THE DATASHEET OF

W5300

High-Performance Internet Connectivity Solution

W5300

Version 1.2.5

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

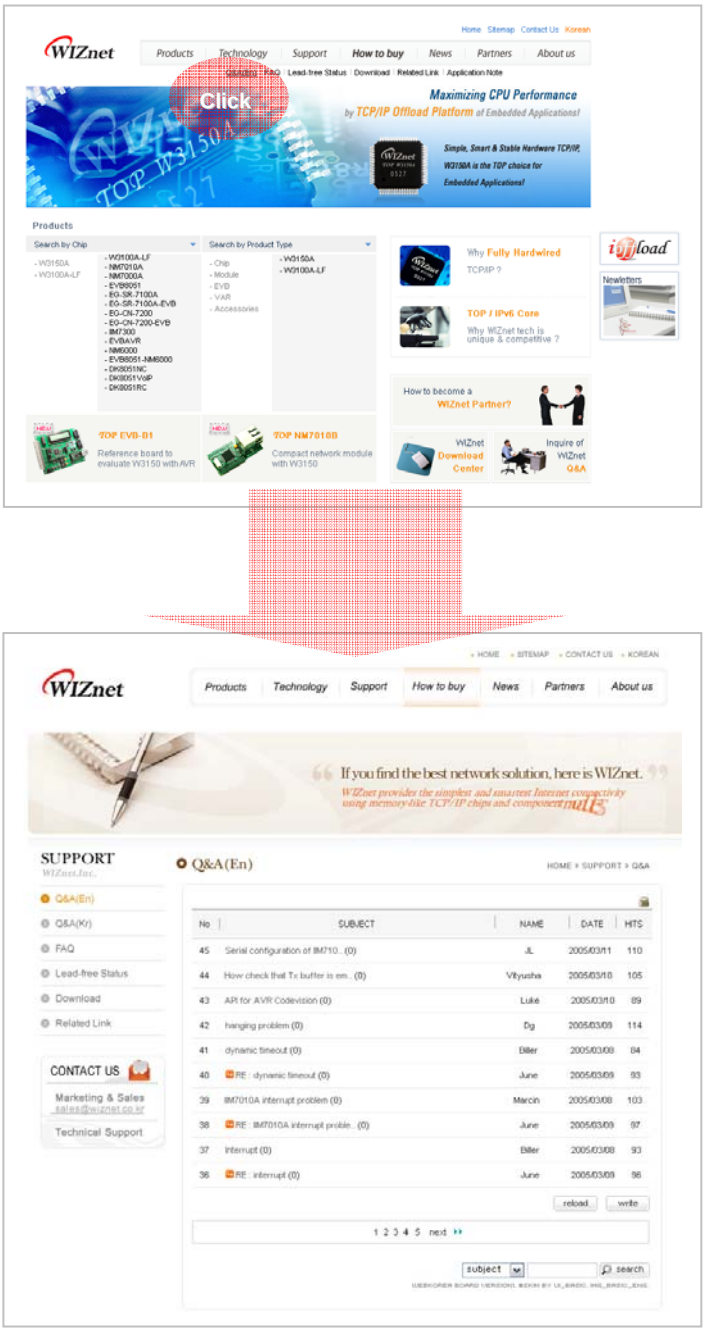
Document History Information

Version	Date	Descriptions
Ver. 1.0.0	Mar. 11, 2008	Release with W5300 launching
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.
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 schemati
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.19) --1V8O: **1.8V regulator output voltage** Capacitor value: 0.1uF -> 10Uf - Modified Fig.2 Power Design (p.21)

