</td>
</tr>
<tr>
<td></td>
<td>0x2C9</td>
<td>S3_SSR</td>
</tr>
<tr>
<td>0x2CA</td>
<td>0x2CA</td>
<td>S3_PORTR</td>
</tr>
<tr>
<td></td>
<td>0x2CB</td>
<td>S3_PORTR</td>
</tr>
<tr>
<td>0x2CC</td>
<td>0x2CC</td>
<td>S3_DHAR</td>
</tr>
<tr>
<td></td>
<td>0x2CD</td>
<td>S3_DHAR</td>
</tr>
<tr>
<td>0x2CE</td>
<td>0x2CE</td>
<td>S3_DHAR</td>
</tr>
<tr>
<td></td>
<td>0x2CF</td>
<td>S3_DHAR</td>
</tr>
<tr>
<td>0x2D0</td>
<td>0x2D0</td>
<td>S3_DHAR4</td>
</tr>
<tr>
<td></td>
<td>0x2D1</td>
<td>S3_DHAR4</td>
</tr>
<tr>
<td>0x2D2</td>
<td>0x2D2</td>
<td>S3_DPORTR</td>
</tr>
<tr>
<td></td>
<td>0x2D3</td>
<td>S3_DPORTR</td>
</tr>
<tr>
<td>0x2D4</td>
<td>0x2D4</td>
<td>S3_DIPR</td>
</tr>
<tr>
<td></td>
<td>0x2D5</td>
<td>S3_DIPR</td>
</tr>
<tr>
<td>0x2D6</td>
<td>0x2D6</td>
<td>S3_DIPR</td>
</tr>
<tr>
<td></td>
<td>0x2D7</td>
<td>S3_DIPR</td>
</tr>
<tr>
<td>0x2D8</td>
<td>0x2D8</td>
<td>S3_MSSR</td>
</tr>
<tr>
<td></td>
<td>0x2D9</td>
<td>S3_MSSR</td>
</tr>
<tr>
<td>0x2DA</td>
<td>0x2DA</td>
<td>S3_PORTR</td>
</tr>
<tr>
<td></td>
<td>0x2DB</td>
<td>S3_PORTR</td>
</tr>
<tr>
<td>0x2DC</td>
<td>0x2DC</td>
<td>S3_TOSR</td>
</tr>
<tr>
<td></td>
<td>0x2DD</td>
<td>S3_TOSR</td>
</tr>
<tr>
<td>0x2DE</td>
<td>0x2DE</td>
<td>S3_TTLR</td>
</tr>
<tr>
<td></td>
<td>0x2DF</td>
<td>S3_TTLR</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>0x2E0</td>
<td>S3_TX_WRSR0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2E1</td>
<td>S3_TX_WRSR1</td>
<td>SOCKET3 TX Write Size Register</td>
</tr>
<tr>
<td>0x2E2</td>
<td>S3_TX_WRSR2</td>
<td>SOCKET3 TX Write Size Register</td>
</tr>
<tr>
<td>0x2E3</td>
<td>S3_TX_WRSR3</td>
<td>SOCKET3 TX Write Size Register</td>
</tr>
<tr>
<td>0x2E4</td>
<td>S3_TX_FSR0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2E5</td>
<td>S3_TX_FSR1</td>
<td>SOCKET3 TX Free Size Register</td>
</tr>
<tr>
<td>0x2E6</td>
<td>S3_TX_FSR2</td>
<td>SOCKET3 TX Free Size Register</td>
</tr>
<tr>
<td>0x2E7</td>
<td>S3_TX_FSR3</td>
<td>SOCKET3 TX Free Size Register</td>
</tr>
<tr>
<td>0x2E8</td>
<td>S3_RX_RSR0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2E9</td>
<td>S3_RX_RSR1</td>
<td>SOCKET3 RX Receive Size Register</td>
</tr>
<tr>
<td>0x2EA</td>
<td>S3_RX_RSR2</td>
<td>SOCKET3 RX Receive Size Register</td>
</tr>
<tr>
<td>0x2EB</td>
<td>S3_RX_RSR3</td>
<td>SOCKET3 RX Receive Size Register</td>
</tr>
<tr>
<td>0x2EC</td>
<td>S3_FGRA0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2ED</td>
<td>S3_FGRA1</td>
<td>SOCKET3 IP FLAG Field Register</td>
</tr>
<tr>
<td>0x2EE</td>
<td>S3_TX_FIFOR0</td>
<td>SOCKET3 TX FIFO Register</td>
</tr>
<tr>
<td>0x2EF</td>
<td>S3_TX_FIFOR1</td>
<td>SOCKET3 TX FIFO Register</td>
</tr>
<tr>
<td>0x2F0</td>
<td>S3_RX_FIFOR0</td>
<td>SOCKET3 RX FIFO Register</td>
</tr>
<tr>
<td>0x2F1</td>
<td>S3_RX_FIFOR1</td>
<td>SOCKET3 RX FIFO Register</td>
</tr>
<tr>
<td>0x2F2</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2F3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2FE</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x300</td>
<td>S4_MR</td>
<td>SOCKET4 Mode Register</td>
</tr>
<tr>
<td>0x301</td>
<td></td>
<td>S4_MR1</td>
</tr>
<tr>
<td>0x302</td>
<td>S4_CR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x303</td>
<td>S4_CRI</td>
<td>SOCKET4 Command Register</td>
</tr>
<tr>
<td>0x304</td>
<td>S4_IMR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x305</td>
<td>S4_IMR1</td>
<td>SOCKET4 Interrupt Mask Register</td>
</tr>
<tr>
<td>0x306</td>
<td>S4_IR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x307</td>
<td>S4_IR1</td>
<td>SOCKET4 Interrupt Register</td>
</tr>
<tr>
<td>0x308</td>
<td>S4_SSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x309</td>
<td></td>
<td>SOCKET4 SOCKET Status Register</td>
</tr>
<tr>
<td>0x30A</td>
<td>S4_PORTR</td>
<td>SOCKET4 Source Port Register</td>
</tr>
<tr>
<td>0x30B</td>
<td></td>
<td>S4_PORTR1</td>
</tr>
<tr>
<td>0x30C</td>
<td>S4_DHAR</td>
<td>SOCKET4 Destination Hardware Address Register</td>
</tr>
<tr>
<td>0x30D</td>
<td>S4_DHAR1</td>
<td></td>
</tr>
<tr>
<td>0x30E</td>
<td>S4_DHAR2</td>
<td></td>
</tr>
<tr>
<td>0x30F</td>
<td>S4_DHAR3</td>
<td></td>
</tr>
<tr>
<td>0x310</td>
<td>S4_DHAR4</td>
<td></td>
</tr>
<tr>
<td>0x311</td>
<td>S4_DHAR5</td>
<td></td>
</tr>
<tr>
<td>0x312</td>
<td>S4_DPORTR</td>
<td>SOCKET4 Destination Port Register</td>
</tr>
<tr>
<td>0x313</td>
<td></td>
<td>S4_DPORTR1</td>
</tr>
<tr>
<td>0x314</td>
<td>S4_DIPR</td>
<td>SOCKET4 Destination IP Address Register</td>
</tr>
<tr>
<td>0x315</td>
<td>S4_DIPR1</td>
<td></td>
</tr>
<tr>
<td>0x316</td>
<td>S4_DIPR2</td>
<td></td>
</tr>
<tr>
<td>0x317</td>
<td>S4_DIPR3</td>
<td></td>
</tr>
<tr>
<td>0x318</td>
<td>S4_MSSR</td>
<td>SOCKET4 Maximum Segment Size Register</td>
</tr>
<tr>
<td>0x319</td>
<td>S4_MSSR1</td>
<td></td>
</tr>
<tr>
<td>0x31A</td>
<td>S4_PORTOR</td>
<td>SOCKET4 Keep Alive Time Register</td>
</tr>
<tr>
<td>0x31B</td>
<td>S4_KPALVTR</td>
<td></td>
</tr>
<tr>
<td>0x31C</td>
<td>S4_TOSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x31D</td>
<td>S4_TOSR1</td>
<td>SOCKET4 TOS Register</td>
</tr>
<tr>
<td>0x31E</td>
<td>S4_TTLR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x31F</td>
<td>S4_TTLR1</td>
<td>SOCKET4 TTL Register</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>0x320</td>
<td>S4_TX_WRSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x321</td>
<td>S4_TX_WRSR1</td>
<td>SOCKET4 TX Write Size Register</td>
</tr>
<tr>
<td>0x322</td>
<td>S4_TX_WRSR2</td>
<td>SOCKET4 TX Write Size Register</td>
</tr>
<tr>
<td>0x323</td>
<td>S4_TX_WRSR3</td>
<td>SOCKET4 TX Write Size Register</td>
</tr>
<tr>
<td>0x324</td>
<td>S4_TX_FSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x325</td>
<td>S4_TX_FSR1</td>
<td>SOCKET4 TX Free Size Register</td>
</tr>
<tr>
<td>0x326</td>
<td>S4_TX_FSR2</td>
<td>SOCKET4 TX Free Size Register</td>
</tr>
<tr>
<td>0x327</td>
<td>S4_TX_FSR3</td>
<td>SOCKET4 TX Free Size Register</td>
</tr>
<tr>
<td>0x328</td>
<td>S4_RX_RSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x329</td>
<td>S4_RX_RSR1</td>
<td>SOCKET4 RX Receive Size Register</td>
</tr>
<tr>
<td>0x32A</td>
<td>S4_RX_RSR2</td>
<td>SOCKET4 RX Receive Size Register</td>
</tr>
<tr>
<td>0x32B</td>
<td>S4_RX_RSR3</td>
<td>SOCKET4 RX Receive Size Register</td>
</tr>
<tr>
<td>0x32C</td>
<td>S4_FRAGR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x32D</td>
<td>S4_FRAGR1</td>
<td>SOCKET4 IP FLAG Field Register</td>
</tr>
<tr>
<td>0x32E</td>
<td>S4_TX_FIFOR</td>
<td>SOCKET4 TX FIFO Register</td>
</tr>
<tr>
<td>0x32F</td>
<td>S4_TX_FIFOR1</td>
<td>SOCKET4 TX FIFO Register</td>
</tr>
<tr>
<td>0x330</td>
<td>S4_RX_FIFOR</td>
<td>SOCKET4 RX FIFO Register</td>
</tr>
<tr>
<td>0x331</td>
<td>S4_RX_FIFOR1</td>
<td>SOCKET4 RX FIFO Register</td>
</tr>
<tr>
<td>0x332</td>
<td>0x332</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x333</td>
<td>0x333</td>
<td></td>
</tr>
<tr>
<td>0x33E</td>
<td>0x33E</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x33F</td>
<td>0x33F</td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x340</td>
<td>S5_MR</td>
<td>SOCKET5 Mode Register</td>
</tr>
<tr>
<td>0x341</td>
<td>S5_MR1</td>
<td></td>
</tr>
<tr>
<td>0x342</td>
<td>S5_CR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x343</td>
<td>S5_CR1</td>
<td>SOCKET5 Command Register</td>
</tr>
<tr>
<td>0x344</td>
<td>S5_IMR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x345</td>
<td>S5_IMR1</td>
<td>SOCKET5 Interrupt Mask Register</td>
</tr>
<tr>
<td>0x346</td>
<td>S5_IR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x347</td>
<td>S5_IR1</td>
<td>SOCKET5 Interrupt Register</td>
</tr>
<tr>
<td>0x348</td>
<td>S5_SSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x349</td>
<td>S5_SSR1</td>
<td>SOCKET5 SOCKET Status Register</td>
</tr>
<tr>
<td>0x34A</td>
<td>S5_PORTR</td>
<td>SOCKET5 Source Port Register</td>
</tr>
<tr>
<td>0x34B</td>
<td>S5_PORTR1</td>
<td></td>
</tr>
<tr>
<td>0x34C</td>
<td>S5_DHAR</td>
<td>SOCKET5 Destination Hardware Address Register</td>
</tr>
<tr>
<td>0x34D</td>
<td>S5_DHAR1</td>
<td></td>
</tr>
<tr>
<td>0x34E</td>
<td>S5_DHAR2</td>
<td></td>
</tr>
<tr>
<td>0x34F</td>
<td>S5_DHAR3</td>
<td></td>
</tr>
<tr>
<td>0x350</td>
<td>S5_DHAR4</td>
<td></td>
</tr>
<tr>
<td>0x351</td>
<td>S5_DHAR5</td>
<td></td>
</tr>
<tr>
<td>0x352</td>
<td>S5_DPORTR</td>
<td>SOCKET5 Destination Port Register</td>
</tr>
<tr>
<td>0x353</td>
<td>S5_DPORTR1</td>
<td></td>
</tr>
<tr>
<td>0x354</td>
<td>S5_DIPR</td>
<td>SOCKET5 Destination IP Address Register</td>
</tr>
<tr>
<td>0x355</td>
<td>S5_DIPR1</td>
<td></td>
</tr>
<tr>
<td>0x356</td>
<td>S5_DIPR2</td>
<td></td>
</tr>
<tr>
<td>0x357</td>
<td>S5_DIPR3</td>
<td></td>
</tr>
<tr>
<td>0x358</td>
<td>S5_MSSR</td>
<td>SOCKET5 Maximum Segment Size Register</td>
</tr>
<tr>
<td>0x359</td>
<td>S5_MSSR1</td>
<td></td>
</tr>
<tr>
<td>0x35A</td>
<td>S5_PORTOR</td>
<td>SOCKET5 Keep Alive Time Register</td>
</tr>
<tr>
<td>0x35B</td>
<td>S5_PROTOR</td>
<td>SOCKET5 Protocol Number Register</td>
</tr>
<tr>
<td>0x35C</td>
<td>S5_TOSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x35D</td>
<td>S5_TOSR1</td>
<td>SOCKET5 TOS Register</td>
</tr>
<tr>
<td>0x35E</td>
<td>S5_TTLR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x35F</td>
<td>S5_TTLR1</td>
<td>SOCKET5 TTL Register</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>0x360</td>
<td>S5_TX_WRSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x361</td>
<td>S5_TX_WRSR0</td>
<td>SOCKET5 TX Write Size Register</td>
</tr>
<tr>
<td>0x362</td>
<td>S5_TX_WRSR2</td>
<td>SOCKET5 TX Receive Size Register</td>
</tr>
<tr>
<td>0x363</td>
<td>S5_TX_WRSR3</td>
<td>SOCKET5 RX Receive Size Register</td>
</tr>
<tr>
<td>0x364</td>
<td>S5_TX_FSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x365</td>
<td>S5_TX_FSR0</td>
<td>SOCKET5 TX Write Size Register</td>
</tr>
<tr>
<td>0x366</td>
<td>S5_TX_FSR2</td>
<td>SOCKET5 TX Receive Size Register</td>
</tr>
<tr>
<td>0x367</td>
<td>S5_TX_FSR3</td>
<td>SOCKET5 RX Receive Size Register</td>
</tr>
<tr>
<td>0x368</td>
<td>S5_RX_RSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x369</td>
<td>S5_RX_RSR0</td>
<td>SOCKET5 RX Receive Size Register</td>
</tr>
<tr>
<td>0x36A</td>
<td>S5_RX_RSR1</td>
<td>SOCKET5 RX Receive Size Register</td>
</tr>
<tr>
<td>0x36B</td>
<td>S5_RX_RSR2</td>
<td>SOCKET5 RX Receive Size Register</td>
</tr>
<tr>
<td>0x36C</td>
<td>S5_FGRA0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x36D</td>
<td>S5_FGRA1</td>
<td>SOCKET5 TX FIFO Register</td>
</tr>
<tr>
<td>0x36E</td>
<td>S5_TX_FIFOR</td>
<td>SOCKET5 TX FIFO Register</td>
</tr>
<tr>
<td>0x36F</td>
<td>S5_TX_FIFOR0</td>
<td>SOCKET5 TX FIFO Register</td>
</tr>
<tr>
<td>0x370</td>
<td>S5_RX_FIFOR</td>
<td>SOCKET5 RX FIFO Register</td>
</tr>
<tr>
<td>0x371</td>
<td>S5_RX_FIFOR0</td>
<td>SOCKET5 RX FIFO Register</td>
</tr>
<tr>
<td>0x372</td>
<td>S5_RX_FIFOR1</td>
<td>SOCKET5 RX FIFO Register</td>
</tr>
<tr>
<td>0x373</td>
<td>S5_RX_FIFOR</td>
<td>SOCKET5 RX FIFO Register</td>
</tr>
<tr>
<td>0x37E</td>
<td>S5_TX_FIFOR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x37F</td>
<td>S5_RX_FIFOR</td>
<td>Reserved</td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>0x380</td>
<td>S6_MR</td>
<td>SOCKET6 Mode Register</td>
</tr>
<tr>
<td></td>
<td>S6_MR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_MR1</td>
<td></td>
</tr>
<tr>
<td>0x382</td>
<td>S6.CR</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>S6.CR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6.CR1</td>
<td>SOCKET6 Command Register</td>
</tr>
<tr>
<td>0x384</td>
<td>S6.IMR</td>
<td>SOCKET6 Interrupt Mask Register</td>
</tr>
<tr>
<td></td>
<td>S6.IMR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6.IMR1</td>
<td></td>
</tr>
<tr>
<td>0x386</td>
<td>S6.CR</td>
<td>SOCKET6 Command Register</td>
</tr>
<tr>
<td></td>
<td>S6.CR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6.CR1</td>
<td></td>
</tr>
<tr>
<td>0x388</td>
<td>S6.IMR</td>
<td>SOCKET6 Interrupt Mask Register</td>
</tr>
<tr>
<td></td>
<td>S6.IMR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6.IMR1</td>
<td></td>
</tr>
<tr>
<td>0x38A</td>
<td>S6_PORT</td>
<td>SOCKET6 Source Port Register</td>
</tr>
<tr>
<td></td>
<td>S6_PORT0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_PORT1</td>
<td></td>
</tr>
<tr>
<td>0x38C</td>
<td>S6_DHAR</td>
<td>SOCKET6 Destination Hardware</td>
</tr>
<tr>
<td></td>
<td>S6_DHAR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_DHAR1</td>
<td></td>
</tr>
<tr>
<td>0x38E</td>
<td>S6_DHAR</td>
<td>SOCKET6 Destination Hardware</td>
</tr>
<tr>
<td></td>
<td>S6_DHAR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_DHAR3</td>
<td></td>
</tr>
<tr>
<td>0x390</td>
<td>S6_DHAR</td>
<td>SOCKET6 Destination Hardware</td>
</tr>
<tr>
<td></td>
<td>S6_DHAR4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_DHAR5</td>
<td></td>
</tr>
<tr>
<td>0x392</td>
<td>S6_PORT</td>
<td>SOCKET6 Destination Port Register</td>
</tr>
<tr>
<td></td>
<td>S6_PORT0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_PORT1</td>
<td></td>
</tr>
<tr>
<td>0x394</td>
<td>S6_DIP</td>
<td>SOCKET6 Destination IP Address</td>
</tr>
<tr>
<td></td>
<td>S6_DIPR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_DIPR1</td>
<td></td>
</tr>
<tr>
<td>0x396</td>
<td>S6_DIP</td>
<td>SOCKET6 Destination IP Address</td>
</tr>
<tr>
<td></td>
<td>S6_DIPR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_DIPR3</td>
<td></td>
</tr>
<tr>
<td>0x398</td>
<td>S6_MSSR</td>
<td>SOCKET6 Maximum Segment Size</td>
</tr>
<tr>
<td></td>
<td>S6_MSSR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_MSSR1</td>
<td></td>
</tr>
<tr>
<td>0x39A</td>
<td>S6_PORTOR</td>
<td>SOCKET6 Keep Alive Time Register</td>
</tr>
<tr>
<td></td>
<td>S6_KPALVTR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_PROTOR</td>
<td></td>
</tr>
<tr>
<td>0x39C</td>
<td>S6_TOSR</td>
<td>SOCKET6 TOS Register</td>
</tr>
<tr>
<td></td>
<td>S6_TOSR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_TOSR1</td>
<td></td>
</tr>
<tr>
<td>0x39E</td>
<td>S6_TTLR</td>
<td>SOCKET6 TTL Register</td>
</tr>
<tr>
<td></td>
<td>S6_TTLR0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S6_TTLR1</td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>0x3A0</td>
<td>S6_TX_WRSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x3A1</td>
<td></td>
<td>S6_TX_WRSR1</td>
</tr>
<tr>
<td>0x3A2</td>
<td>S6_TX_WRSR2</td>
<td>SOCKET6 TX Write Size Register</td>
</tr>
<tr>
<td>0x3A3</td>
<td>S6_TX_WRSR3</td>
<td>SOCKET6 TX Write Size Register</td>
</tr>
<tr>
<td>0x3A4</td>
<td>S6_TX_FSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x3A5</td>
<td></td>
<td>S6_TX_FSR1</td>
</tr>
<tr>
<td>0x3A6</td>
<td>S6_TX_FSR2</td>
<td>SOCKET6 TX Free Size Register</td>
</tr>
<tr>
<td>0x3A7</td>
<td>S6_TX_FSR3</td>
<td>SOCKET6 TX Free Size Register</td>
</tr>
<tr>
<td>0x3A8</td>
<td>S6_RX_RSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x3A9</td>
<td></td>
<td>S6_RX_RSR1</td>
</tr>
<tr>
<td>0x3AA</td>
<td>S6_RX_RSR2</td>
<td>SOCKET6 RX Receive Size Register</td>
</tr>
<tr>
<td>0x3AB</td>
<td>S6_RX_RSR3</td>
<td>SOCKET6 RX Receive Size Register</td>
</tr>
<tr>
<td>0x3AC</td>
<td>S6_FRAGR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x3AD</td>
<td></td>
<td>S6_FRAGR1</td>
</tr>
<tr>
<td>0x3AE</td>
<td>S6_TX_FIFOR</td>
<td>SOCKET6 TX FIFO Register</td>
</tr>
<tr>
<td>0x3AF</td>
<td>S6_TX_FIFOR</td>
<td>SOCKET6 TX FIFO Register</td>
</tr>
<tr>
<td>0x3B0</td>
<td>S6_RX_FIFOR</td>
<td>SOCKET6 RX FIFO Register</td>
</tr>
<tr>
<td>0x3B1</td>
<td>S6_RX_FIFOR</td>
<td>SOCKET6 RX FIFO Register</td>
</tr>
<tr>
<td>0x3B2</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x3B3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3BE</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x3BF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>0x3C0</td>
<td>S7_MR</td>
<td>SOCKET7 Mode Register</td>
</tr>
<tr>
<td>0x3C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C2</td>
<td>S7_CR</td>
<td>SOCKET7 Command Register</td>
</tr>
<tr>
<td>0x3C3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C4</td>
<td>S7_IMR</td>
<td>SOCKET7 Interrupt Mask Register</td>
</tr>
<tr>
<td>0x3C5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C6</td>
<td>S7_IR</td>
<td>SOCKET7 Interrupt Register</td>
</tr>
<tr>
<td>0x3C7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C8</td>
<td>S7_SSR</td>
<td>SOCKET7 SOCKET Status Register</td>
</tr>
<tr>
<td>0x3C9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3CA</td>
<td>S7_PORTR</td>
<td>SOCKET7 Source Port Register</td>
</tr>
<tr>
<td>0x3CB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3CC</td>
<td>S7_DHAR</td>
<td>SOCKET7 Destination Hardware Address Register</td>
</tr>
<tr>
<td>0x3CD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3CE</td>
<td>S7_DHAR2</td>
<td></td>
</tr>
<tr>
<td>0x3CF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3D0</td>
<td>S7_DHAR4</td>
<td></td>
</tr>
<tr>
<td>0x3D1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3D2</td>
<td>S7_DPORTR</td>
<td>SOCKET7 Destination Port Register</td>
</tr>
<tr>
<td>0x3D3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3D4</td>
<td>S7_DIPR</td>
<td>SOCKET7 Destination IP Address Register</td>
</tr>
<tr>
<td>0x3D5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3D6</td>
<td>S7_DIPR2</td>
<td></td>
</tr>
<tr>
<td>0x3D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3D8</td>
<td>S7_MSSR</td>
<td>SOCKET7 Maximum Segment Size Register</td>
</tr>
<tr>
<td>0x3D9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3DA</td>
<td>S7_PORTOR</td>
<td>SOCKET7 Keep Alive Time Register</td>
</tr>
<tr>
<td>0x3DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3DC</td>
<td>S7_TOSR</td>
<td>SOCKET7 TOS Register</td>
</tr>
<tr>
<td>0x3DD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3DE</td>
<td>S7_TTLR</td>
<td>SOCKET7 TTL Register</td>
</tr>
<tr>
<td>0x3DF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address offset</td>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>0x3E0</td>
<td>0x3E0</td>
<td>S7_TX_WRSR0</td>
</tr>
<tr>
<td></td>
<td>0x3E1</td>
<td>S7_TX_WRSR1</td>
</tr>
<tr>
<td></td>
<td>0x3E2</td>
<td>S7_TX_WRSR2</td>
</tr>
<tr>
<td></td>
<td>0x3E3</td>
<td>S7_TX_WRSR3</td>
</tr>
<tr>
<td>0x3E4</td>
<td>0x3E4</td>
<td>S7_TX_FSR0</td>
</tr>
<tr>
<td></td>
<td>0x3E5</td>
<td>S7_TX_FSR1</td>
</tr>
<tr>
<td>0x3E6</td>
<td>0x3E6</td>
<td>S7_TX_FSR2</td>
</tr>
<tr>
<td></td>
<td>0x3E7</td>
<td>S7_TX_FSR3</td>
</tr>
<tr>
<td>0x3E8</td>
<td>0x3E8</td>
<td>S7_RX_RSR0</td>
</tr>
<tr>
<td></td>
<td>0x3E9</td>
<td>S7_RX_RSR1</td>
</tr>
<tr>
<td>0x3EA</td>
<td>0x3EA</td>
<td>S7_RX_RSR2</td>
</tr>
<tr>
<td></td>
<td>0x3EB</td>
<td>S7_RX_RSR3</td>
</tr>
<tr>
<td>0x3EC</td>
<td>0x3EC</td>
<td>S7_FRAGR0</td>
</tr>
<tr>
<td></td>
<td>0x3ED</td>
<td>S7_FRAGR1</td>
</tr>
<tr>
<td>0x3EE</td>
<td>0x3EE</td>
<td>S7_TX_FIFOR0</td>
</tr>
<tr>
<td></td>
<td>0x3EF</td>
<td>S7_TX_FIFOR1</td>
</tr>
<tr>
<td>0x3F0</td>
<td>0x3F0</td>
<td>S7_RX_FIFOR0</td>
</tr>
<tr>
<td></td>
<td>0x3F1</td>
<td>S7_RX_FIFOR1</td>
</tr>
<tr>
<td>0x3F2</td>
<td>0x3F2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x3F3</td>
<td></td>
</tr>
<tr>
<td>0x3FE</td>
<td>0x3FE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x3FF</td>
<td></td>
</tr>
</tbody>
</table>

- SOCKET7 TX Write Size Register
- SOCKET7 TX Free Size Register
- SOCKET7 RX Receive Size Register
- SOCKET7 IP FLAG Field Register
- SOCKET7 TX FIFO Register
- SOCKET7 RX FIFO Register
- Reserved
4. Register Description

[Notation]

1. Symbol(Name)[R/W,RO,WO][AO1/AO2][Reset]
   - Symbol : Register Symbol
   - Name : Register Name
   - R/W : Read/Write
   - RO : Read Only
   - WO : Write Only
   - AO1 : Physical Address of W5300 reg. in T.M.S (For Direct address mode)
   - AO2 : Address Offset of W5300 reg. in W.M.S (For Indirect address mode)
   - Reset : Reset value

For convenience, we assume the Base Address(BA) of T.M.S is 0x08000, and BA of the Physical Address of W5300 Register is 0x08000.

2. Pn_ : Buffer Ready PIN n(“BRDYn”) register prefix
   - Pn_BRDYR(BRDYn Configure register, 0 <= n <= 3)

3. Sn_ : SOCKETn register prefix
   - Sn_MR ( SOCKETn mode register, 0 <= n <= 7)

4. Symbol of low address Reg. Bit 15 14 13 12 11 10 9 8
   Physical Address Symbol - - - - - - - -
   Address offset Reset Value 1 0 0 0 X U(R) 0 0
   Symbol of high address Bit 7 6 5 4 3 2 1 0
   Reg.
   Physical Address Symbol - - - - - - - -
   Address offset Reset Value 0 0 0 0 0 0 0 0
   - : Reserved Bit 1 : Logical High 0 : Logical Low
   X : Don’t Care U : 1 or 0 (R) : Read Only Bit

<table>
<thead>
<tr>
<th>16 bit Register Symbol(AO1/AO2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8bit Register Symbol (AO1/AO2)</td>
</tr>
<tr>
<td>8bit Register Symbol (AO1/AO2)</td>
</tr>
<tr>
<td>MSB(Value)</td>
</tr>
<tr>
<td>LSB(Value)</td>
</tr>
</tbody>
</table>
4.1 Mode Register

**MR (Mode Register) [R/W] [0x08000/----][0x3800 or 0xB800]**

MR sets the mode of W5300 such like that host Interface mode, MSB/LSB swap of Sn_TX_FIOR & Sn_RX_FIFOR, S/W reset, internal TX/RX memory test, MSB/LSB swap of data bus and address mode.

<table>
<thead>
<tr>
<th>MR0</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>DBW</td>
<td>MPF</td>
<td>WDF2</td>
<td>WDF1</td>
<td>WDF0</td>
<td>RDH</td>
<td>-</td>
<td>FS</td>
</tr>
<tr>
<td>----</td>
<td>U(R)</td>
<td>0(R)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MR1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0801</td>
<td>RST</td>
<td>-</td>
<td>MT</td>
<td>PB</td>
<td>PPPoE</td>
<td>DBS</td>
<td>-</td>
<td>IND</td>
</tr>
<tr>
<td>----</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**MR(15:8)/MR0(7:0)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15  | DBW    | **Data Bus Width**
|     |        | 0 : 8 bit data bus
|     |        | 1 : 16 bit data bus
|     |        | At the reset time of W5300, it is fixed according to logic level of PIN “BIT16EN”. After reset, it is not changed.
|     |        | Refer to BIT16EN description of “1.1 PIN Layout” |
| 14  | MPF    | **MAC Layer Pause Frame**
|     |        | 0 : Normal frame
|     |        | 1 : Pause frame
|     |        | It is set as ‘1’, when pause frame is received from router or switch. When set as ‘1’, all data transmit is paused until changing to ‘0’.
| 13  | WDF2   | **Write Data Fetch Time**
| 12  | WDF1   | When Host-Write operation, since ’/CS’ is asserted low, W5300fetches Write-Data after WRF X PLL_CLK.
| 11  | WDF0   | If Host-Write operation is finished (’/CS’ is de-asserted high) before WRF X PLL_CLK, Write-Data is fetched at the time that ’/CS’ is de-asserted high.
| 10  | RDH    | **Read Data Hold Time**
## FIFO Swap Bit

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable swap</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Enable swap</td>
<td></td>
</tr>
</tbody>
</table>

It swaps the most significant byte (MSB) and least significant byte (LSB). Basically, the byte ordering of W5300 is big-endian. If the target host system is based on little-endian, you can switch the byte ordering of Sn_TX_FIFO/Sn_RX_FIFO by setting this bit as ‘1’, and use it as like little-endian.

## MR(7:0)/MR1(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>RST</td>
<td>S/W Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If it’s set as ‘1’, W5300 is reset. This bit is automatically cleared after reset.</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>MT</td>
<td>Memory Test Bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 : Disable internal TX/RX memory test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : Enable internal TX/RX memory test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Basically, internal TX memory of W5300 supports Host-Write operation through Sn_TX_FIFO, and internal RX memory does Host-Read operation through Sn_RX_FIFO. However if this bit is set as ‘1’, internal TX/RX memory supports both of Host-Read and Host-Write operation through Sn_TX_FIFO/Sn_RX_FIFO, and verifies the internal TX/RX memory. After testing W5300 internal TX/RX memory, be sure to reset or close the SOCKET. For the detail, refer to “How to test internal TX/RX memory”.</td>
</tr>
<tr>
<td>4</td>
<td>PB</td>
<td>Ping Block Mode</td>
</tr>
</tbody>
</table>
|   | 0 : Disable Ping block  
|   | 1 : Enable Ping block  
|   |  
|   | When this bit is set as ‘1’, Auto-ping-reply-process of ICMP logic block is disabled, and Ping-reply(ICMP echo reply) is not processed to the Ping-request(ICMP echo request).  
|   | cf> Even though ping block mode is ‘0’, when a user uses ICMP SOCKET (Sn_MR(P3:P0)=Sn_MR_IPRAW and Sn_PROTOR=0x01), Auto-ping-reply is not processed. Auto-ping-reply supports max.119Bytes.  
| 3 | PPPoE | PPPoE Mode |  
|   | 0 : Disable PPPoE mode  
|   | 1 : Enable PPPoE mode  
|   |  
|   | This bit should be set as ‘1’, when connecting to PPPOE server without router or others. For the detail, refer to “How to use PPPoE in W5300”  
| 2 | DBS | Data Bus Swap |  
|   | 0 : Disable swap  
|   | 1 : Enable swap  
|   |  
|   | FS bit only swaps MSB and LSB of Sn_TX_FIFOR/Sn_RX_FIFOR. However, this bit swaps MSB and LSB of all registers including Sn_TX_FIFOR/Sn_RX_FIFOR. This bit is valid when DBW bit is ‘1’  
| 1 | - | Reserved |  
| 0 | IND | Indirect Bus I/F mode |  
|   | 0 : Direct address Mode  
|   | 1 : Indirect address Mode  
|   |  
|   | It sets host interface mode of W5300. |
4.2 Indirect Mode Registers

In case of MR(IND) = ‘1’, W5300 operates as indirect address mode. Target host system can access indirectly COMMON and SOCKET registers using only MR, IDM_AR, IDM_DR(That is, Target host system can access directly MR, IDM_AR, IDM_DR which are only mapped in T.M.S).

**IDM_AR(Indirect Mode Address Register) [R/W] [0x08002/----][0x0000]**

It sets an address offset of COMMON registers or SOCKET registers that are indirectly accessible. IDM_AR(0) or IDM_AR1(0) which is the least significant bit of IDM_AR, are ignored.

Ex) Accessing S4_RX_FIFOR(0x330) is as below.

<table>
<thead>
<tr>
<th>IDM_AR0 = MSB (0x03) of address offset of S4_RX_FIFOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDM_AR1 = LSB (0x30) of address offset of S4_RX_FIFOR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDM_AR(0x08002/----)</th>
<th>IDM_AR0(0x08002/----)</th>
<th>IDM_AR1(0x08003/----)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03</td>
<td>0x30</td>
<td></td>
</tr>
</tbody>
</table>

**IDM_DR(Indirect Mode Data Register) [R/W] [0x08004/----][0x0000]**

It accesses a real data value of COMMON or SOCKET registers that are indirectly accessible. IDM_DR0 corresponds to MSB values of the register addressed by IDM_AR, and IDM_DR1 does to LSB value of that.

When using 8bit data bus width and accessing LSB of any register, IDM_DR1 should be accessed. When accessing MSB, IDM_DR0 should be accessed.

It accesses the real value of COMMON or SOCKET registers which have the address offset in IDM_AR.

The MSB and LSB value of register addressed by IDM_AR corresponds to DM_DR0 and IDM_DR1 respectively.

At 8 bit data bus width, if the host access the LSB value of register addressed by IDM_AR then use IDM_DR1, and if the host access the MSB value of that then use IDM_DR0.

Ex1) When the host writes IR(0x002) with the value 0x80F0,

<table>
<thead>
<tr>
<th>16 Bit Data Bus Width ( MR(DBW) = ‘1’)</th>
<th>8 Bit Data Bus Width ( MR(DBW) = ‘0’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDM_AR = 0x0002</td>
<td>IDM_AR0 = 0x00</td>
</tr>
<tr>
<td>IDM_DR = 0x80F0</td>
<td>IDM_AR1 = 0x02</td>
</tr>
<tr>
<td></td>
<td>IDM_DR0 = 0x80</td>
</tr>
<tr>
<td></td>
<td>IDM_DR1 = 0xF0</td>
</tr>
</tbody>
</table>
4.3 COMMON Registers

IR (Interrupt Register) [R/W] [0x08002/0x002] [0x0000]

IR is the register to notify W5300 interrupt type to the host. When any interrupt occurs, the related bit of IR is set as ‘1’, and if the related interrupt mask bit is ‘1’ then ‘/INT’ signal is asserted low.

‘/INT’ signal keeps low until all bits of IR becomes ‘0’. If all bits of IR become ‘0’, it is de-asserted high. In order to clear IR0’s bit which was set as ‘1’, the host should write the bit as ‘1’. In case of IR1’s bit which was set as ‘1’, it is automatically cleared when clearing all bits of the related Sn_IR.
### IR(15:8)/IR0(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>IPCF</td>
<td>IP Conflict</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It’s set as ‘1’ when IP address is conflicted (when receiving ARP-request packet having same IP address as source IP address of W5300). When it’s set as ‘1’, there is another device using same IP address on the network to cause communication problem. Therefore, quick step is required to settle this problem.</td>
</tr>
<tr>
<td>14</td>
<td>DPUR</td>
<td>Destination Port unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It’s set as ‘1’ when receiving ICMP(Destination port unreachable) packet. Refer to UIPR and UPORTR.</td>
</tr>
<tr>
<td>13</td>
<td>PPPT</td>
<td>PPPoE Terminate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the connection with server is closed at the PPPoE mode, it is set as ‘1’.</td>
</tr>
<tr>
<td>12</td>
<td>FMTU</td>
<td>Fragment MTU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When receiving ICMP (Fragment MTU) packet, it’s set as ‘1’ Refer to FMTUR.</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### IR(7:0)/IR1(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>S7_INT</td>
<td>Occurrence of SOCKET7 Interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET7, it becomes ‘1’. This interrupt information is applied to S7_IR1. This bit is automatically cleared when S7_IR1 is cleared to 0x00 by host.</td>
</tr>
<tr>
<td>6</td>
<td>S6_INT</td>
<td>Occurrence of SOCKET6 Interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET6, it becomes ‘1’. This interrupt</td>
</tr>
<tr>
<td>5</td>
<td>S5_INT</td>
<td>Occurrence of SOCKET5 Interrupt</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET5, it becomes ‘1’. This interrupt information is applied to S5_IR1. This bit is automatically cleared when S5_IR1 is cleared to 0x00 by host.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>S4_INT</th>
<th>Occurrence of SOCKET4 Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET4, it becomes ‘1’. This interrupt information is applied to S4_IR1. This bit is automatically cleared when S4_IR1 is cleared to 0x00 by host.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>S3_INT</th>
<th>Occurrence of SOCKET3 Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET3, it becomes ‘1’. This interrupt information is applied to S3_IR1. This bit is automatically cleared when S3_IR1 is cleared to 0x00 by host.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>S2_INT</th>
<th>Occurrence of SOCKET2 Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET2, it becomes ‘1’. This interrupt information is applied to S2_IR1. This bit is automatically cleared when S2_IR1 is cleared to 0x00 by host.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>S1_INT</th>
<th>Occurrence of SOCKET1 Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET1, it becomes ‘1’. This interrupt information is applied to S1_IR1. This Bit is automatically cleared when S1_IR1 is cleared to 0x00 by host.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>S0_INT</th>
<th>Occurrence of SOCKET0 Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When an interrupt occurs at the SOCKET0, it becomes ‘1’. This interrupt information is applied to S1_IR1. This bit is automatically cleared when S1_IR1 is cleared to 0x00 by host.</td>
</tr>
</tbody>
</table>

**IMR (Interrupt Mask Register) [R/W] [0x08004/0x004] [0x0000]**

It configures W5300’s interrupt to notify the host. Each interrupt mask bit of IMR corresponds to
each interrupt bit of IR. When any bit of IR is set as ‘1’ and its corresponding bit of IMR is also set as ‘1’, interrupt is issued to the host. (‘/INT’ pin is asserted from high to low). If corresponding IMR bit is not set as ‘0’, the interrupt is not issued to the host (‘INT’ pin keeps high) even though IR bit is set as ‘1’.