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

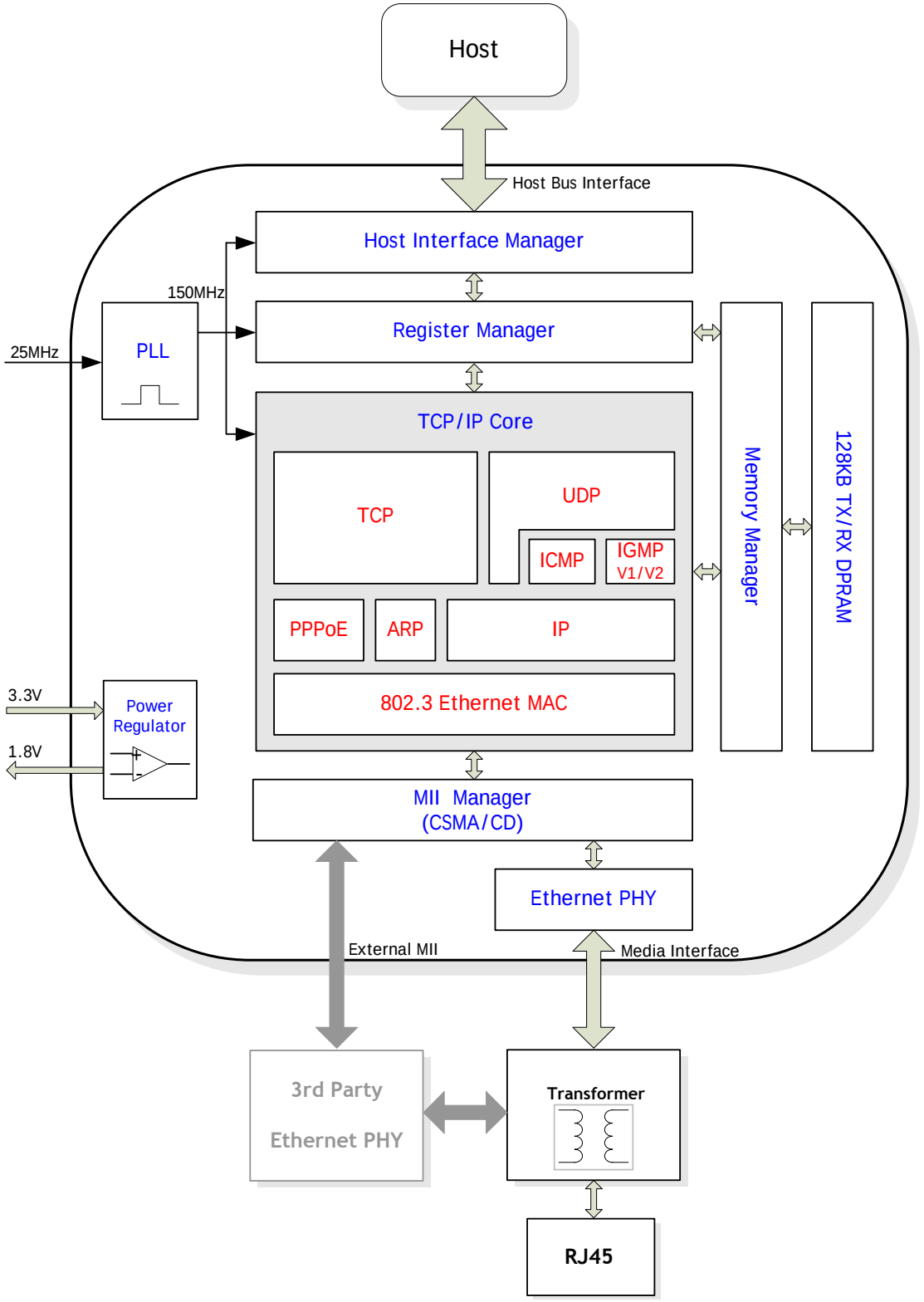
The W5300 is well-suited for many embedded applications, including:

- 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



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.