<table>
<thead>
<tr>
<th>IMR0</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>IPCF</td>
<td>DPUR</td>
<td>PPPT</td>
<td>FMTU</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0x005</td>
<td>S7_INT</td>
<td>S6_INT</td>
<td>S5_INT</td>
<td>S4_INT</td>
<td>S3_INT</td>
<td>S2_INT</td>
<td>S1_INT</td>
<td>S0_INT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMR(15:8)/IMR0(7:0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMR(7:0)/IMR1(7:0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
SHAR (Source Hardware Address Register) [R/W] [0x08008/0x008] [00.00.00.00.00.00]

It configures source hardware address (MAC address).

Ex) In case of “00.08.DC.01.02.03”

<table>
<thead>
<tr>
<th>SHAR(0x08008/0x008)</th>
<th>SHAR0(0x08008/0x008)</th>
<th>SHAR1(0x08009/0x009)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x00</td>
<td>0x08</td>
</tr>
<tr>
<td>SHAR2(0x0800A/0x00A)</td>
<td>SHAR2(0x0800A/0x00A)</td>
<td>SHAR3(0x0800B/0x00B)</td>
</tr>
<tr>
<td></td>
<td>0xDC</td>
<td>0x01</td>
</tr>
<tr>
<td>SHAR4(0x0800C/0x00C)</td>
<td>SHAR4(0x0800C/0x00C)</td>
<td>SHAR5(0x0800D/0x00D)</td>
</tr>
<tr>
<td></td>
<td>0x02</td>
<td>0x03</td>
</tr>
</tbody>
</table>

GAR (Gateway IP Address Register) [R/W] [0x08010/0x010] [00.00.00.00]

It configures gateway IP address.

Ex) in case of “192.168.0.1”

<table>
<thead>
<tr>
<th>GAR(0x08010/0x010)</th>
<th>GAR2(0x08012/0x012)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAR(0x08010/0x010)</td>
<td>GAR1(0x08011/0x011)</td>
</tr>
<tr>
<td></td>
<td>GAR2(0x08012/0x012)</td>
</tr>
<tr>
<td></td>
<td>GAR3(0x08013/0x013)</td>
</tr>
<tr>
<td>192(0xC0)</td>
<td>168(0xA8)</td>
</tr>
<tr>
<td>0(0x00)</td>
<td>1(0x01)</td>
</tr>
</tbody>
</table>

SUBR (Subnet Mask Register) [R/W] [0x08014/0x014] [00.00.00.00]

It configures subnet mask address.

Ex) in case of “255.255.255.0”

<table>
<thead>
<tr>
<th>SUBR(0x08014/0x014)</th>
<th>SUBR2(0x08016/0x016)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBR0(0x08014/0x01)</td>
<td>SUBR1(0x08015/0x01)</td>
</tr>
<tr>
<td></td>
<td>SUBR2(0x08016/0x01)</td>
</tr>
<tr>
<td></td>
<td>SUBR3(0x08017/0x01)</td>
</tr>
<tr>
<td>255 (0xFF)</td>
<td>255 (0xFF)</td>
</tr>
<tr>
<td>255 (0xFF)</td>
<td>0 (0x00)</td>
</tr>
</tbody>
</table>

SIPR (Source IP Address Register) [R/W] [0x08018/0x018] [00.00.00.00]

It configures source IP address or notifies source IP address acquired by PPPoE-process of W5300.

Ex) in case of “192.168.0.3”

<table>
<thead>
<tr>
<th>SIPR(0x08018/0x018)</th>
<th>SIPR2(0x0801A/0x01A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPR0(0x08018/0x018)</td>
<td>SIPR1(0x08019/0x019)</td>
</tr>
<tr>
<td>SIPR2(0x0801A/0x01A)</td>
<td>SIPR3(0x0801B/0x01B)</td>
</tr>
</tbody>
</table>
High-performance Internet Connectivity Solution  W5300

RTR (Retransmission Timeout-period Register) [R/W] [0x0801C/0x01C] [0x07D0]
It configure retransmission timeout-period. The standard unit of RTR is 100us. RTR is initialized with 2000(0x07D0) and has 200ms timeout-period.

Ex) When timeout-period is set as 400ms, RTR = (400ms / 1ms) X 10 = 4000(0x0FA0)

<table>
<thead>
<tr>
<th>RTR(0x0801C/0x01C)</th>
<th>RTR0(0x0801C/0x01C)</th>
<th>RTR1(0x0801D/0x01D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF0</td>
<td>0xA0</td>
<td></td>
</tr>
</tbody>
</table>

RCR (Retransmission Retry-Count Register) [R/W] [0x0801E/0x001E] [0x--08]
It configures the number of retransmission times. When retransmission occurs as many as ‘RCR+1’ times, Timeout interrupt is set (‘TIMEOUT’ bit of Sn_IR is set as ‘1’).
In TCP communication, the value of Sn_SSR is changed to ‘SOCK_CLOSED’ at the same time with Sn_IR(TIMEOUT) = ‘1’. Not in TCP communication, only Sn_IR(TIMEOUT) = ‘1’.

Ex) RCR = 0x0007

<table>
<thead>
<tr>
<th>RCR(0x0801E/0x01E)</th>
<th>RCR0(0x0801E/0x01C)</th>
<th>RCR1(0x0801F/0x01F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0x07</td>
<td></td>
</tr>
</tbody>
</table>

The timeout of W5300 can be configurable with RTR and RCR. W5300’s timeout has ARP and TCP retransmission timeout.
At the ARP(Refer to RFC 826, [http://www.ietf.org/rfc.html](http://www.ietf.org/rfc.html)) retransmission timeout, W5300 automatically sends ARP-request to the peer’s IP address in order to acquire MAC address information (used for communication of IP, UDP, or TCP). As waiting for ARP-response from the peer, if there is no response during the time set in RTR, Timeout occurs and ARP-request is retransmitted. It is repeated as many as ‘RCR + 1’ times.
Even after ARP-request retransmissions are repeated ‘RCR + 1’ times, if there is no ARP-response, the final timeout occurs and Sn_IR(TIMEOUT) becomes ‘1’. The value of final timeout (ARP_TO) of ARP-request is as below.

\[
ARP_{TO} = ( RTR \times 0.1ms ) \times ( RCR + 1 )
\]

At the TCP packet retransmission timeout, W5300 transmits TCP packets (SYN, FIN, RST, DATA packets) and waits for the acknowledgement (ACK) during the time set in RTR and RCR.
If there is no ACK from the peer, Timeout occurs and TCP packets (sent earlier) are retransmitted. The retransmissions are repeated as many as ‘RCR + 1’ times. Even after TCP packet retransmissions are repeated ‘RCR +1’ times, if there is no ACK from the peer, final timeout occurs and Sn_SSR is changed to ‘SOCK_CLOSED’ at the same time with Sn_IR(TIMEOUT) = ‘1’.

The value of final timeout (TCP_{TO}) of TCP packet retransmission can be calculated as below.

\[
\text{TCP}_{\text{TO}} = (\sum_{N=0}^{M} (\text{RTR} \times 2^N) + ((\text{RCR}-M) \times \text{RTR}_{\text{MAX}})) \times 0.1 \text{ms}
\]

- \(N\): Retransmission count, \(0 \leq N \leq M\)
- \(M\): Minimum value when \(\text{RTR} \times 2^{(M+1)} > 65535\) and \(0 \leq M \leq \text{RCR}\)
- \(\text{RTR}_{\text{MAX}}\): \(\text{RTR} \times 2^M\)

Ex) When \(\text{RTR} = 2000(\text{0x07D0})\), \(\text{RCR} = 8(\text{0x0008})\), \(\text{ARP}_{\text{TO}} = 2000 \times 0.1\text{ms} \times 9 = 1800\text{ms} = 1.8\text{s}\)
\(\text{TCP}_{\text{TO}} = (\text{0x07D0} + 0x0FA0 + 0x1F40 + 0x3E80 + 0x7D00 + 0xFA00 + 0xFA00 + 0xFA00 + 0xFA00) \times 0.1\ms\)
\[= (2000 + 4000 + 8000 + 16000 + 32000 + ((8 - 4) \times 64000)) \times 0.1\ms\]
\[= 318000 \times 0.1\ms = 31.8\text{s}\]
**TMSR (TX Memory Size Register) [R/W] [0x08020/0x020] [08.08.08.08.08.08.08.08]**

It configures internal TX memory size of each SOCKET. TX memory size of each SOCKET is configurable in the range of 0~64Kbytes. 8Kbytes is assigned when reset. Total memory size of each SOCKET’s TX memory (TMS_SUM) should be the multiple of 8. The sum of TMS_SUM and RMS_SUM (Total size of each SOCKET’s RX memory) is 128KBytes.

TMS01R (TX Memory Size of SOCKET0/1 Register) [R/W] [0x08020/0x020] [0x0808]

It configures internal TX memory size.

**Ex1) SOCKET0 : 4KB, SOCKET1 : 16KB**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS01R(0x08020/0x020)</td>
<td></td>
</tr>
<tr>
<td>TMSR0(0x08020/0x020)</td>
<td>4 (0x04)</td>
</tr>
<tr>
<td>TMSR1(0x08021/0x021)</td>
<td>16 (0x10)</td>
</tr>
</tbody>
</table>

TMS23R (TX Memory Size of SOCKET2/3 Register) [R/W] [0x08022/0x022] [0x0808]

It configures internal TX memory size of SOCKET2 and SOCKET3.

**Ex2) SOCKET2 : 1KB, SOCKET3 : 20KB**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS23R(0x08022/0x022)</td>
<td></td>
</tr>
<tr>
<td>TMSR2(0x08022/0x022)</td>
<td>1 (0x01)</td>
</tr>
<tr>
<td>TMSR3(0x08023/0x023)</td>
<td>20 (0x14)</td>
</tr>
</tbody>
</table>

TMS45R (TX Memory Size of SOCKET4/5 Register) [R/W] [0x08024/0x024] [0x0808]

It configures internal TX memory size of SOCKET4 and SOCKET5.

**Ex3) SOCKET4 : 0KB, SOCKET5 : 7KB**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS45R(0x08024/0x024)</td>
<td></td>
</tr>
<tr>
<td>TMSR4(0x08024/0x024)</td>
<td>0 (0x00)</td>
</tr>
<tr>
<td>TMSR5(0x08025/0x025)</td>
<td>7 (0x07)</td>
</tr>
</tbody>
</table>

TMS67R (TX Memory Size of SOCKET6/7 Register) [R/W] [0x08026/0x026] [0x0808]

It configures internal TX memory size of SOCKET6 and SOCKET7.

**Ex4) SOCKET6 : 12KB, SOCKET7 : 12KB**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS67R(0x08026/0x026)</td>
<td></td>
</tr>
<tr>
<td>TMSR6(0x08026/0x026)</td>
<td>12 (0x0C)</td>
</tr>
<tr>
<td>TMSR7(0x08027/0x027)</td>
<td>12 (0x0C)</td>
</tr>
</tbody>
</table>

As shown in above Ex1) ~ Ex4), TMS_SUM (TMSR0 + TMSR1 + TMSR2 + TMSR3 + TMSR4 + TMSR5 + TMSR6 + TMSR7) is 72, the multiple of 8 (72 % 8 = 0)
### RMSR (RX Memory Size Register) [R/W] [0x08028/0x028] [08.08.08.08.08.08.08.08.08]

It configures internal RX memory size of each SOCKET.

RX memory size of each SOCKET is configurable in the range of 0Kbyte ~ 64Kbytes. 8Kbytes is assigned when reset. The sum of RMS\textsubscript{SUM} and TMS\textsubscript{SUM} should be 128KB.

RMS01R (RX Memory Size of SOCKET0/1 Register) [R/W] [0x08028/0x028] [0x0808]

It configures internal RX memory size of SOCKET0 and SOCKET1.

**Ex5) SOCKET0 : 17KB, SOCKET1 : 3KB**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value (0x11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS01R</td>
<td>RMSR0</td>
</tr>
<tr>
<td>RMSR0</td>
<td>17</td>
</tr>
<tr>
<td>RMSR1</td>
<td>3</td>
</tr>
</tbody>
</table>

RMS23R (RX Memory Size of SOCKET2/3 Register) [R/W] [0x0802A/0x02A] [0x0808]

It configures internal RX memory size of SOCKET2 and SOCKET3.

**Ex6) SOCKET2 : 5KB, SOCKET3 : 16KB**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value (0x05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS23R</td>
<td>RMSR2</td>
</tr>
<tr>
<td>RMSR2</td>
<td>5</td>
</tr>
<tr>
<td>RMSR3</td>
<td>16</td>
</tr>
</tbody>
</table>

RMS45R (RX Memory Size of SOCKET4/5 Register) [R/W] [0x0802C/0x02C] [0x0808]

It configures internal RX memory size of SOCKET4 and SOCKET5.

**Ex7) SOCKET4 : 3KB, SOCKET5 : 4KB**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value (0x03)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS45R</td>
<td>RMSR4</td>
</tr>
<tr>
<td>RMSR4</td>
<td>3</td>
</tr>
<tr>
<td>RMSR5</td>
<td>4</td>
</tr>
</tbody>
</table>

RMS67R (TX Memory Size of SOCKET6/7 Register) [R/W] [0x0802E/0x02F] [0x0808]

It configures internal RX memory size of SOCKET6 and SOCKET7.

**Ex8) SOCKET6 : 4KB, SOCKET7 : 4KB**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value (0x04)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS67R</td>
<td>RMSR6</td>
</tr>
<tr>
<td>RMSR6</td>
<td>4</td>
</tr>
<tr>
<td>RMSR7</td>
<td>4</td>
</tr>
</tbody>
</table>

As shown above Ex1) ~ Ex8), RMS\textsubscript{SUM}(RMSR0 + RMSR1 + RMSR2 + RMSR3 + RMSR4 + RMSR5 + RMSR6 + RMSR7) is set as 56. The sum of TMS\textsubscript{SUM} and RMS\textsubscript{SUM} is 128.
### MTYPER(Memory Type Register) [R/W] [0x08030/0x030] [0x00FF]

W5300’s 128Kbytes data memory (Internal TX/RX memory) is composed of 16 memory blocks of 8Kbytes. MTYPER configures type of each 8KB memory block – RX or TX memory. The type of 8KB memory block corresponds to each bit of MTYPER. When the bit is ‘1’, it is used as TX memory, and the bit is ‘0’, it is used as RX memory. MTYPER is configured as TX memory type from the lower bit. The rest of the bits not configured as TX memory, should be set as ‘0’.

<table>
<thead>
<tr>
<th>MTYPER0 (0x08030)</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MTYPER1 (0x08031)</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### MTYPER(15:8)/MTYPER0(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>MB15</td>
<td>16th Memory Block Type</td>
</tr>
<tr>
<td>14</td>
<td>MB14</td>
<td>15th Memory Block Type</td>
</tr>
<tr>
<td>13</td>
<td>MB13</td>
<td>14th Memory Block Type</td>
</tr>
<tr>
<td>12</td>
<td>MB12</td>
<td>13th Memory Block Type</td>
</tr>
<tr>
<td>11</td>
<td>MB11</td>
<td>12th Memory Block Type</td>
</tr>
<tr>
<td>10</td>
<td>MB10</td>
<td>11th Memory Block Type</td>
</tr>
<tr>
<td>9</td>
<td>MB9</td>
<td>10th Memory Block Type</td>
</tr>
<tr>
<td>8</td>
<td>MB8</td>
<td>9th Memory Block Type</td>
</tr>
</tbody>
</table>

#### MTYPER(7:0)/MTYPER1(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>MB7</td>
<td>6th Memory Block Type</td>
</tr>
<tr>
<td>6</td>
<td>MB6</td>
<td>7th Memory Block Type</td>
</tr>
<tr>
<td>5</td>
<td>MB5</td>
<td>6th Memory Block Type</td>
</tr>
<tr>
<td>4</td>
<td>MB4</td>
<td>5th Memory Block Type</td>
</tr>
<tr>
<td>3</td>
<td>MB3</td>
<td>4th Memory Block Type</td>
</tr>
<tr>
<td>2</td>
<td>MB2</td>
<td>3rd Memory Block Type</td>
</tr>
<tr>
<td>1</td>
<td>MB1</td>
<td>2nd Memory Block Type</td>
</tr>
<tr>
<td>0</td>
<td>MB0</td>
<td>1st Memory Block Type</td>
</tr>
</tbody>
</table>
Ex1) TMS_{SUM} = 72, RMS_{SUM} = 56
As 72 / 8 = 9, from MB0 to MB8 are set as TX memory.

<table>
<thead>
<tr>
<th>Value</th>
<th>MTYPER0(0x08030/0x030)</th>
<th>MTYPER1(0x08031/0x031)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xFF</td>
<td></td>
</tr>
</tbody>
</table>

Ex2) TMS_{SUM} = 128, RMS_{SUM} = 0

<table>
<thead>
<tr>
<th>Value</th>
<th>MTYPER0(0x08030/0x030)</th>
<th>MTYPER1(0x08031/0x031)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF</td>
<td>0xFF</td>
<td></td>
</tr>
</tbody>
</table>

Ex3) TMS_{SUM} = 0, RMS_{SUM} = 128

<table>
<thead>
<tr>
<th>Value</th>
<th>MTYPER0(0x08030/0x030)</th>
<th>MTYPER1(0x08031/0x031)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

**PATR (PPPoE Authentication Type Register) [R] [0x08032/0x032] [0x0000]**

It notifies authentication method negotiated with PPPoE server.

W5300 supports 2 types of authentication methods.

<table>
<thead>
<tr>
<th>Value</th>
<th>Authentication method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC023</td>
<td>PAP</td>
</tr>
<tr>
<td>0xC223</td>
<td>CHAP</td>
</tr>
</tbody>
</table>

Ex) PATR = ‘CHAP’

<table>
<thead>
<tr>
<th>Value</th>
<th>PATR(0x08032/0x032)</th>
<th>PATR0(0x08032/0x032)</th>
<th>PATR1(0x08033/0x033)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC2</td>
<td>0x23</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**PTIMER (PPP Link Control Protocol Request Timer Register) [R/W] [0x08036/0x037] [0x–28]**

It configures transmitting timer of link control protocol (LCP) echo request. Value 1 is about 25ms.

Ex) PTIMER = 200 (200 * 25ms = 5000ms = 5s)

<table>
<thead>
<tr>
<th>Value</th>
<th>PTIMER(0x08036/0x037)</th>
<th>PTIMER0(0x08036/0x036)</th>
<th>PTIMER1(0x08037/0x037)</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>200 (0xC8)</td>
</tr>
</tbody>
</table>
PMAGICR(PPP LCP Magic number Register)[R/W][0x08038/0x038][0x--00]
It configures byte value to be used for 4bytes "Magic Number" during LCP negotiation with PPPoE server. For the detail, refer to “How to use PPPoE in W5300”.

Ex) PMAGICR = 0x01

<table>
<thead>
<tr>
<th>PMAGICR0(0x08038/0x038)</th>
<th>PMAGICR1(0x08039/0x039)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08038</td>
<td>0x038</td>
</tr>
<tr>
<td>Magic number = 0x01010101</td>
<td></td>
</tr>
</tbody>
</table>

PSIDR(PPPoE Session ID Register)[R][0x0803C/0x03C][0x0000]
It notifies PPP session ID to be used for communication with PPPoE server (acquired by PPPoE-process of W5300).

Ex) PSIDR = 0x0017

<table>
<thead>
<tr>
<th>PSIDR0(0x0803C/0x03C)</th>
<th>PSIDR1(0x0803D/0x03D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x17</td>
</tr>
</tbody>
</table>

PDHAR(PPPoE Destination Hardware Address Register)[R][0x08040/0x040]
[00.00.00.00.00.00]
It notifies hardware address of PPPoE server (acquired by PPPoE-process of W5300).

Ex) PDHAR = 00.01.02.03.04.05

<table>
<thead>
<tr>
<th>PDHAR0(0x08040/0x040)</th>
<th>PDHAR1(0x08041/0x041)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x01</td>
</tr>
<tr>
<td>PDHAR2(0x08042/0x042)</td>
<td>PDHAR3(0x08043/0x043)</td>
</tr>
<tr>
<td>0x02</td>
<td>0x03</td>
</tr>
<tr>
<td>PDHAR4(0x08044/0x044)</td>
<td>PDHAR5(0x08045/0x045)</td>
</tr>
<tr>
<td>0x04</td>
<td>0x05</td>
</tr>
</tbody>
</table>
High-performance Internet Connectivity Solution

WIZnet

UIPR (Unreachable IP Address Register) [R] [0x08048/0x048] [00.00.00.00]

UPORTR (Unreachable Port Register) [R] [0x0804C/0x04C] [0x0000]

When trying to transmit UDP data to destination port number which is not open, W5300 can receive ICMP (Destination port unreachable) packet. In this case, IR(DPUR) becomes ‘1’ and destination IP address and unreachable port number of ICMP packet can be acquired through UIPR and UPORTR.

Ex1) UIPR = 192.168.0.11

<table>
<thead>
<tr>
<th>UIPR(0x08048/0x048)</th>
<th>UIPR2(0x0804A/0x04A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIPR0(0x08048/0x048)</td>
<td>UIPR1(0x08049/0x049)</td>
</tr>
<tr>
<td>192 (0xC0)</td>
<td>168 (0xA8)</td>
</tr>
<tr>
<td>0 (0x00)</td>
<td>11 (0x0B)</td>
</tr>
</tbody>
</table>

Ex2) UPORT = 5000(0x1388)

<table>
<thead>
<tr>
<th>UPORTR(0x0804C/0x04C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPORTR0(0x0804C/0x04C)</td>
</tr>
<tr>
<td>UPORTR1(0x0804D/0x04D)</td>
</tr>
<tr>
<td>0x13</td>
</tr>
</tbody>
</table>

FMTUR (Fragment MTU Register) [R] [0x0804E/0x04E] [0x0000]

When communicating with the peer having a different MTU, W5300 can receive an ICMP(Fragment MTU) packet. At this case, IR(FMTU) becomes ‘1’ and destination IP address and fragment MTU value of ICMP packet can be acquired through UIPR and FMTUR. In order to keep communicating with the peer having Fragment MTU, set the FMTUR first in Sn_MSSR of the SOCKETn, and try the next communication.

Ex) FMTUR = 512(0x200)

<table>
<thead>
<tr>
<th>FMTUR(0x0804E/0x04E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMTUR0(0x0804E/0x04E)</td>
</tr>
<tr>
<td>FMTUR1(0x0804F/0x04F)</td>
</tr>
<tr>
<td>0x02</td>
</tr>
</tbody>
</table>

Pn_BRDYR (PIN "BRDYn" Configure Register) [R/W] [0x08060+4n/0x060+4n] [0x--00]

It configures the PIN "BRDYn" which is monitoring TX/RX memory status of the specified SOCKET. If the free buffer size of TX memory is same or bigger than the buffer depth of Pn_BDPTH, or received buffer size of RX memory is same or bigger than the Pn_BDPTH, PIN "BRDYn" is signaled.
### Pn_BRDYR0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Pn_BRDYR1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PEN</td>
<td>PIN &quot;BRDYn&quot; Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 : Disable BRDYn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : Enable BRDYn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When using PIN &quot;BRDYn&quot;, set it as &quot;1&quot;.</td>
</tr>
<tr>
<td>6</td>
<td>PMT</td>
<td>PIN Memory Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 : RX memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : TX memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It sets the type of SOCKET memory to monitor.</td>
</tr>
<tr>
<td>5</td>
<td>PPL</td>
<td>PIN Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 : Low sensitive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : High sensitive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When Free/Received buffer size of TX/RX memory is same or bigger than Pn_DPTHR, set the logic level of PIN &quot;BRDYn&quot; to be signalled to the host.</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>SN2</td>
<td>SOCKET Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set the SOCKET number to monitor through PIN &quot;BRDYn&quot;.</td>
</tr>
<tr>
<td>1</td>
<td>SN1</td>
<td>7 1 1 1 3 0 1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 1 1 0 2 0 1 0</td>
</tr>
<tr>
<td>0</td>
<td>SN0</td>
<td>5 1 0 1 0 0 0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 1 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
P0_BRDYR (PIN "BRDY0" Configure Register) [R/W] [0x08060/0x060] [0x--00]
It configures PIN "BRDY0".

P1_BRDYR (PIN "BRDY1" Configure Register) [R/W] [0x08064/0x064] [0x--00]
It configures PIN "BRDY1".

P2_BRDYR (PIN "BRDY2" Configure Register) [R/W] [0x08068/0x068] [0x--00]
It configures PIN "BRDY2".

P3_BRDYR (PIN "BRDY3" Configure Register) [R/W] [0x0806C/0x06C] [0x--00]
It configures PIN "BRDY3".

Pn_BDPTHR (PIN "BRDYn" Buffer Depth Register) [R/W] [0x08062/0x062] [0xUUUU]
It configures buffer depth of PIN "BRDYn". When monitoring TX memory and Sn_TX_FSR is same or bigger than Pn_DPTHR, the PIN "BRDYn" is signaled. When monitoring RX memory and if Sn_RX_RSR is same or bigger than Pn_BDPTHR, PIN "BRDYn" is signaled. The value for Pn_BDPTHR can't exceed TX/RX memory size allocated by TMSR or RMSR.

P0_BDPTHR (PIN "BRDY0" Buffer Depth Register) [R/W] [0x08062/0x062] [0xUUUU]
Sets buffer depth of PIN "BRDY0".

P1_BDPTHR (PIN "BRDY1" Buffer Depth Register) [R/W] [0x08066/0x066] [0xUUUU]
Sets buffer depth of PIN "BRDY1".

P2_BDPTHR (PIN "BRDY2" Buffer Depth Register) [R/W] [0x0806A/0x06A] [0xUUUU]
Sets buffer depth of PIN "BRDY2".

P3_BDPTHR (PIN "BRDY3" Buffer Depth Register) [R/W] [0x0806E/0x06E] [0xUUUU]
Sets buffer depth of PIN "BRDY3".

Ex) When monitoring if the free size of SOCKET5 TX memory is 2048 through PIN "BRDY3" with high sensitive,

P3_BRDYR = 0x00E5

<table>
<thead>
<tr>
<th>P3_BRDYR(0x0806C/0x06C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3_BRDYR0(0x0806C/0x06C)</td>
</tr>
<tr>
<td>Reserved</td>
</tr>
</tbody>
</table>
### P3_BDPTH = 2048(0x0800)

<table>
<thead>
<tr>
<th>P3_BDPTH(0x0806E/0x06E)</th>
<th>P3_BDPTH(0x0806F/0x06F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08</td>
<td>0x00</td>
</tr>
</tbody>
</table>

When monitoring RX memory of SOCKETn with 'BRDYRn', ‘BRDYRn' timing is as below.

- **a.** Sn_RX_RSR > Sn_BDPTH detected
- **b.** After 1 NIC_CLK, PIN ‘BRDYn’ is asserted high
- **c.** Sn_RX_RSR is decreased by host' RX memory Read, and "Sn_RX_RSR < Sn_BDPTH" is detected.
- **d.** After 1 NIC_CLK, PIN ‘BRDYn’ is de-asserted low.

**Assert Time** : Active Time of BRDYRn. It maintains during "Sn_RX_RSR > Sn_BDPTH" (at least 80ns).

Fig 4. ‘BRDYn’ Timing

#### IDR (Identification Register) [R] [0x080FE/0x0FF] [0x5300]

It notifies W5300's ID value.
### 4.4 SOCKET Registers

**Sn_MR (SOCKETn Mode Register) [R/W] [0x08200+0x40n/0x200+0x40n] [0x0000]**

It configures the protocol type or option of SOCKETn.

<table>
<thead>
<tr>
<th>Sn_MR0</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08200 + 0x40n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>ALIGN</td>
</tr>
<tr>
<td>0x200 + 0x40n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sn_MR1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08201 + 0x40n</td>
<td>MULTI</td>
<td>-</td>
<td>ND/MC</td>
<td>-</td>
<td>P3</td>
<td>P2</td>
<td>P1</td>
<td>P0</td>
</tr>
<tr>
<td>0x201 + 0x40n</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Sn_MR(15:8)/Sn_MR0(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>ALIGN</td>
<td>Alignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 : No use alignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : Use alignment</td>
</tr>
</tbody>
</table>

It is valid only in the TCP (P3 ~ P0 : "0001")

With TCP communication, when every the received DATA packet size is of even number and set as ‘1’, data receiving performance can be improved by removing PACKET-INFO(data size) that is attached to every the received DATA packet. For the detail, refer to "5.2.1.1 TCP SERVER"

#### Sn_MR(7:0)/Sn_MR1(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>MULTI</td>
<td>Multicasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 : Disable multicasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : Enable multicasting</td>
</tr>
</tbody>
</table>
## W5300

### High-performance Internet Connectivity Solution

#### It is valid only in UDP (P3~03 : "0010").
In order to implement multicasting, set the IP address and port number in Sn_DIPR and Sn_DPORTR respectively before "OPEN" command.

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>MAC Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MF</td>
<td>0 : Disable MAC filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : Enable MAC filter</td>
</tr>
</tbody>
</table>

#### It is valid in MACRAW(P3~P0 : "0100").
When this bit is set as ‘1’, W5300 can receive packet that is belong in itself or broadcasting. When this bit is set as ‘0’, W5300 can receive all packets on Ethernet. When using the hybrid TCP/IP stack, it is recommended to be set as ‘1’ for reducing the receiving overhead of host.

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>ND/IGMPv</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0 : using IGMP version 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 : using IGMP version 1</td>
</tr>
</tbody>
</table>

#### It is valid in case of MULTI='1' and UDP(P3~P0 : "0010").
It configures IGMP version to send IGMP message such as Join/Leave/Report to multicast-group.

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>Reserved</th>
</tr>
</thead>
</table>

#### Protocol
It configures communication protocol (TCP, UDP, IP RAW, MAC RAW) in each SOCKET or PPPoE SOCKET to operate with PPPoE server.
### Symbol Table

<table>
<thead>
<tr>
<th>Symbol</th>
<th>P3</th>
<th>P2</th>
<th>P1</th>
<th>P0</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sn_MR_CLOSE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Closed</td>
</tr>
<tr>
<td>Sn_MR_TCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>TCP</td>
</tr>
<tr>
<td>Sn_MR_UDP</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>UDP</td>
</tr>
<tr>
<td>Sn_MR_IPRAW</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>IP RAW</td>
</tr>
<tr>
<td>S0_MR_MACRAW</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>MAC RAW</td>
</tr>
<tr>
<td>S0_MR_PPPoE</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

S0_MR_MACRAW and S0_MR_PPPoE are valid only in SOCKET0.
S0_MR_PPPoE is temporarily used for PPPoE server connection/termination. After PPPoE connection is established, it can be used as another protocol.

### Sn_CR Command

**Sn_CR (SOCKETn Command Register) [R/W] [0x08202+0x40n/0x202+0x40n] [0x--00]**

It sets command type such as open, close, connect, listen, send, recv for SOCKETn. When W5300 detects any command, Sn_CR is automatically cleared to 0x00. Even though Sn_CR is cleared to 0x00, the command can be still performing. It can be checked by Sn_IR or Sn_SSR if command is completed or not.

<table>
<thead>
<tr>
<th>Sn_CR(0x08202+0x40n/0x202+0x40n)</th>
<th>Sn_CR0(0x08202+0x40n/0x202+0x40n)</th>
<th>Sn_CR1(0x08203+0x40n/0x203+0x40n)</th>
<th>Reserved</th>
<th>Command</th>
</tr>
</thead>
</table>

#### Sn_CR(7:0)/Sn_CR1(7:0)

<table>
<thead>
<tr>
<th>Value</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>OPEN</td>
<td>It initializes SOCKETn and opens according to protocol type set in Sn_MR(P3:P0).</td>
</tr>
</tbody>
</table>

Below is the value change of Sn_SSR according to Sn_MR(P3:P0):

<table>
<thead>
<tr>
<th>Sn_MR(P3:P0)</th>
<th>Sn_SSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sn_MR_CLOSE</td>
<td>-</td>
</tr>
<tr>
<td>Sn_MR_TCP</td>
<td>SOCK_INIT</td>
</tr>
<tr>
<td>Sn_MR_UDP</td>
<td>SOCK_UDPD</td>
</tr>
<tr>
<td>Sn_MR_IPRAW</td>
<td>SOCK_IPRAW</td>
</tr>
<tr>
<td>S0_MR_MACRAW</td>
<td>SOCK_MACRAW</td>
</tr>
<tr>
<td>S0_MR_PPPoE</td>
<td>SOCK_PPPoE</td>
</tr>
<tr>
<td></td>
<td>LISTEN</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 0x02 | It is valid only in TCP mode (Sn_MR(P3:P0)=Sn_MR_TCP). It operates SOCKETn as "TCP SERVER". It changes Sn_SSR to SOCK_LISTEN at the SOCK_INIT in order to wait for connect-request (SYN packet) from any "TCP CLIENT". When Sn_SSR is SOCK_LISTEN and connect-request from a "TCP CLIENT" is successfully processed, Sn_IR(0) becomes '1' and Sn_SSR is changed to SOCK_ESTABLISHED. In case that the connect-request is not processed (SYN/ACK transmission is failed), TCP_TO occurs (Sn_IR(3)='1') and Sn_SSR is changed to SOCK_CLOSED.  

   cf> If TCP connect-request port number of "TCP CLIENT" does not exist, W5300 transmits RST packet and Sn_SSR is not changed. |

<table>
<thead>
<tr>
<th></th>
<th>CONNECT</th>
<th></th>
</tr>
</thead>
</table>
| 0x04 | Only valid in TCP mode. It operates SOCKETn as "TCP CLIENT". It transmits connect-request(SYN packet) to the "TCP SERVER" designated with Sn_DIPR and Sn_DPORTR. When connect-request is successfully processed (when receiving SYN/ACK packet), Sn_IR(0) becomes '1', and Sn_SSR is changed to SOCK_ESTABLISHED. There are 3 cases if connect-request is failed  

   - when ARP_TO occurs (Sn_IR(3)='1') because Destination Hardware Address is not acquired through ARP process  
   - when SYN/ACK packet is not received and TCP_TO (Sn_IR(3) is '1')  
   - When RST packet is received instead of SYN/ACK packet. In above 3 cases, Sn_SSR is changed to SOCK_CLOSED. |

<table>
<thead>
<tr>
<th></th>
<th>DISCON</th>
<th></th>
</tr>
</thead>
</table>
| 0x08 | Only valid in TCP mode. Regardless of "TCP SERVER" or "TCP CLIENT", it performs disconnect-process.  

   - Active close : it transmits disconnect-request(FIN packet) to the connected peer. |
<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLOSE</strong> (0x10)</td>
<td>It closes SOCKETn. Sn_SSR is changed to SOCK_CLOSED.</td>
</tr>
<tr>
<td><strong>SEND</strong> (0x20)</td>
<td>It transmits data as big as the size of Sn_TX_WRSR. At the TCP or UDP mode, if Sn_TX_WRSR is bigger than maximum segment size (MSS), W5300 automatically divides data in MSS unit, and transmits the divided data (DATA packet). However, this function is not supported in IPRAW or MACRAW mode. The host should divide the data in MSS unit and transmit the divided data. When completing the SEND process, Sn_IR (SENDOK) becomes ‘1’. After checking Sn_IR(SENDOK) = ‘1’, the host can give SEND command to the next data. If DATA packet is successfully transmitted to the peer by SEND (when DATA/ACK packet is received from the peer), Sn_TX_FSR is increased by the size of transmitting DATA packet. If not (when DATA/ACK packet is not received), TCP_TO occurs (Sn_IR(3)=‘1’) and Sn_SSR is changed to SOCK_CLOSED. cf&gt; Host copies TX data into internal TX memory of SOCKETn through Sn_TX_FIFOR before SEND command, and set the data size to Sn_TX_WRSR.</td>
</tr>
<tr>
<td><strong>SEND_MAC</strong> (0x21)</td>
<td>Valid only in UDP (Sn_MR(P3:P0)=Sn_MR_UDP) or IPRAW((Sn_MR(P3:P0) = Sn_MR_IPRAW) mode.</td>
</tr>
</tbody>
</table>

- **Passive close**: When receiving disconnect-request (FIN packet) from the peer, it transmits FIN packet.
- If disconnect-request is successful (when receiving FIN/ACK packet), Sn_SSR is changed to SOCK_CLOSED.
- If disconnect-request is failed, TCP_TO occurs (Sn_IR(3)=‘1’) and Sn_SSR is changed to SOCK_CLOSED.

cf> If CLOSE is used instead of DISCON, only Sn_SSR is changed to SOCK_CLOSED without disconnect-process(disconnect-request). If RST packet is received from the peer during communication, Sn_SSR is unconditionally changed to SOCK_CLOSED.
### High-performance Internet Connectivity Solution

**W5300**

The basic operation is same as **SEND**.

**SEND** transmits data after acquiring destination hardware address through ARP-process, but **SEND_MAC** transmits data by regarding Sn_DHAR as destination hardware address. **SEND_MAC** can reduce network traffic by removing ARP-process when sending UDP or IP raw data to the destination.

<table>
<thead>
<tr>
<th>0x22</th>
<th><strong>SEND_KEEP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid only in TCP mode.</td>
<td></td>
</tr>
<tr>
<td>In order to check TCP connection status with the peer, <strong>KEEP ALIVE(KA)</strong> packet can be transmitted. <strong>SEND_KEEP</strong> is available only in case of ‘Sn_KPALVTR=0’, but ignored in case of ‘Sn_KPALVTR&gt;0’. In case of ‘Sn_KPALVTR &gt; 0’, KA packet is automatically transmitted if there is no data communication during the time of Sn_KPALVTR.</td>
<td></td>
</tr>
<tr>
<td>If KA packet is successfully transmitted (when KA/ACK packet is received from the peer), Sn_SSR maintains SOCK_ESTABLISHED status. If it is failed to transmit the KA packet (when the peer already closed the connection, or KA/ACK is not transmitted), TCP_TO will occurs (Sn_IR(3)='1') and Sn_SSR is changed to SOCK_CLOSED.</td>
<td></td>
</tr>
<tr>
<td>cf&gt; KA packet can be transmitted after one or more data communication is processed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0x40</th>
<th><strong>RECV</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>It notifies that the host received the data packet of SOCKETn</td>
<td></td>
</tr>
<tr>
<td>cf&gt; Before RECV command, the host should copy receiving data packet from internal RX memory into the host memory through Sn_RX_FIFOR.</td>
<td></td>
</tr>
</tbody>
</table>

**Below commands are valid at the SOCKET0 and S0_MR(P3:P0)=S0_MR_PPPoE.**

*For more detail, refer to “How to use PPPoE in W5300”.

<table>
<thead>
<tr>
<th>0x23</th>
<th><strong>PCON</strong></th>
<th>PPPoE connection begins by transmitting PPPoE discovery packet.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td><strong>PDISCON</strong></td>
<td>Closes PPPoE connection.</td>
</tr>
<tr>
<td>0x25</td>
<td><strong>PCR</strong></td>
<td>In each phase, it transmits REQ message.</td>
</tr>
<tr>
<td>0x26</td>
<td><strong>PCN</strong></td>
<td>In each phase, it transmits NAK message.</td>
</tr>
<tr>
<td>0x27</td>
<td><strong>PCJ</strong></td>
<td>In each phase, it transmits REJECT message.</td>
</tr>
</tbody>
</table>
Sn_IMR (SOCKETn Interrupt Mask Register) [R/W] [0x08204+0x40n/0x204+0x40n] [0x--FF]

It configures the interrupt of SOCKETn so as to notify to the host.

Interrupt mask bit of Sn_IMR corresponds to interrupt bit of Sn_IR. If interrupt occurs in any SOCKET and the bit is set as '1', its corresponding bit of Sn_IR is set as '1'. When the bits of Sn_IMR and Sn_IR are '1', IR(n) becomes '1'. At this time, if IMR(n) is '1', the interrupt is issued to the host ('/INT' signal is asserted low.)

<table>
<thead>
<tr>
<th>Sn_IMR0</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08204 + 0x40n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0x204 + 0x40n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sn_IMR1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08205 + 0x40n</td>
<td>PRECV</td>
<td>PFAIL</td>
<td>PNEXT</td>
<td>SENDOK</td>
<td>TIMEOUT</td>
<td>RECV</td>
<td>DISCON</td>
<td>CON</td>
</tr>
<tr>
<td>0x205 + 0x40n</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sn_IMR(15:8)/Sn_IMR0(7:0) : All Reserved

Sn_IMR(7:0)/Sn_IMR1(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PRECV</td>
<td>Sn_IR(PRECV) Interrupt Mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid only in case of 'SOCKET=0' &amp; 'S0_MR(P3:P0)=S0_MR_PPPoE'</td>
</tr>
<tr>
<td>6</td>
<td>PFAIL</td>
<td>Sn_IR(PFAIL) Interrupt Mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid only in case of 'SOCKET=0' &amp; 'S0_MR(P3:P0)=S0_MR_PPPoE'</td>
</tr>
<tr>
<td>5</td>
<td>PNEXT</td>
<td>Sn_IR(PNEXT) Interrupt Mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid only in case of 'SOCKET=0' &amp; 'S0_MR(P3:P0)=S0_MR_PPPoE'</td>
</tr>
<tr>
<td>4</td>
<td>SENDOK</td>
<td>Sn_IR(SENDOK) Interrupt Mask</td>
</tr>
<tr>
<td>3</td>
<td>TIMEOUT</td>
<td>Sn_IR(TIMEOUT) Interrupt Mask</td>
</tr>
<tr>
<td>2</td>
<td>RECV</td>
<td>Sn_IR(RECV) Interrupt Mask</td>
</tr>
<tr>
<td>1</td>
<td>DISCON</td>
<td>Sn_IR(DISCON) Interrupt Mask</td>
</tr>
<tr>
<td>0</td>
<td>CON</td>
<td>Sn_IR(CON) Interrupt Mask</td>
</tr>
</tbody>
</table>

Sn_IR (SOCKETn Interrupt Register) [R/W] [0x08206+0x40n/0x206+0x40n] [0x--00]

Sn_IR is the register to notify interrupt type (establishment, termination, receiving data, timeout) of SOCKETn to the host.

When any Interrupt occurs and the mask bit of Sn_IMR is '1', the interrupt bit of Sn_IR becomes '1'.

© Copyright 2008-2014 WIZnet Co., Ltd. All rights reserved.
In order to clear the bit of Sn_IR which is set as '1', the host should write the bit as '1'. When all the bits of Sn_IR is cleared as '0', IR(n) is automatically cleared.

<table>
<thead>
<tr>
<th>Sn_IR0</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08206 + 0x40n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0x206 + 0x40n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sn_IR1</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x08207 + 0x40n</td>
<td>PRECV</td>
<td>PFAIL</td>
<td>PNEXT</td>
<td>SENDOK</td>
<td>TIMEOUT</td>
<td>RECV</td>
<td>DISCON</td>
<td>CON</td>
</tr>
<tr>
<td>0x207 + 0x40n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Sn_IR(15:8)/Sn_IR0(7:0) : All Reserved

### Sn_IR(7:0)/Sn_IR1(7:0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7   | PRECV  | PPP Receive Interrupt  
Setting for the case that option data which is not supported is received |
| 6   | PFAIL  | PPP Fail Interrupt 
Setting for the case that PAP authentication is failed |
| 5   | PNEXT  | PPP Next Phase Interrupt 
Setting for the case that the phase is changed during PPPoE connection process |
| 4   | SENDOK | SEND OK Interrupt 
Setting for the case that the SEND command is completed |
| 3   | TIMEOUT | TIMEOUT Interrupt 
Setting for the case that ARP_TO or TCP_TO occurs |
| 2   | RECV   | Receive Interrupt 
Setting for the case whenever data packet is received from the peer |
| 1   | DISCON | Disconnect Interrupt 
Setting for the case that FIN or FIN/ACK packet is received from the peer |
| 0   | CON    | Connect Interrupt 
Setting for the case that the connection with the peer is successfully established. |

Sn_SSR (SOCKETn Status Register) [R] [0x08208+0x40n/0x208+0x40n] [0x--00]

It notifies the status of SOCKETn. The status of SOCKETn can be changed by command of Sn_CR or packet transmission/receipt.
### Sn_SSR(15:8)/Sn_SSR0(7:0)
All Reserved

### Sn_SSR(7:0)/Sn_SSR1(7:0)

<table>
<thead>
<tr>
<th>Value</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SOCK_CLOSED</td>
<td>It is the status that resource of SOCKETn is released. When DISCON or CLOSE command is performed, or ARP\textsubscript{TO}, or TCP\textsubscript{TO} occurs, it is changed to SOCK_CLOSED regardless of previous value.</td>
</tr>
<tr>
<td>0x13</td>
<td>SOCK_INIT</td>
<td>It is the status that SOCKETn is open as TCP mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is changed to SOCK_INIT when Sn_MR(P3:P0) is Sn_MR_TCP and OPEN command is performed. It is the initial step of TCP connection establishment.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is possible to perform LISTEN command at the &quot;TCP SERVER&quot; mode and CONNECT command at the &quot;TCP CLIENT&quot;.</td>
</tr>
<tr>
<td>0x14</td>
<td>SOCK_LISTEN</td>
<td>It is the status that SOCKETn operates as &quot;TCP SERVER&quot; and waits for connection-request (SYN packet) from &quot;TCP CLIENT&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When LISTEN command is performed, it is changed to SOCK_LISTEN.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When connect-request(SYN packet) from &quot;TCP CLIENT&quot; is successfully processed, SOCK_LISTEN is changed to SOCK_ESTABLISHED. If it is failed, TCP\textsubscript{TO} occurs(Sn_IR(TIME OUT)=1) and changed to SOCK_CLOSED.</td>
</tr>
<tr>
<td>0x17</td>
<td>SOCK_ESTABLISHED</td>
<td>It is the status that TCP connection is established.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is changed to SOCK_ESTABLISHED when SYN packet from &quot;TCP CLIENT&quot; is successfully processed at the SOCK_LISTEN, or CONNECT command is successfully performed. At this status, DATA packet can be transferred,</td>
</tr>
<tr>
<td>Status Code</td>
<td>Mode</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x1C</td>
<td>SOCK_CLOSE_WAIT</td>
<td>It is the status that disconnect-request (FIN packet) is received from the peer. As TCP connection is half-closed, it is possible to transfer data packet. In order to complete the TCP disconnection, DISCON command should be performed. For SOCKETn close without disconnection-process, CLOSE command should be just performed.</td>
</tr>
<tr>
<td>0x22</td>
<td>SOCK_UDP</td>
<td>It is the status that SOCKETn is open as UDP mode. It is changed to SOCK_UDP when Sn_MR(P3:P0) is Sn_MR_UDP and OPEN command is performed. DATA packet can be transferred without connection that is necessary to TCP mode SOCKET.</td>
</tr>
<tr>
<td>0x32</td>
<td>SOCK_IPRAW</td>
<td>It is the status that SOCKETn is open as IPRAW mode. It is changed to SOCK_IPRAW when Sn_MR(P3:P0) is Sn_MR_IPRAW and OPEN command is performed. IP packet can be transferred without connection such like SOCK_UDP.</td>
</tr>
<tr>
<td>0x42</td>
<td>SOCK_MACRAW</td>
<td>It is the status that SOCKET0 is open as MACRAW mode. It is changed to SOCK_MACRAW in case of S0_MR (P3:P0)=S0_MR_MACRAW and S0_CR=OPEN. MAC packet (Ethernet frame) can be transferred such like SOCK_UDP.</td>
</tr>
<tr>
<td>0x5F</td>
<td>SOCK_PPPoE</td>
<td>It is the status that SOCKET0 is open as PPPoE mode. It is changed to SOCK_PPPoE in case of S0_MR (P3:P0)=S0_MR_PPPoE and S0_CR=OPEN. It is temporarily used at the PPPoE connection. For the detail, refer to “How to use PPPoE in W5300”.</td>
</tr>
</tbody>
</table>

Below shows temporary status that can be observed during Sn_SSR is changed.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x15</td>
<td>SOCK_SYNSENT</td>
<td>It is the status that connect-request (SYN packet) is transmitted to &quot;TCP SERVER&quot;.</td>
</tr>
</tbody>
</table>
This status shows changing process from SOCK_INIT to 
SOCK_ESTABLISHED by CONNECT command.

At this status, if connect-accept(SYN/ACK packet) is received 
from "TCP SERVER", it is automatically changed to SOCK_ 
ESTABLISHED. If SYN/ACK packet is not received from the 
"TCP SERVER" before TCP_TO occurs (Sn_IR(TIMEOUT)='1'),
it is changed to SOCK_CLOSED.

| 0x16 | SOCK_SYNRECV | It is the status that connect-request(SYN packet) is received 
from "TCP CLIENT".

It is automatically changed to SOCK_ESTABLISHED when 
W5300 successfully transmits connect-accept (SYN/ACK 
packet) to the "TCP CLIENT". If it is failed, TCP_TO occurs 
(Sn_IR(TIMEOUT)='1'), and it is changed to SOCK_CLOSED.

| 0x18 | SOCK_FIN_WAIT | It is the status that SOCKETn is closed.

It is observed in the disconnect-process of active close or 
passive close. It is changed to SOCK_CLOSED when 
disconnect-process is successfully finished or TCP_TO occurs 
(Sn_IR (TIMEOUT)=’1’).

| 0x1B | SOCK_TIME_WAIT | It is the status that ARP-request is transmitted in order to 
acquire destination hardware address.

This status is observed when SEND command is performed 
at the SOCK_UDP or SOCK_IPRAW, or CONNECT 
command is performed at the SOCK_INIT.

If hardware address is successfully acquired from destination 
(when ARP-response is received), it is changed to 
SOCK_UDP, SOCK_IPRAW or SOCK_SYNSENT. If it's 
failed and ARP_TO occurs (Sn_IR(TIMEOUT)=’1’), in case of 
UDP or IPRAW mode it goes back to the previous status(the 
SOCK_UDP or SOCK_IPRAW), in case of TCP mode it goes 
to the SOCK_CLOSED.

cf> ARP-process operates at the SOCK_UDP or 
SOCK_IPRAW when the previous and current values of
Sn_DIPR are different. If the previous and current values of Sn_DIPR are same, ARP-process doesn’t operate because the destination hardware address is already acquired.

Sn_PORT(R)(SOCKETn Source Port Register)[R/W]
[0x0820A+0x40n/0x20A+0x40n] [0x0000]
It sets source port number.
It is valid when SOCKETn is used as TCP or UDP mode, and ignored when used as other modes.
It should be set before OPEN command.
Ex) Sn_PORTR = 5000(0x1388)

<table>
<thead>
<tr>
<th>Sn_PORTR</th>
<th>Sn_PORTR1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0820A+0x40n/0x20A+0x40n</td>
<td>0x0820B+0x40n/0x20B+0x40n</td>
</tr>
</tbody>
</table>

Sn_DHAR (SOCKETn Destination Hardware Address Register) [R/W]
[0x0820C+0x40n/0x20C+0x40n] [FF,FF,FF,FF,FF]

It sets or is set as destination hardware address of SOCKETn. Also, if SOCKET0 is used for PPPoE mode, S0_DHAR sets as PPPoE server hardware address that is already known.

When using SEND_MAC command at the UDP or IPRAW mode, it sets destination hardware address of SOCKETn. At the TCP, UDP and IPRAW mode, Sn_DHAR is set as destination hardware address that is acquired by ARP-process of CONNECT or SEND command. The host can acquire the destination hardware address through Sn_DHAR after successfully performing CONNET or SEND command.

When using PPPoE-process of W5300, PPPoE server hardware address is not required to be set.

However, even if PPPoE-process of W5300 is not used, but implemented by yourself with MACRAW mode, in order to transmit or receive the PPPoE packet, PPPoE server hardware address(acquired by your PPPoE-process), PPPoE server IP address, and PPP session ID should be set, and MR(PPPoE) also should be set as ‘1’.

S0_DHAR sets PPPoE server hardware address before OPEN command. PPPoE server hardware address which is set by S0_DHAR is applied to PDHAR after performing OPEN command.

The configured PPPoE information is internally valid even after CLOSE command.

Ex) Sn_DHAR = 00.08.DC.01.02.10

<table>
<thead>
<tr>
<th>Sn_DHAR</th>
<th>Sn_DHAR1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0820C+0x40n/0x20C+0x40n</td>
<td>0x0820D+0x40n/0x20D+0x40n</td>
</tr>
<tr>
<td>0x00</td>
<td>0x08</td>
</tr>
<tr>
<td>Sn_DHAR2</td>
<td>Sn_DHAR3</td>
</tr>
<tr>
<td>0x0820E+0x40n/0x20E+0x40n</td>
<td>0x0820F+0x40n/0x20F+0x40n</td>
</tr>
<tr>
<td>0xDC</td>
<td>0x01</td>
</tr>
<tr>
<td>Sn_DHAR4</td>
<td>Sn_DHAR5</td>
</tr>
<tr>
<td>0x08210+0x40n/0x210+0x40n</td>
<td>0x08211+0x40n/0x211+0x40n</td>
</tr>
<tr>
<td>0x02</td>
<td>0x10</td>
</tr>
</tbody>
</table>
Sn_DPORTR (SOCKETn Destination Port Register) [WO]
[0x08212+0x40n/0x212+0x40n] [0x0000]
It sets as destination port number of SOCKETn. If SOCKET0 is used as PPPoE mode, S0_DPORTR sets PPP session ID that is already known.
It is valid only in TCP, UDP or PPPoE mode, and ignored in other modes.
At the TCP mode, when operating as "TCP CLIENT" it sets as the listen port number of "TCP SERVER" before performing CONNECT command.
At the UDP mode, Sn_DPORTR sets as the destination port number to be used for transmitting UDP DATA packet before performing SEND or SEND_MAC command.
At the PPPoE mode, S0_DPORTR sets as PPP session ID that is already known. PPP session ID (set by S0_DPORTR) is applied to PSIDR after performing OPEN command.

Ex) Sn_DPORTR = 5000(0x1388)

<table>
<thead>
<tr>
<th>Sn_PORTR0(0x08212+0x40n/0x212+0x40n)</th>
<th>Sn_PORTR1(0x08213+0x40n/0x213+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x13</td>
<td>0x88</td>
</tr>
</tbody>
</table>

Sn_DIPR (SOCKETn Destination IP Address Register) [R/W]
[0x08214+0x40n/0x214+0x40n] [00.00.00.00]
It sets or is set as destination IP address of SOCKETn. If SOCKET0 is used as PPPoE mode, S0_DIPR sets PPPoE server IP address that is already known.
It is valid only in TCP, UDP, IPRAW or PPPoE mode, but ignored in MACRAW mode.
At the TCP mode, when operating as "TCP CLIENT" it sets as IP address of "TCP SERVER" before performing CONNECT command and when operating as "TCP SERVER", it is internally set as IP address of "TCP CLIENT" after successfully establishing connection.
At the UDP or IPRAW mode, Sn_DIPR sets as destination IP address to be used for transmitting UDP or IPRAW DATA packet before performing SEND or SEND_MAC command.
At the PPPoE mode, S0_DIPR sets as PPPoE server IP address that is already known.

Ex) Sn_DIPR = 192.168.0.11

<table>
<thead>
<tr>
<th>Sn_DIPR0(0x08214+0x40n/0x214+0x40n)</th>
<th>Sn_DIPR1(0x08215+0x40n/0x215+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>192 (0xC0)</td>
<td>168 (0xA8)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sn_DHAR2(0x08216+0x40n/0x216+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0x00)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sn_DIPR2(0x08216+0x40n/0x216+0x40n)</th>
<th>Sn_DIPR3(0x08217+0x40n/0x217+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 (0x0B)</td>
<td></td>
</tr>
</tbody>
</table>
Sn_MSSR (SOCKETn Maximum Segment Size Register) [R/W] 
[0x08218+0x40n/0x218+0x40n] [0x0000]

It sets MTU(Maximum Transfer Unit) of SOCKETn or notifies MTU that is already set.
If the host does not set the Sn_MSSR, it is set as default MTU.
It just supports TCP or UDP mode. When using PPPoE (MR(PPPoE)=’1’), the MTU of TCP or UDP mode is assigned in the range of MTU of PPPoE.
At the IPRAW or MACRAW, MTU is not processed internally, but default MTU is used. Therefore, when transmitting the data bigger than default MTU, the host should manually divide the data into the unit of default MTU.
At the TCP or UDP mode, if transmitting data is bigger than MTU, W5300 automatically divides the data into the unit of MTU.
MTU is called as MSS at the TCP mode. By selecting from Host-Written-Value and peer's MSS, MSS is set as smaller value through TCP connection process.
At the UDP mode, there is no connection-process of TCP mode, and Host-Written-Value is just used. When communicating with the peer having different MTU, W5300 is able to receive ICMP(Fragment MTU) packet. In this case, IR(FMTU) becomes ‘1’, and the host can acquire the fragment MTU and destination IP address through FMTUR and UIPR respectively. In case of IR(FMTU)=’1’, the UDP communication with the peer, is not possible. So, you should close the SOCKET, set FMTU as Sn_MSSR and retry the communication with OPEN command.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Normal (MR(PPPoE)=’0’)</th>
<th>PPPoE (MR(PPPoE)=’1’)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default MTU</td>
<td>Range</td>
</tr>
<tr>
<td>TCP</td>
<td>1460</td>
<td>1 ~ 1460</td>
</tr>
<tr>
<td>UDP</td>
<td>1472</td>
<td>1 ~ 1472</td>
</tr>
<tr>
<td>IPRAW</td>
<td>1480</td>
<td>1472</td>
</tr>
<tr>
<td>MACRAW</td>
<td>1514</td>
<td></td>
</tr>
</tbody>
</table>

Ex) Sn_MSSR = 1460 (0x05B4)

<table>
<thead>
<tr>
<th>Sn_MSSR (0x08218+0x40n/0x218+0x40n)</th>
<th>Sn_MSSR0 (0x08218+0x40n/0x218+0x40n) Sn_MSSR1 (0x08219+0x40n/0x219+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x05</td>
<td>0xB4</td>
</tr>
</tbody>
</table>

Sn_KPALVTR(SOCKETn Keep Alive Time Register)[R/W] 
[0x0821A+0x40n/0x21A+0x40n] [0x00]

It is 1 byte register that sets transmitting timer of KEEP ALIVE(KA) packet of SOCKETn. It is valid only in TCP mode, and ignored in other modes. The unit is 5s.
KA packet can be transmitted after Sn_SSR is changed to SOCK_ESTABLISHED and more than one time DATA packet transmitting or receiving. In case of 'Sn_KPALVTR > 0', W5300 automatically transmits KA packet after time-period, and checks TCP connection (Auto-keep-alive-process). In case of 'Sn_KPALVTR = 0', Auto-keep-alive-process does not operate, and KA packet can be transmitted by SEND_KEEP command by the host (Manual-keep-alive-process). Manual-keep-alive-process is ignored in case of 'Sn_KPALVTR > 0'.

Ex) In case of 'Sn_KPALVTR = 10', KA packet is transmitted every 50s.

<table>
<thead>
<tr>
<th>Sn_PROTOR(0x0821A+0x40n/0x21A+0x040n)</th>
<th>Sn_KPALVTR(0x0821A+0x40n/0x21A+0x040n)</th>
<th>Sn_PROTOR (0x0821B+0x40n/0x21B+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 (0x0A)</td>
<td>Sn_PROTOR(0x0821A+0x40n/0x21A+0x040n)</td>
<td>Sn_PROTOR</td>
</tr>
</tbody>
</table>
Sn_TTLR (SOCKETn TTL Register) [R/W] [0x0821E+0x40n/0x21E+0x40n] [0x80]
It sets TTL(Time To Live) field of IP header at the IP layer. It should be set before OPEN command. Refer to http://www.iana.org/assignments/ip-parameters.

Ex) Sn_TTLR = 128 (0x80)

<table>
<thead>
<tr>
<th>Sn_TTLR(0x0821E+0x40n/0x21E+0x40n)</th>
<th>Sn_TTLR0(0x0821E+0x40n/0x21E+0x40n)</th>
<th>Sn_TTLR1(0x0821F+0x40n/0x21F+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0x00</td>
<td>0x80</td>
</tr>
</tbody>
</table>

Sn_TX_WRSR (SOCKETn TX Write Size Register) [R/W]
[0x08220+0x40n/0x220+0x40n] [0x00000000]
It sets the byte size of the data written in internal TX memory through Sn_TX_FIFOR. It is set before SEND or SEND_MAC command, and can't be bigger than internal TX memory size set by TMSRn.

W5300 automatically divides the data in the unit of Sn_MSSR in case of ‘Sn_TX_WRSR > Sn_MSSR’ at the TCP or UDP mode. In other modes, Sn_TX_WRSR should not be set bigger than Sn_MSSR.

Ex1) Sn_TX_WRSR = 64KB = 65536 = 0x00010000

<table>
<thead>
<tr>
<th>Sn_TX_WRSR(0x08220+0x40n/0x220+0x40n)</th>
<th>Sn_TX_WRSR0(0x08220+0x40n/0x220+0x40n)</th>
<th>Sn_TX_WRSR1(0x08221+0x40n/0x221+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sn_TX_WRSR2(0x08222+0x40n/0x222+0x40n)</td>
<td>Sn_TX_WRSR3(0x08223+0x40n/0x21D+0x40n)</td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td></td>
<td>0x00</td>
</tr>
</tbody>
</table>

Ex2) Sn_TX_WRSR = 2017 = 0x000007E1

<table>
<thead>
<tr>
<th>Sn_TX_WRSR(0x08220+0x40n/0x220+0x40n)</th>
<th>Sn_TX_WRSR0(0x08220+0x40n/0x220+0x40n)</th>
<th>Sn_TX_WRSR1(0x08221+0x40n/0x221+0x40n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sn_TX_WRSR2(0x08222+0x40n/0x222+0x40n)</td>
<td>Sn_TX_WRSR3(0x08223+0x40n/0x223+0x40n)</td>
<td></td>
</tr>
<tr>
<td>0x07</td>
<td></td>
<td>0xE1</td>
</tr>
</tbody>
</table>
Sn_TX_FSR (SOCKETn TX Free Size Register) [R]
[0x08224+0x40n/0x224+0x40n] [0x00002000]

It notifies the free size of internal TX memory (the byte size of transmittable data) of SOCKETn. The host can’t write data through Sn_TX_FIFOR as the size bigger than Sn_TX_FSR. Therefore, be sure to check Sn_TX_FSR before transmitting data, and if data size is smaller than or same as Sn_TX_FSR, transmit the data with SEND or SEND_MAC command after copying the data.

At the TCP mode, if the peer checks the transmitted DATA packet (if DATA/ACK packet is received from the peer), Sn_TX_FSR is automatically increased by the size of transmitted DATA packet. At the other modes, when Sn_IR(SENDOK) is ‘1’, Sn_TX_FSR is automatically increased by the size of transmitted data.

Ex1) Sn_TX_FSR = 64KB = 65536 = 0x00010000

<table>
<thead>
<tr>
<th>Sn_TX_FSR(0x08224+0x40n/0x224+0x40n)</th>
<th>Sn_TX_FSR0(0x08224+0x40n/0x214+0x040n)</th>
<th>Sn_TX_FSR1(0x08225+0x40n/0x225+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)</td>
<td>Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)</td>
<td>Sn_TX_FSR3(0x08227+0x40n/0x227+0x040n)</td>
</tr>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Ex2) Sn_TX_FSR = 33332 = 0x00008234

<table>
<thead>
<tr>
<th>Sn_TX_FSR(0x08224+0x40n/0x224+0x040n)</th>
<th>Sn_TX_FSR0(0x08224+0x40n/0x224+0x040n)</th>
<th>Sn_TX_FSR1(0x08225+0x40n/0x225+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)</td>
<td>Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)</td>
<td>Sn_TX_FSR3(0x08227+0x40n/0x227+0x040n)</td>
</tr>
<tr>
<td>0x82</td>
<td>0x34</td>
<td></td>
</tr>
</tbody>
</table>

Sn_RX_RSR (SOCKETn RX Received Size Register) [R]
[0x08228+0x40n/0x228+0x40n] [0x00000000]

It informs the byte size of received data in internal RX memory of SOCKETn. The host can’t read data through Sn_RX_FIFOR as the size bigger than Sn_RX_RSR. So, after checking Sn_RX_RSR, the host read the received data though Sn_RX_FIFOR smaller than or as same size as Sn_RX_RSR, and copies the data into the host system memory. After memory copy, the host should inform the copy completion of data to W5300 by RECV command.
Sn_RX_RSR automatically decreases by 2 bytes whenever the host reads Sn_RX_FIFOR. In case of ‘Sn_RX_RSR > 0’, there is one or more DATA packet in internal RX memory. And the received data should be processed in DATA packet unit. Refer to Sn_RX_FIFOR.

Ex1) Sn_RX_RSR = 64KB = 65536 = 0x00010000

<table>
<thead>
<tr>
<th>Sn_RX_RSR(0x08228+0x40n/0x228+0x040n)</th>
<th>Sn_RX_RSR0(0x08228+0x40n/0x21C+0x040n)</th>
<th>Sn_RX_RSR1(0x08229+0x40n/0x229+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)</td>
<td>Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)</td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

Ex2) Sn_RX_RSR = 3800 = 0x000000ED8

<table>
<thead>
<tr>
<th>Sn_RX_RSR(0x08228+0x40n/0x228+0x040n)</th>
<th>Sn_RX_RSR0(0x08228+0x40n/0x21C+0x040n)</th>
<th>Sn_RX_RSR1(0x08229+0x40n/0x229+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)</td>
<td>Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)</td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>0xD8</td>
<td></td>
</tr>
</tbody>
</table>

Sn_FragR (SOCKETn Fragment Register) [R/W] [0x0822C+0x40n/0x22C+0x040n] [0x40]

It sets the fragment field of the IP header at the IP layer. W5300 does not support the packet fragment at the IP layer. Even though Sn_FragR is configured, IP data is not fragmented. And its configuration is not recommended. It should be configured before performing OPEN command.

Ex) Sn_FragR = 0x40 (Don’t Fragment)

<table>
<thead>
<tr>
<th>Sn_FragR(0x0822C+0x40n/0x22C+0x040n)</th>
<th>Sn_FragR0(0x0822C+0x40n/0x22C+0x040n)</th>
<th>Sn_FragR1(0x0822D+0x40n/0x22D+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x40</td>
</tr>
</tbody>
</table>

Sn_TX_FIFOR (SOCKETn TX FIFO Register) [R/W] [0x0822E+0x40n/0x22E+0x040n] [0xUUUU]

It indirectly accesses internal TX memory of SOCKETn. The internal TX memory can’t be accessed directly by the host, but can be accessed through
Sn_TX_FIFOR. If MR(MT) = ‘0’, only the Host-Write of internal TX memory is allowed through Sn_TX_FIFOR. But if MR(MT) is ‘1’, both of Host-Read and Host-Write are allowed. Be sure to set it as ‘0’ after verifying interface between W5300 and the host system. (for the detail, refer to “How to Test Internal TX/RX memory”)

If the host system uses 8 bit data bus width, Sn_TX_FIFOR0 and Sn_TX_FIFOR1 should be accessed in a pair. When copying 1 byte data into internal TX memory, the host writes the 1 byte data in Sn_TX_FIFOR0 and dummy data in Sn_TX_FIFOR1. Sn_TX_FIFOR should be accessed with 2 byte size. Access the Sn_TX_FIFOR0 of low address register first, and the Sn_TX_FIFOR1 of high address register. After accessing Sn_TX_FIFOR0, it is not allowed to access other W5300 registers except for Sn_TX_FIFOR1.

When any data is written by the host through Sn_TX_FIFOR, the data is sequentially copied into internal TX memory. The data of Sn_TX_FIFOR0 and Sn_TX_FIFOR1 are respectively saved in low and high addresses of internal TX memory. The data in internal TX memory is transmitted in order of low address by SEND or SEND_MAC command.

Ex1) Sn_TX_FIFOR = 0x1122

<table>
<thead>
<tr>
<th>Sn_TX_FIFOR0(0x0822E+0x40n/0x22E+0x040n)</th>
<th>Sn_TX_FIFOR1(0x0822F+0x40n/0x22F+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x11</td>
<td>0x22</td>
</tr>
</tbody>
</table>

Ex2) When transmitting 5 Byte String Data “abcde” (abcde - 0x61 0x62 0x63 0x64 0x65)

<table>
<thead>
<tr>
<th>16 Bit Data Bus Width ( MR(DBW) = ‘1’)</th>
<th>8 Bit Data Bus Width ( MR(DBW) = ‘0’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sn_TX_FIFOR = 0x6162</td>
<td>Sn_TX_FIFOR0 = 0x61</td>
</tr>
<tr>
<td>Sn_TX_FIFOR = 0x6364</td>
<td>Sn_TX_FIFOR1 = 0x62</td>
</tr>
<tr>
<td>Sn_TX_FIFOR = 0x6500</td>
<td>Sn_TX_FIFOR0 = 0x63</td>
</tr>
<tr>
<td>Sn_TX_WRSR0 = 0x0000</td>
<td>Sn_TX_FIFOR1 = 0x64</td>
</tr>
<tr>
<td>Sn_TX_WRSR1 = 0x0005</td>
<td>Sn_TX_FIFOR0 = 0x65</td>
</tr>
<tr>
<td>Sn_CR = 0x0020 (SEND command)</td>
<td>Sn_TX_FIFOR1 = 0x00</td>
</tr>
<tr>
<td></td>
<td>Sn_TX_WRSR0 = 0x00</td>
</tr>
<tr>
<td></td>
<td>Sn_TX_WRSR1 = 0x00</td>
</tr>
<tr>
<td></td>
<td>Sn_TX_WRSR2 = 0x00</td>
</tr>
<tr>
<td></td>
<td>Sn_CR1 = 0x20 (SEND command)</td>
</tr>
</tbody>
</table>
**Sn_RX_FIFOR (SOCKETn RX FIFO Register) [R/W] [0x08230+0x40n/0x230+0x40n] [0xUUUU]**

It indirectly accesses to internal RX memory of SOCKETn. The internal RX memory can't be directly accessed by the host, but can be accessed through Sn_RX_FIFOR. If MR(MT) = '0', only the Host-Read of internal RX memory is allowed through Sn_RX_FIFOR. But if MR(MT) is '1', both of Host-Read and Host-Write are allowed. It should be set as '0' after verifying the interface between W5300 and the host system. (Refer to “How to Test Internal TX/RX memory”)

If the host system uses 8 bit data bus width, Sn_RX_FIFOR0 and Sn_RX_FIFOR1 should be accessed in a pair as like Sn_TX_FIFOR. It is not allowed to access Sn_RX_FIFOR0 and Sn_RX_FIFOR1 right after accessing Sn_TX_FIFOR0 and Sn_TX_FIFOR1. These are cause for the incorrect read. In order to prevent this, after reading Sn_TX_FIFOR0 and Sn_TX_FIFOR1, the host reads any register such as Sn_MR and then access Sn_RX_FIFOR.

When the host reads the received DATA packet in internal RX memory through Sn_RX_FIFOR by 2 bytes, the low and high data in internal RX memory can be read through Sn_RX_FIFOR0 and Sn_RX_FIFOR1 respectively. The host performs RECV command after processing the received DATA packet in internal RX memory.

According to Sn_MR(P3:P0), PACKET-INFO is added in front of all received DATA packet in internal RX memory. The added PACKET-INFO contains the packet information such as size. The host should process PACKET-INFO first and DATA packet later. If the size of received DATA packet is odd number, 1 byte dummy data is added. The host should read this dummy data first and ignore it. It is possible to check if the last byte of DATA packet is dummy or not with the size information of PACKET-INFO.

The host sequentially processes the pairs of PACKET-INFO and DATA packet in internal RX memory through Sn_RX_FIFOR.
PACKET-INFO has fixed size – 2bytes at the TCP or MACRAW mode, 8bytes at the UDP mode, 6bytes at the IPRAW mode. For the detailed information on PACKET-INFO, refer to mode description of “Chapter 5. Functional Description”

Ex1) Sn_RX_FIFOR = 0x3344

<table>
<thead>
<tr>
<th>Sn_RX_FIFOR0(0x08230+0x40n/0x230+0x040n)</th>
<th>Sn_RX_FIFOR1(0x08231+0x40n/0x231+0x040n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x33</td>
<td>0x44</td>
</tr>
</tbody>
</table>

Ex2) receiving 5Byte string data “abcde” and saving in "str" variable at the TCP mode

<table>
<thead>
<tr>
<th>16 Bit Data Bus Width ( MR(DBW) = ‘1’)</th>
<th>8 Bit Data Bus Width ( MR(DBW) = ‘0’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT16 pack_size, idx,temp</td>
<td>INT16 pack_size, idx,temp</td>
</tr>
<tr>
<td>INT8 str[5]</td>
<td>INT8 str[5], dummy</td>
</tr>
<tr>
<td>pack_size = Sn_RX_FIFOR</td>
<td>pack_size = Sn_RX_FIFOR0</td>
</tr>
<tr>
<td>idx = 0</td>
<td>pack_size = (pack_size &lt;&lt; 8)</td>
</tr>
<tr>
<td>LOOP pack_size/2</td>
<td>pack_size = pack_size + Sn_RX_FIFOR1</td>
</tr>
<tr>
<td>temp = Sn_RX_FIFOR</td>
<td>idx = 0</td>
</tr>
<tr>
<td>str[idx] = (INT8)(temp &gt;&gt; 8)</td>
<td>str[idx] = Sn_RX_FIFOR0</td>
</tr>
<tr>
<td>idx = idx + 1</td>
<td>idx = idx + 1</td>
</tr>
<tr>
<td>str[idx] = (INT8)(temp &amp; 0x00FF)</td>
<td>str[idx] = Sn_RX_FIFOR1</td>
</tr>
<tr>
<td>idx = idx + 1</td>
<td>idx = idx + 1</td>
</tr>
<tr>
<td>END LOOP</td>
<td>END LOOP</td>
</tr>
<tr>
<td>IF pack_size is odd ? THEN</td>
<td>IF pack_size is odd ? THEN</td>
</tr>
<tr>
<td>temp = Sn_RX_FIFOR</td>
<td>temp = Sn_RX_FIFOR</td>
</tr>
<tr>
<td>str[idx] = (INT8)(temp &gt;&gt; 8)</td>
<td>str[idx] = Sn_RX_FIFOR0</td>
</tr>
<tr>
<td>END IF</td>
<td>idx = idx + 1</td>
</tr>
<tr>
<td>Sn_CR = 0x0040 (RECV command)</td>
<td>str[idx] = Sn_RX_FIFOR1</td>
</tr>
<tr>
<td></td>
<td>dummy = Sn_RX_FIFOR0</td>
</tr>
<tr>
<td></td>
<td>END IF</td>
</tr>
<tr>
<td></td>
<td>Sn_CR1 = 0x040 (RECV command)</td>
</tr>
</tbody>
</table>
5. Functional Description

W5300 can provide Internet connectivity simply by setting some register. In this chapter, we can learn how to initialize W5300 and communicate according to the protocol types (TCP, UDP, IPRAW and MACRAW) by reviewing the pseudo code.

5.1 Initialization

The initialization of W5300 is processed through 3 steps: Host interface setting, network information setting, and internal TX/RX memory allocation.

- **STEP 1**: Setting host interface
  1. Setting data bus width, host interface mode & timing (Refer to MR)
  2. Setting host interrupt (Refer to IMR)

- **STEP 2**: Setting network information
  1. Setting the basic network information for data communication (Refer to SHAR, GAR, SUBR and SIPR)
  2. Setting the retransmission time-period and retry-count to be used in case of failure of packet retransmission. (Refer to RTR, RCR)

The source hardware address to be set by SHAR, is the unique hardware address of Ethernet device (Ethernet MAC address) used in Ethernet MAC layer.

The MAC address allotment is managed by IEEE. The manufacturers should assign MAC addresses acquired from IEEE to their network devices.


- **STEP 3**: Allocation internal TX/RX memory for SOCKETn
1. Defining internal TX/RX memory size (Refer to MTYPEPER)
2. Defining TX/RX memory of SOCKETn (Refer to TMSR & RMSR)

WS300 internally contains 16 memory blocks of 8Kbyte. The memory blocks are mapped in address space of 128Kbytes in sequence. 128Kbytes memory can be divided into the transmission(TX) and reception(RX) memory. The internal TX and RX memory can be allocated with 8Kbytes unit in the range of 128KBytes. Allocated internal TX/RX memory can be re-allocated to each SOCKET by 1Kbyte unit in the range of 0~64Kbytes. Below is showing that 72Kbytes is allocated to the internal TX memory and 56Kbytes is allocated to the internal RX memory. The internal TX memory is re-allocated to from SOCKET0 to SOCKET7 with the value 4, 16, 1, 20, 0, 7, 12, 12Kbytes in the range of 72Kbytes. RX memory is re-allocated with the value 17, 3, 5, 16, 3, 4, 4, 4Kbytes. Socket 4 can't transmit data because its allocated memory for TX is 0Kbyte.
Fig 8. Allocation Internal TX/RX memory of SOCKETn
5.2 Data Communication

After initialization, W5300 can transmit or receive data by opening the SOCKET as TCP, UDP, IPRAW, or MACRAW mode. W5300 supports 8 SOCKETS to be used independently and simultaneously. In this chapter, the communication method in each mode is described.

5.2.1 TCP

TCP is the connection-oriented protocol. At the TCP, a connection SOCKET is established by pairing its IP address & port number with the peer’s ones. Through this connection SOCKET, data can be transmitted and received.

There are “TCP SERVER” and “TCP CLIENT” in the method of establishing connection SOCKET. The method can be distinguished according as who transmits connect-request (SYN packet). “TCP SERVER” waits for connect-request from the peer, and establishes the connection SOCKET by accepting the request (Passive-open). “TCP CLIENT” transmits connect-request to the peer to establish the connection SOCKET (Active-open).

![Diagram of TCP SERVER and TCP CLIENT](image-url)

**Fig 9. "TCP SERVER" & "TCP CLIENT"**
5.2.1.1 TCP SERVER

Fig 10. "TCP SERVER" Operation Flow

- SOCKET Initialization
  For the TCP data communication, SOCKET initialization is required in order to open a SOCKET. To open a SOCKET, select one of 8 SOCKETs (the selected SOCKET called as SOCKETn), set the protocol mode & source port number (called as listen port number at the "TCP SERVER") to Sn_MR(P3:P0) & Sn_PORTR respectively, and perform the OPEN command. After OPEN command, if Sn_SSR is changed to SOCK_INIT then SOCKET initialization is completed.
  SOCKET initialization is identically processed both in "TCP SERVER" and "TCP CLIENT". Below is to show Initialization of SOCKETn as TCP mode.
High-performance Internet Connectivity Solution

W5300

---

If all data size received from the peer are even number, Sn_MR(ALIGN) can be set as ‘1’. In case of Sn_MR(ALIGN) = ‘1’, W5300 does not add the PACKET-INFO of TCP mode, and save only DATA packet in internal RX memory of SOCKETn. This can improve the performance by reducing the host's overhead of PACKET-INFO process. (In above code, Sn_MR = 0x0101 can be replaced with Sn_MR = 0x0001)

**LISTEN**

It operates "TCP SERVER" by performing LISTEN command.

```c
{  
    /* listen SOCKET */  
    Sn_CR = LISTEN;  
    /* wait until Sn_SSR is changed to SOCK_LISTEN */  
    if (Sn_SSR != SOCK_LISTEN) Sn_CR = CLOSE; goto START;  
}
```

**ESTABLISHED ?**

When Sn_SSR is SOCK_LISTEN, if SYN packet is received then Sn_SSR is changed to SOCK_SYNRECV. After transmitting SYN/ACK packet, the connection of SOCKETn is established.

There are two methods to check if the connection of SOCKETn is established or not. After establishing the connection of SOCKETn, data communication is available.

**First method :**

```c
{  
    if (Sn_IR(CON) == '1') Sn_IR(CON) = '1'; goto ESTABLISHED stage;  
    /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR Sn_IMR and Sn_IR. */  
}
```

**Second method :**

```c
{  
}
```
if (Sn_SSR == SOCK_ESTABLISHED) goto ESTABLISHED stage;
}

### ESTABLISHED : Received Data?

It checks if TCP data is received from the peer.

**First method:**

```c
if (Sn_IR(RECV) == '1')
    Sn_IR(RECV) = '1';
    goto Receiving Process stage;
/* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR Sn_IMR and Sn_IR. */
```

**Second Method:**

```c
if (Sn_RX_RSR != 0x00000000) goto Receiving Process stage;
```

At the first method, `Sn_IR(RECV)` is set as ‘1’ whenever receiving DATA packet. In this case, if the host could not process the `Sn_IR(RECV)` of the previously received DATA packet yet but W5300 receives the next DATA packet, the host holding the previous `Sn_IR(RECV)` could not recognize the `Sn_IR(RECV)` of the next DATA packet. Therefore if the host doesn't have enough capability to process each `Sn_IR(RECV)` of all DATA packets, this method is not recommended.

### ESTABLISHED : Receiving Process

It processes TCP data received in internal RX memory. The format of received TCP data is as below.

![Fig 11. The received TCP data format](image)

TCP data is composed of PACKET-INFO and DATA packet in case of Sn_MR(ALIGN)='0'. In case of Sn_MR(ALIGN) = '1', TCP data has only DATA packet by removing PACKET-INFO.

At the TCP mode, if the data size transmitted by the peer, is bigger than RX memory free...
size of the SOCKETn then W5300 can't receive the data, keeps the connection, and waits until RX memory free size becomes bigger than the data size.

```c
{  
    /* first, check Sn_MR(ALIGN) */  
    if (Sn_MR(ALIGN) == '0')  
    {  
        pack_size = Sn_RX_FIFOR; /* extract size of DATA packet from internal RX memory */  
    }  
    else  
    {  
        pack_size = Sn_RX_RSR; /* check the total received data size */  
    }  

    /* calculate the read count of Sn_RX_FIFOR */  
    if (pack_size is odd ?)  
    {  
        read_cnt = (pack_size + 1) / 2;  
    }  
    read_cnt = pack_size / 2;  

    /* extract DATA packet from internal RX memory */  
    for( i = 0; i < read_cnt; i++)  
    {  
        data_buf[i] = Sn_RX_FIFOR; /* data_buf is array of 16bit */  
    }  

    /* set RECV command */  
    Sn_CR = RECV;  
}
```

<Notice> In case that SOCKETn is used only to receive data without transmitting data
The slow data receiving process by the host can cause internal RX memory full.
In this case, even though W5300 window size (the maximum size of receivable data) is not ‘0’, by misunderstanding the window size as ‘0’, the peer does not transmit the data, and waits until window size is increased. It is the cause of decreasing data receiving performance of W5300. In order to solve the problem, the host processes the data received in internal RX memory first and notify the peer that the window size of W5300 is increased by received data size. To the above code, add the below code after RECV command.
/* set RECV command */
Sn_CR = RECV;

/* Add the code that notifies the update of window size to the peer */
/* check the received data process to finish or not */
if(Sn_RX_RSR == 0) /* send the window-update packet when the window size is full */
{
    /* Sn_RX_RSR can be compared with another value instead of '0',
     according to the host performance of receiving data */
    Sn_TX_WRSR = 0x00000001; /* set Dummy Data size to Sn_TX_WRSR */
    Sn_TX_FIFOR = 0x0000; /* Write Dummy Data into TX memory */
    Sn_CR = SEND; /* set SEND command */
    while(Sn_CR != 0x00); /* check SEND command completion */
    while(Sn_IR(SENDOK) == '0'); /* wait for SEND OK */
    Sn_IR(SENDOK) = '1'; /* Clear SENDOK bit */
}

<table>
<thead>
<tr>
<th>ESTABLISHED : Send DATA ? / Sending Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>It tries to transmit the data to the peer after saving the data in the internal TX memory through Sn_TX_FIFOR. TX data should not be bigger than internal TX memory allocated to SOCKETn. If TX data is bigger than MSS, it is automatically divided into MSS and transmitted.</td>
</tr>
<tr>
<td>In order to send the next data, it should be checked if previous SEND command is completed. If the next SEND command is performed before previous one is not completed, it can cause any error. The bigger data size is, the longer it takes to complete the SEND command. So, it is more effective to divide the data into appropriate size.</td>
</tr>
</tbody>
</table>

{ /* first, get the free TX memory size */

FREESIZE:
get_free_size = Sn_TX_FSR;
if (Sn_SSR != SOCK_ESTABLISHED && Sn_SSR != SOCK_CLOSE_WAIT) goto CLOSED state;
    if (get_free_size < send_size) goto FREESIZE;

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;

    /* copy data to internal TX memory */
for (i = 0; i < write_cnt; i++)
{
    Sn_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
}

/* check previous SEND command completion */
if (is first send?) /* skip check Sn_IR(SENDOK) */
else
{
    while(Sn_IR(SENDOK)=='0')
    {
        if(Sn_SSR == SOCK_CLOSED) goto CLOSED state; /* check connection establishment */
    }
    Sn_IR(SENDOK) = '1'; /* clear previous interrupt of SEND completion */
}

/* sets transmission data size to Sn_TX_WRSR */
Sn_TX_WRSR = send_size;

/* set SEND command */
Sn_CR = SEND;

ESTABLISHED : Received FIN?

It checks if disconnect-request(FIN packet) is received or not. It can be checked as below.

First method :
{
    if (Sn_IR(DISCON) == '1') Sn_IR(DISCON)='1'; goto CLOSED stage;
    /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR Sn_IMR and Sn_IR */
}

Second method :
{
    if (Sn_SSR == SOCK_CLOSE_WAIT) goto CLOSED stage;
}
ESTABLISHED : Disconnect ? / Disconnecting Process

The connection SOCKET should be disconnected when no more data communication is required, or FIN packet is received.

```c
{ /* set DISCON command */
  Sn_CR = DISCON;
}
```

ESTABLISHED : CLOSED ?

It checks if SOCKETn is disconnected or closed by DISCON or CLOSE command.

First method :
```
{  
  if (Sn_IR(DISCON) == '1') goto CLOSED stage;
  /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR Sn_IMR and Sn_IR. */
}
```

Second method :
```
{  
  if (Sn_SSR == SOCK_CLOSED) goto CLOSED stage;
}
```

ESTABLISHED : Timeout

Timeout can occur when transmitting the TCP packet such as connect-request(SYN packet) or its response packet(SYN/ACK packet), data(DATA packet) or its response packet(DATA/ACK packet), disconnect-request(FIN packet) or its response packet(FIN/ACK packet). If above packets are not transmitted during timeout value set in RTR and RCR, TCP Final Timeout(TCP_TO) occurs and Sn_SSR is changed to SOCK_CLOSED.

TCP_TO can be checked as below.

First method :
```
{  
  if (Sn_IR(TIMEOUT bit) == '1') Sn_IR(TIMEOUT)='1'; goto CLOSED stage;
  /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR Sn_IMR and Sn_IR. */
}
```

Second method :
```
{  
  if (Sn_SSR == SOCK_CLOSED) goto CLOSED stage;
}
- SOCKET Close

It is used for closing SOCKETn which is already disconnected by disconnect-process or is closed by TCP\textsubscript{Rto}. When the host wants for SOCKETn to be just closed without disconnect-process, it is also used.

```c
{
    /* clear remained interrupts */
    Sn_IR = 0x00FF;
    IR(n) = ‘1’;
    /* set CLOSE command */
    Sn_CR = CLOSE;
}
```
5.2.1.2 TCP CLIENT

Except for the CONNECT state, all states are the same as "TCP SERVER". Refer to "5.2.1.1 TCP SERVER".

Fig 12. "TCP CLIENT" Operation Flow

- CONNECT

It transmits connect-request(SYN packet) to the peer. Timeout such as ARP_TO, or TCP_TO can occur during establishing connection SOCKET with the peer.

```c
{
  Sn_DIPR = server_ip;    /* set TCP SERVER IP address*/
  Sn_DPORTR = server_port; /* set TCP SERVER listen port number*/
  Sn_CR = CONNECT;       /* set CONNECT command */
}
```
5.2.2 UDP

UDP is a connection-less protocol. UDP transmits or receives data without establishing a connection SOCKET as TCP does. TCP guarantees reliable data communications, but UDP doesn't. UDP is a datagram communication protocol. As UDP doesn't establish a connection SOCKET, it is allowed to communicate with multi-peers that already know about the source IP address and the source port number. This datagram communication has the ability to communicate with multi-peers through one SOCKET, but a possible problem is to lose data or to receive data from undesired peers. In order to prevent the problem, the host itself should re-process the lost data or ignore the received data from the undesired peer. UDP supports unicast, broadcast and multicast method; the communication flow is shown below:

![UDP Operation Flow Diagram]

Fig 13. UDP Operation Flow

5.2.2.1 Unicast & Broadcast

Unicast method is the most common UDP communication that transmits data to one peer at a time. Broadcast method is, by using broadcast IP address (255.255.255.255), transmits data to the all receivable peers at a time.

For example, when there are peers A, B, and C, Unicast transmits data to each A, B or C. At this time, ARP becomes required to acquire destination hardware address of A, B, C. It is not possible to transmit the data to the peer of ARP. Broadcast transmits data to A, B, and C simultaneously through IP address "255.255.255.255". Not like unicast, the ARP-process to acquire destination hardware address of A, B, C is not required, and ARP doesn't occur.

- SOCKET Initialization
For UDP data communication, SOCKET initialization is required. It opens a SOCKET.

For the SOCKET to open, select one of 8 SOCKETs (the selected SOCKET called as SOCKETn), set the protocol mode & source port number to Sn_MR(P3:P0) & Sn_PORTR respectively, and perform OPEN command. After OPEN command, if SOCKET status is changed to SOCK_UDP, SOCKET initialization is completed.

```
{ 
    START: 
    Sn_MR = 0x02; /* sets UDP mode */
    Sn_PORTR = source_port; /* sets source port number */
    Sn_CR = OPEN; /* sets OPEN command */
    /* wait until Sn_SSR is changed to SOCK_UDP */
    if (Sn_SSR != SOCK_UDP) Sn_CR = CLOSE; goto START;
}
```

- Received DATA?

  It checks if UDP data is received from the peer. It checks in the same way of TCP communication. The first method is not recommended. For the detail, refer to “5.2.1.1 TCP SERVER”.

  **First method:**
  ```
  if (Sn_IR(RECV) == ‘1’) Sn_IR(RECV) = ‘1’; goto Receiving Process stage;
  /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR Sn_IMR and Sn_IR. */
  ```

  **Second Method:**
  ```
  if (Sn_RX_RSR != 0x00000000) goto Receiving Process stage;
  ```

- Receiving Process

  It processes UDP data received in internal RX memory. The received UDP data format is as below.

  ![Fig 14. The received UDP data format](image)

<table>
<thead>
<tr>
<th>PACKET-INFO</th>
<th>DATA packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination IP Address</td>
<td>Destination Port number</td>
</tr>
<tr>
<td>4 Bytes</td>
<td>2 Bytes</td>
</tr>
</tbody>
</table>
UDP data is composed of 8 byte PACKET-INFO having sender's information (IP address, Port number) and DATA packet size. UDP can receive the UDP data from multi-sender. The host can know who is a sender through the destination IP address and port number of PACKET-INFO. If a sender broadcasts data using broadcast IP address "255.255.255.255", the broadcasted data can be also received. The host should ignore unnecessary DATA packet by analyzing the PACKET-INFO.

If sender's data size is bigger than RX memory free size of SOCKETn, the data can't be received. Fragmented data also can't be received.

```c
{ /* process PACKET-INFO read from internal RX memory */
    temp = Sn_RX_FIFOR; /* extract destination IP address from internal RX memory */
    dest_ip[0] = ((temp & 0xFF00) >> 8);
    dest_ip[1] = (temp & 0x00FF);
    temp = Sn_RX_FIFOR;
    dest_ip[2] = ((temp & 0xFF00) >> 8);
    dest_ip[3] = (temp & 0x00FF);
    dest_port = Sn_RX_FIFOR; /* extract destination port number from internal RX memory */
    pack_size = Sn_RX_FIFOR; /* extract length of DAT packet from internal RX memory */
    /* calculate the read count of Sn_RX_FIFOR */
    if (pack_size is odd ?) read_cnt = (pack_size + 1) / 2;
    read_cnt = pack_size / 2;
    for ( i = 0 ; i < read_cnt ; i++ )
    {
        data_buf[i] = Sn_RX_FIFOR; /* data_buf is array of 16bit */
    }
    /* set RECV command */
    Sn_CR = RECV;
}
```

- **Send Data? / Sending Process**

  It sets IP address and port number of the peer, saves the transmitting data in the internal TX memory through Sn_TX_FIFOR, and tries to transmit the data to the peer.

  Transmitting data size can't be bigger than internal TX memory of SOCKETn. If the data size is bigger than MTU, it is automatically divided into MTU unit and transmits the divided
data to the peer.

In case of broadcast, Sn_DIPR is set as "255.255.255.255".

```c
{
    /* first, get the free TX memory size */
    FREESIZE:
    get_free_size = Sn_TX_FSR;
    if (get_free_size < send_size) goto FREESIZE;

    /* Set the destination information */
    Sn_DIPR0 = dest_ip[0]; //or 255; /* Set the 4 bytes destination IP address to Sn_DIPR */
    Sn_DIPR1 = dest_ip[1]; //or 255;
    Sn_DIPR2 = dest_ip[2]; //or 255;
    Sn_DIPR3 = dest_ip[3]; //or 255;
    Sn_DPORTR = dest_port; /* Set the 2 bytes destination port number to Sn_DPORTR */

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;

    /* copy data to internal TX memory */
    for (i = 0; i < write_cnt; i++)
    {
        Sn_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
    }

    /* sets transmission data size to Sn_TX_WRSR */
    Sn_TX_WRSR = send_size;

    /* set SEND command */
    Sn_CR = SEND;
}
```

- Complete Sending? & Timeout

In order to transmit the next data, be sure to check if the previous SEND command is completed. As the bigger data size is, the longer it takes to complete the SEND command, it is more effective to divide the data into appropriate size.
When transmitting UDP data, ARP_TO can occur. In this case, UDP data transmission has failed.

```c
{ 
  /* check SEND command completion */
  while(Sn_IR(SENDOK)=='0') /* wait interrupt of SEND completion */
  { 
    /* check ARP_TO */
    if (Sn_IR(TIMEOUT)=='1') Sn_IR(TIMEOUT)='1'; goto Next stage;
  }
  Sn_IR(SENDOK) = '1'; /* clear previous interrupt of SEND completion */
}
```

- Finished? / SOCKET Close

If there is any more communication, SOCKETn is closed.

```c
{ 
  /* clear remained interrupts */
  Sn_IR = 0x00FF;
  IR(n) = '1';
  /* set CLOSE command */
  Sn_CR = CLOSE;
}
```

### 5.2.2.2 Multicast

Broadcast method communicates with undefined multi-peers, but multicast method communicates with defined multi-peers who are registered as a member for multicast-group.

For example, A, B, and C are registered as a member of multicast-group. If A transmits data to the multicast-group, B & C can receive the data. For multicast communication, register as a member of multicast-group by using IGMP protocol. All multicast-groups are distinguished by group hardware address, group IP address and group port number.

Group hardware address and IP address use already assigned addresses, but group port number can be used any.

As for group hardware address, it is selectable in the range from “01:00:5e:00:00:00” to “01:00:5e:7f:ff:ff”. As for group IP address, it's in the range of D-class IP address (“224.0.0.0” ~ “239.255.255.255”). At this time, the lower 23 bit of group hardware address (6bytes) and IP address (4bytes) should be same. For example, if the group IP address is set as “224.1.1.11”, the group hardware address should be set as “01:00:5e:01:01:0b”.

© Copyright 2008-2014 WIZnet Co., Ltd.  All rights reserved.
Refer to "RFC1112"(http://www.ietf.org/rfc.html).

In the W5300, the IGMP required for registering multicast-group is automatically processed. When opening SOCKETn as multicast mode, "Join" message of IGMP is automatically transmitted. When closing the SOCKET, "Leave" message is transmitted. After opening SOCKET, "Report" message is automatically & periodically transmitted. W5300 supports IGMP version 1 & 2. If upper version needs to be used, the host should manually process IGMP protocol using IPRAW mode SOCKET.

- SOCKET Initialization

For the multicast communication, select one of 8 SOCKETs(the selected SOCKET called as SOCKETn), and set Sn_DHAR as multicast-group hardware address and Sn_DIPR as multicast-group IP address. Sn_PORTR and Sn_DPORTR are set as multicast-group port number. After setting Sn_MR(P3:P0) as UDP and Sn_MR(MULTI) as '1', perform OPEN command. After OPEN command, when SOCKET status is changed to SOCK_UDP, SOCKET initialization is completed.

```c
{ 
    START:
    /* set Multicast-Group information */
    Sn_DHAR0 = 0x01;  /* set Multicast-Group H/W address(01:00:5e:01:01:0b) */
    Sn_DHAR1 = 0x00;
    Sn_DHAR2 = 0x5E;
    Sn_DHAR3 = 0x01;
    Sn_DHAR4 = 0x01;
    Sn_DHAR5 = 0x0B;
    Sn_DIPR0 = 211;  /* set Multicast-Group IP address(211.1.1.11) */
    Sn_DIPR1 = 1;
    Sn_DIPR2 = 1;
    Sn_DIPR3 = 11;
    Sn_DPORTR = 0x0BB8;  /* set Multicast-Group Port number(3000) */
    Sn_PORTR = 0x0BB8;  /* set Source Port number(3000) */
    Sn_MR = 0x0002 | 0x0080;  /* set UDP mode & Multicast on SOCKETn Mode Register */

    Sn_CR = OPEN;  /* set OPEN command */

    /* wait until Sn_SSR is changed to SOCK_UDP */
    if (Sn_SSR != SOCK_UDP) Sn_CR = CLOSE; goto START;
}
```
Received DATA?

Receiving Process
Refer to “5.2.2.1 Unicast & Broadcast”.

Send Data? / Sending Process
As multicast-group information is already set at the SOCKET Initialization, it is not necessary to set the destination IP address and port number as like unicast. Therefore, just copy transmitting data into the internal TX memory, and perform SEND command.

{  
    /* first, get the free TX memory size */
    FREESIZE:
        get_free_size = Sn_TX_FSR;
        if (get_free_size < send_size) goto FREESIZE;

    /* calculate the write count of Sn_TX_FIFO */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;

    /* copy data to internal TX memory */
    for (i = 0; i < write_cnt; i++)
    {
        Sn_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
    }

    /* sets transmission data size to Sn_TX_WRSR */
    Sn_TX_WRSR = send_size;

    /* set SEND command */
    Sn_CR = SEND;
}

Complete Sending? & Timeout
As it is communication with previously defined multicast-group, ARP-process is not required. ARP_TO doesn’t occur.

{  
    /* check SEND command completion */
    while(Sn_IR(SENDOK)=='0'); /* wait interrupt of SEND completion */
}
5.2.3 IPRAW

IPRAW is the data communication to use an IP layer lower than TCP and UDP. IPRAW supports IP layer protocol such as ICMP(0x01) or IGMP(0x02) that can be defined according to protocol number. The ping of ICMP or V1/v2 of IGMP is internally designed with hardware logic. However, the host can manually implement them by opening SOCKETn as IPRAW mode.

In case of using IPRAW mode SOCKET, the protocol should be defined in the protocol number field of IP header. Protocol number is defined by IANA (Refer to http://www.iana.org/assignments/protocol-numbers). Protocol number should be set before the SOCKET is opened.

TCP(0x06) or UDP (0x11) protocol number is not supported. The communication of IPRAW mode SOCKET just allows the protocol number which is set in Sn_PROTOR. For example, the SOCKET set Sn_PROTOR as ICMP can't receive any other protocol data whose protocol number is not ICMP.

![Fig 15. IPRAW Operation Flow](image_url)

- SOCKET Initialization
  It selects a SOCKET and sets protocol number. Set the SN_MR(P3:P0) as IPRAW mode,
and perform OPEN command. After OPEN command, when SOCKET status is changed to SOCK_IPRAW, the SOCKET initialization is completed.

```
{  
  START:  
/* sets Protocol number */  
/* The protocol number is used in Protocol Field of IP Header. */  
SnPROTO = protocol_num;  
/* sets IP raw mode */  
SnMR = 0x03;  
/* sets OPEN command */  
SnCR = OPEN;  
/* wait until Sn_SSR is changed to SOCK_IPRAW */  
if (Sn_SSR != SOCK_IPRAW) SnCR = CLOSE; goto START; 
}
```

- Received DATA?
  Refer to “5.2.2.1 Unicast & Broadcast”.

- Receiving Process
  It processes IPRAW data received in the internal RX memory. The received IPRAW data format is as below.

  Fig 16. The received IPRAW data format

  IPRAW data is composed of 6 byte PACKET-INFO and DATA packet. PACKET-INFO includes sender’s information (IP address) and the length of DATA packet. Data receiving process at the IPRAW mode is same as UDP except for processing the port number of PACKET-INFO.

  Refer to “5.2.2.1 Unicast & Broadcast”.

  If the sender’s data size is bigger than RX memory free size of SOCKETn, the data can’t be received. The fragmented data also can’t be received.

- Send DATA? / Sending Process
  Transmitting data can’t be bigger than internal TX memory of a SOCKETn, and default MTU.
Data transmission process at the IPRAW mode is same as UDP, except for configuring the destination port number.

Refer to “5.2.2.1 Unicast & Broadcast”.

- Complete Sending & Timeout
- Finished? / SOCKET Closed

It is same as UDP communication. Refer to “5.2.2 UDP”.

5.2.4 MACRAW

MACRAW is the communication based on Ethernet MAC lower than IP layer. MACRAW mode communication uses SOCKET0 only. Even if SOCKET0 is used as MACRAW, SOCKET1 ~ 7 also can be used with hardwired TCP/IP stack simultaneously. In this case, SOCKET0 operates as NIC (Network Interface Controller) and software TCP/IP stack can be implemented through this.

This is the hybrid TCP/IP stack of W5300 – supporting hardwired TCP/IP & software TCP/IP. By using the hybrid TCP/IP feature, it is possible to overcome the SOCKET limitation of W5300. If high-performing data transmission is required, it can be implemented by using hardwired TCP/IP SOCKET. For the normal data transmission, the software TCP/IP can be used by using MACRAW mode. The SOCKET0 of MACRAW mode can process all protocols except for the protocol used in SOCKET1~ 7. As MACRAW is the communication method to process pure Ethernet packets, the engineer should have knowledge about the software TCP/IP stack.

As MACRAW data is based on Ethernet MAC, it should have 6bytes source hardware address & destination hardware address and 2bytes Ethernet type.

Fig 17. MACRAW Operation Flow
 SOCKET Initialization

It selects a SOCKET and sets Sn_MR(P3:P0) as MACRAW mode, and perform OPEN command.

After OPEN command, when SOCKET status is changed to SOCK_MACRAW, SOCKET initialization is completed. As all the information for the communication (Source hardware address, source IP address, source port number, destination hardware address, destination IP address, destination port number, all type of protocol header, etc) is included in MACRAW data, the related register setting is not required.

```
{  
    START:  
        /* sets MAC raw mode */  
        S0_MR = 0x04;  
        /* sets OPEN command */  
        S0_CR = OPEN;  
        /* wait until Sn_SSR is changed to SOCK_MACRAW */  
        if (Sn_SSR != SOCK_MACRAW) S0_CR = CLOSE; goto START;  
}
```

 Received DATA?

Refer to “5.2.2.1 Unicast & Broadcast”.

 Receiving Process

It processes MACRAW data received in internal RX memory of SOCKET0. The received MACRAW data format is as below.

<table>
<thead>
<tr>
<th>PACKET-INFO</th>
<th>DATA packet</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte size of DATA packet</td>
<td>Real Data</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>2 Bytes</td>
<td>Destination MAC (6Bytes)</td>
<td>Source MAC (6Bytes)</td>
</tr>
</tbody>
</table>

Fig 18. The received MACRAW data format

MACRAW data is composed of 2 bytes PACKET-INFO, DATA packet and 4s byte CRC. PACKET-INFO includes the size of DATA packet, and DATA packet does 6bytes destination MAC address, 6bytes source MAC address, 2bytes type and 46 ~1500 bytes payload. The payload of DATA packet has internet protocol such as ARP or IP. For the detail of Type, refer to [http://www.iana.org/assignments/ethernet-numbers](http://www.iana.org/assignments/ethernet-numbers).
The CRC of MACRAW data should be read by the host through S0_RX_FIFOR and ignored.

```c
{
    /* extract size of DATA packet from internal RX memory */
    pack_size = S0_RX_FIFOR;

    /* calculate the read count of Sn_RX_FIFOR */
    if (pack_size % 2 == 0) read_cnt = pack_size / 2;
    else read_cnt = (pack_size + 1) / 2;

    /* extract DATA packet from internal RX memory */
    for( i = 0; i < read_cnt; i++)
    {
        data_buf[i] = S0_RX_FIFOR; /* data_buf is array of 16bit */
    }

    /* extract 4 bytes CRC from internal RX memory and then ignore it */
    dummy = S0_RX_FIFOR;
    dummy = S0_RX_FIFOR;

    /* set RECV command */
    S0_CR = RECV;
}
```

**Notice**

In case that free buffer size of internal RX memory is smaller than the size of receiving MAC RAW data, some parts of un-acceptable PACKET-INFO and DATA packet of the MACRAW data can be saved in internal RX memory. This can cause the error in analyzing PACKET-INFO (as shown in above code), and receiving correct MACRAW data. This problem is more likely to happen when internal RX memory gets close full. This can be solved by ignoring some loss of MACRAW data.

- By performing internal RX memory process as quick as possible, prevent the memory to be full
- By receiving only its own MACRAW data, reduce the receiving burden.

Set the MF bit of S0_MR in the sample code showing SOCKET initialization.

```c
{
    START:
    /* sets MAC raw mode with enabling MAC filter */
```
S0_MR = 0x44;
/* sets OPEN command */
S0_CR = OPEN;
/* wait until Sn_SSR is changed to SOCK_MACRAW */
if (Sn_SSR != SOCK_MACRAW) S0_CR = CLOSE; goto START;
}

- In case that the free size of internal RX memory is smaller than 1528 - Default MTU(1514)+PACKET-INFO(2)+DATA packet(8)+CRC(4) - close SOCKET 0. After closing the SOCKET0, Process all received MACRAW data and the reopen the SOCKET0.

```
{  
    /* check the free size of internal RX memory */
    if((RMSR0 * 1024) - Sn_RX_RSR < 1528)
    {
        recved_size = Sn_RX_RSR; /* backup Sn_RX_RSR */
        Sn_CR = CLOSE;        /* SOCKET0 Closed */
        while(Sn_SSR != SOCK_CLOSED); /* wait until SOCKET0 is closed */
        /* process all data remained in internal RX memory */
        while(recved_size > 0)
        {
            /* extract size of DATA packet from internal RX memory */
            pack_size = S0_RX_FIFOR;
            /* calculate the read count of Sn_RX_FIFOR */
            if (pack_size is odd ?) read_cnt = (pack_size + 1) / 2;
            read_cnt = pack_size / 2;
            /* extract DATA packet from internal RX memory */
            for( i = 0; i < read_cnt; i++)
            {
                data_buf[i] = S0_RX_FIFOR; /* data_buf is array of 16bit */
            }
            /* extract 4 bytes CRC from internal RX memory and then ignore it */
            dummy = S0_RX_FIFOR;
            dummy = S0_RX_FIFOR;
            /* calculate the size of remained data in internal RX memory*/
            if(pack_size & 0x01)    // if pack_size is odd,
                recved_size = recved_size - 2 - (pack_size +1) - 4;
        }
    }
```
else  // if pack_size is even.
    recved_size = recved_size - 2 - pack_size - 4;

/* Reopen the SOCKET0 */
/* sets MAC raw mode with enabling MAC filter */
S0_MR = 0x44; /* or S0_MR = 0x04 */
/* sets OPEN command */
S0_CR = OPEN;
/* wait until Sn_SSR is changed to SOCK_MACRAW */
while (Sn_SSR != SOCK_MACRAW);
}
else /* process normally the DATA packet from internal RX memory */
{
    /* This block is same as the code of “Receiving process” stage*/
}

Send DATA? / Sending Process

The transmitted data can't be bigger than internal TX memory of SOCKET0 and default MTU. The host creates the MACRAW data in the same format of DATA packet mentioned above "Receiving Process". If the host data which size is under 60bytes, the internal "zero padding" is processed for the real transmitting Ethernet packet to become 60 bytes.

{
    /* first, get the free TX memory size */

FREESIZE:
    get_free_size = S0_TX_FSR;
    if (get_free_size < send_size) goto FREESIZE;

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;

    /* copy data to internal TX memory */
    for (i = 0; i < write_cnt; i++)
    {
        S0_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
    }
/* sets transmission data size to Sn_TX_WRSR */
S0_TX_WRSR = send_size;

/* set SEND command */
S0_CR = SEND;

Complete Sending?
All the protocol for the data communication is processed by the host, thus timeout does not occur.

{ /* check SEND command completion */
while(S0_IR(SENDOK)=='0'); /* wait interrupt of SEND completion */
S0_IR(SENDOK) = '1'; /* clear previous interrupt of SEND completion */
}

Finished? / SOCKET Close
Refer to “5.2.2.1 Unicast & Broadcast”.
6. External Interface

The host interface of W5300 is decided by the direct/indirect address mode and 16/8 bit data. Also, W5300 can be interfaced with internal PHY or external PHY according to the configuration of TEST_MODE[3:0].

6.1 Direct Address Mode

6.1.1 16 Bit Data Bus Width

In case of using a 16bit data bus width, ADDR[9:1] is used and ADDR0 is connected to ground or floated. 'BIT16EN' is internally pulled-up, so it is no problem if it is allowed to float.

6.1.2 8 Bit Data Bus Width

In the case of using an 8bit data bus width, ADDR[9:0] is used. 'BIT16EN' should be logical LOW (ground). Let the unused DATA[15:8] float.
6.2 Indirect Address Mode

6.2.1 16 Bit Data Bus Width
In case of using a 16bit data bus width, only ADDR[2:1] is used, and ADDR[9:3] should be connected to ground, and ADDR0 are connected to ground or floated. As ‘BIT16EN” is internally pulled-up, it can be floated.

6.2.2 8 Bit Data Bus Width
In case of using an 8bit data bus width, only ADDR[2:0] is used, and ADDR[9:3] should be connected to ground. ‘BIT16EN’ should be connected to ground. Let the unused DATA[15:8] float.
6.3 Internal PHY Mode

When using internal PHY of W5300, TEST_MODE[3:0] is connected to ground or floated. According to internal PHY operation mode, OP_MODE[2:0] is configured. For the detail refer to “1.1 Configuration Signals”.

For better impedance-matching between internal PHY and transformer, a termination resistor and a capacitor are required – 50ohm(±1%) resistor & 0.1uF capacitor.

The internal PHY supports 6 network indicator LEDs including LINK and SPEED. Float the unused LED signals. By tying /RXLED and /TXLED with logical AND, an ACT LED(Active LED) can be implemented. For the detail, refer to “1.6 Network Indicator LED Signals”.

![Fig 19. Internal PHY & LED Signals](image-url)
6.4 External PHY Mode

If the internal PHY does not satisfy the user's requirements, an external PHY made by 3rd party can be interfaced. In case of using external PHY mode, W5300 clock source should be selected. When TEST_MODE0 is logically high, a crystal is used, and when TEST_MODE1 is logically high, an oscillator is used.

For the detail refer to "1.1 Configuration Signals" and "1.7 Clock Signals".

For the impedance matching between external PHY and transformer, refer to the document from the PHY manufacturer.

W5300's '/FDX' Pin is connected to duplex indicator signal of the external PHY.

![external PHY interface with MII diagram](image-url)
### 7. Electrical Specifications

#### Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_DD</td>
<td>DC supply voltage</td>
<td>-0.5 to 3.6</td>
<td>V</td>
</tr>
<tr>
<td>V_IN</td>
<td>DC input voltage</td>
<td>-0.5 to 5.5 (5V tolerant)</td>
<td>V</td>
</tr>
<tr>
<td>V_OUT</td>
<td>DC output voltage</td>
<td>-0.5 to 3.6</td>
<td>V</td>
</tr>
<tr>
<td>I_IN</td>
<td>DC input current</td>
<td>±5</td>
<td>mA</td>
</tr>
<tr>
<td>I_OUT</td>
<td>DC output current</td>
<td>2 to 8</td>
<td>mA</td>
</tr>
<tr>
<td>T_OP</td>
<td>Operating temperature</td>
<td>-40 to 85 [1]</td>
<td>°C</td>
</tr>
<tr>
<td>T_STG</td>
<td>Storage temperature</td>
<td>-55 to 125</td>
<td>°C</td>
</tr>
</tbody>
</table>

*COMMENT: Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage.


#### DC Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Condition</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_DD</td>
<td>DC Supply voltage</td>
<td>Junction temperature is from -55°C to 125°C</td>
<td>3.0</td>
<td>3.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>V_IN</td>
<td>High level input voltage</td>
<td></td>
<td>2.0</td>
<td>5.5</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>V_IL</td>
<td>Low level input voltage</td>
<td></td>
<td>- 0.5</td>
<td>0.8</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>V_OH</td>
<td>High level output voltage</td>
<td>I_OH = 2 ~ 16 mA</td>
<td>2.4</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>V_OL</td>
<td>Low level output voltage</td>
<td>I_OL = -2 ~ -12 mA</td>
<td>0.4</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>I_I</td>
<td>Input Current</td>
<td>V_IN = V_DD</td>
<td>±5</td>
<td>μA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I_O</td>
<td>Output Current</td>
<td>V_OUT = V_DD</td>
<td>2</td>
<td>8</td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

#### POWER DISSIPATION

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Condition</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_IA</td>
<td>Power consumption when using the auto-negotiation</td>
<td>Vcc 3.3V, Temperature 25°C</td>
<td>-</td>
<td>180</td>
<td>250</td>
<td>mA</td>
</tr>
</tbody>
</table>
### AC Characteristics

#### Reset Timing

<table>
<thead>
<tr>
<th>Description</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Reset Cycle Time</td>
<td>2 us</td>
<td>-</td>
</tr>
<tr>
<td>2 PLL Lock-in Time</td>
<td>50 us</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

**Register READ Timing**

- **ADDR[9:0]**
  - tADDRs
  - tADDRh

- **/CS**
  - tCS
  - tCSn

- **/RD**
  - tRD
  - tDATAs
  - tDATAh

- **DATA[15:0]**
  - Valid Data
### High-performance Internet Connectivity Solution

#### W5300

<table>
<thead>
<tr>
<th>Description</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tADDRs</code></td>
<td>-</td>
<td>7 ns</td>
</tr>
<tr>
<td><code>tADDRh</code></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><code>tCS</code></td>
<td>65 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tCSn</code></td>
<td>28 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tRD</code></td>
<td>65 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tDATAs</code></td>
<td>-</td>
<td>42 ns</td>
</tr>
<tr>
<td><code>tDATAh</code></td>
<td>-</td>
<td>7 ns</td>
</tr>
<tr>
<td><code>tDATAhe</code></td>
<td>-</td>
<td>2XPLL_CLK</td>
</tr>
</tbody>
</table>

*Note* `tDATAhe` is the data holding time when MR(RDH) is ‘1’. During this time, data bus is driven during 2XPLL_CLK after /CS is de-asserted high. So, be careful of data bus collision.

#### Register WRITE Timing

![Register WRITE Timing Diagram](image)

<table>
<thead>
<tr>
<th>Description</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tADDRs</code></td>
<td>-</td>
<td>7 ns</td>
</tr>
<tr>
<td><code>tADDRh</code></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><code>tCS</code></td>
<td>50 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tCSn</code></td>
<td>28 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tWR</code></td>
<td>50 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tDATAs</code></td>
<td>7 ns</td>
<td>-</td>
</tr>
<tr>
<td><code>tDATAh</code></td>
<td>7 ns</td>
<td>-</td>
</tr>
</tbody>
</table>

*Note* `tDATAs` is holding time of Host-Write data Fetch during 7 PLL_CLK according to the
setting value of MR(WDF2-WDF0).

As ‘tDATAf’ is the time to fetch the Host-Write data, if /WR is de-asserted High before this time, the Host-Write data is fetched at the time of /WR High-De-assert regardless of ‘tDATAf’.

In order to fetch the valid data at this time, the host should guarantee ‘tDATAh’.

Crystal Characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>25 MHz</td>
</tr>
<tr>
<td>Frequency Tolerance (at 25°C)</td>
<td>±50 ppm</td>
</tr>
<tr>
<td>Shunt Capacitance</td>
<td>7pF Max</td>
</tr>
<tr>
<td>Drive Level</td>
<td>1 ~ 500uW (100uW typical)</td>
</tr>
<tr>
<td>Load Capacitance</td>
<td>27pF</td>
</tr>
<tr>
<td>Aging (at 25°C)</td>
<td>±5ppm/year Max</td>
</tr>
</tbody>
</table>

Transformer Characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Transmit End</th>
<th>Receive End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn Ratio</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>Inductance</td>
<td>350 uH</td>
<td>350 uH</td>
</tr>
</tbody>
</table>

In case of using internal PHY mode, be sure to use symmetric transformer in order to support Auto MDI/MDIX(Crossover).

In case of using External PHY mode, use the transform which is suitable for external PHY specification.
8. IR Reflow Temperature Profile (Lead-Free)

Moisture Sensitivity Level : 3
Dry Pack Required : Yes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Ramp-Up Rate</td>
<td>3° C/second max.</td>
</tr>
<tr>
<td>(Ts(_{\text{max}}) to Tp)</td>
<td></td>
</tr>
<tr>
<td>Preheat</td>
<td></td>
</tr>
<tr>
<td>– Temperature Min (Ts(_{\text{min}}))</td>
<td>150 °C</td>
</tr>
<tr>
<td>– Temperature Max (Ts(_{\text{max}}))</td>
<td>200 °C</td>
</tr>
<tr>
<td>– Time (ts(<em>{\text{min}}) to ts(</em>{\text{max}}))</td>
<td>60-180 seconds</td>
</tr>
<tr>
<td>Time maintained above:</td>
<td></td>
</tr>
<tr>
<td>– Temperature (TL)</td>
<td>217 °C</td>
</tr>
<tr>
<td>– Time (tL)</td>
<td>60-150 seconds</td>
</tr>
<tr>
<td>Peak/Classification Temperature (Tp)</td>
<td>260 + 0 °C</td>
</tr>
<tr>
<td>Time within 5 °C of actual Peak Temperature (tp)</td>
<td>20-40 seconds</td>
</tr>
<tr>
<td>Ramp-Down Rate</td>
<td>6 °C/second max.</td>
</tr>
<tr>
<td>Time 25 °C to Peak Temperature</td>
<td>8 minutes max.</td>
</tr>
</tbody>
</table>
9. Package Descriptions
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MILLIMETER</th>
<th>INCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN.</td>
<td>NOM.</td>
</tr>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A₁</td>
<td>0.05</td>
<td>-</td>
</tr>
<tr>
<td>A₂</td>
<td>1.35</td>
<td>1.40</td>
</tr>
<tr>
<td>b</td>
<td>0.17</td>
<td>0.22</td>
</tr>
<tr>
<td>b₁</td>
<td>0.17</td>
<td>0.20</td>
</tr>
<tr>
<td>c</td>
<td>0.09</td>
<td>-</td>
</tr>
<tr>
<td>c₁</td>
<td>0.09</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>15.85</td>
<td>16.00</td>
</tr>
<tr>
<td>D₁</td>
<td>13.90</td>
<td>14.00</td>
</tr>
<tr>
<td>E</td>
<td>15.85</td>
<td>16.00</td>
</tr>
<tr>
<td>E₁</td>
<td>13.90</td>
<td>14.00</td>
</tr>
<tr>
<td>e</td>
<td>0.50 BSC</td>
<td>0.020 BSC</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
</tr>
<tr>
<td>L₁</td>
<td>1.00 REF</td>
<td>0.039 REF</td>
</tr>
<tr>
<td>R₁</td>
<td>0.08</td>
<td>-</td>
</tr>
<tr>
<td>R₂</td>
<td>0.08</td>
<td>-</td>
</tr>
<tr>
<td>S</td>
<td>0.20</td>
<td>-</td>
</tr>
<tr>
<td>θ</td>
<td>0°</td>
<td>3.5°</td>
</tr>
<tr>
<td>θ₁</td>
<td>0°</td>
<td>-</td>
</tr>
<tr>
<td>θ₂</td>
<td>12° TYP</td>
<td>12° TYP</td>
</tr>
<tr>
<td>θ₃</td>
<td>12° TYP</td>
<td>12° TYP</td>
</tr>
</tbody>
</table>

**<NOTE>**

- 1 To be determined at seating plane C.
- 2 Dimensions ‘D₁’ and ‘E₁’ do not include mold protrusion.
- 3 ‘D₁’ and ‘E₁’ are maximum plastic body size dimensions including mold mismatch.
- 4 Dimension ‘b’ does not include dambar protrusion.
- 5 Dambar can not be located on the lower radius or the foot.
- 6 Exact shape of each corner is optional
- 7 These Dimensions apply to the flat section of the lead between 0.10mm and 0.25mm from the lead tip.
- 8 A₁ is defined as the distance from the seating plane to the lowest point of the package body.
- 7 Controlling dimension : Millimeter
- 8 Reference Document : JEDEC MS-026 , BED.