- **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

List of Figures

1. PIN Description

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

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

OM : Multi-functional Output PIN

1.1 PIN Layout

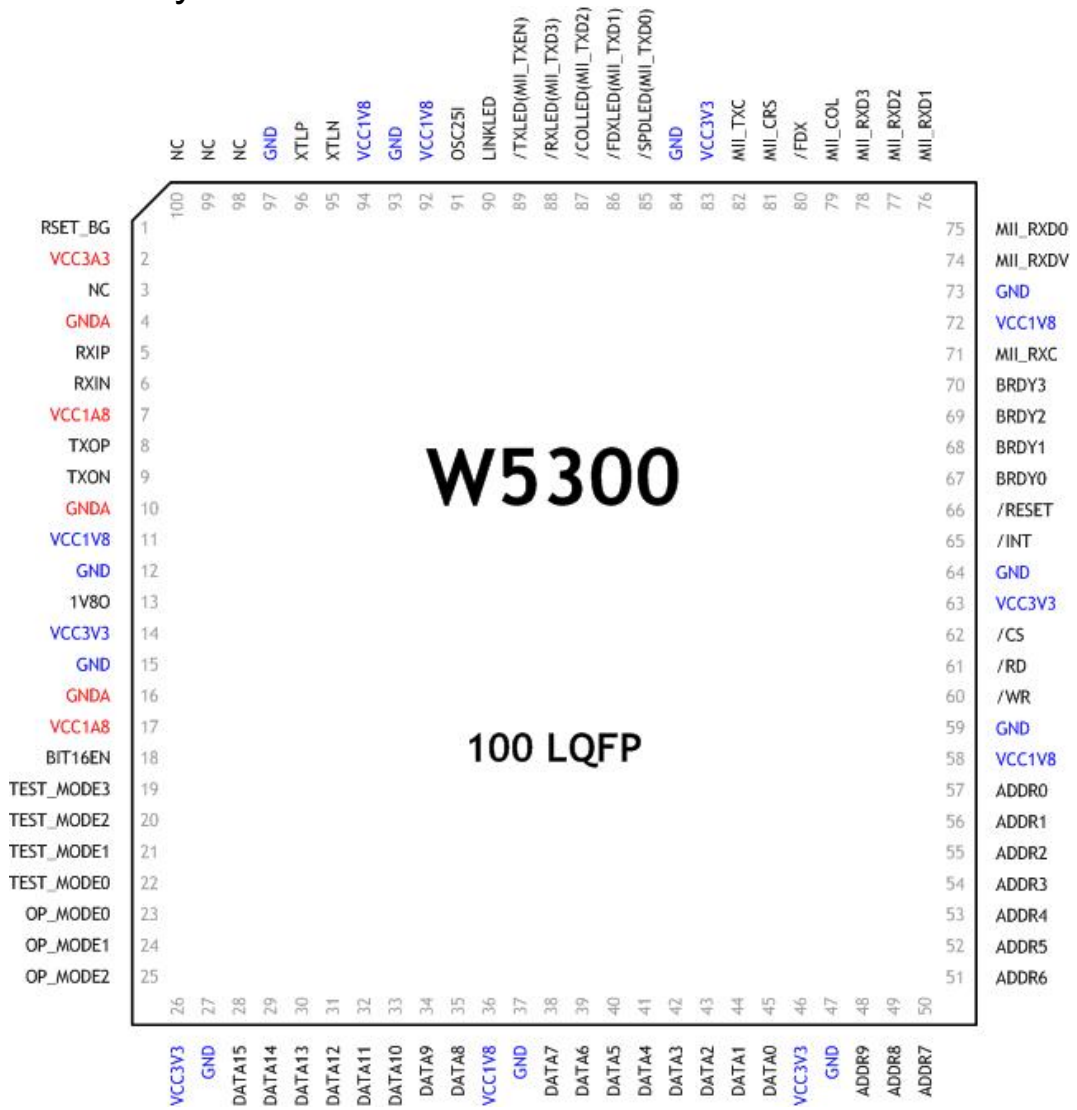


Fig 1. PIN Layout

1.2 Configuration Signals

<table>
<tr><th>Symbol</th><th>Type</th><th>Description</th></tr>
<tr><td>TEST_MODE[3:0]</td><td>ID</td><td>W5300 mode select

It configures PHY mode and factory test mode of W5300.

<table>
<tr><th colspan="4">TEST_MODE</th><th rowspan="2">Description</th></tr>
<tr><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Internal PHY Mode (Normal Operation)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>External PHY Mode with Crystal clock</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>External PHY Mode with Oscillator clock</td></tr>
<tr><td colspan="4">Others</td><td>Reserved (Factory Test Mode)</td></tr>
</table>

At the external PHY mode, Clock input pin is changed by clock source. Refer to “1.7 Clock Signals”.</td></tr>
<tr><td>OP_MODE[2:0]</td><td>ID</td><td>Internal PHY operation control mode

It configures the operation mode of internal PHY.
<table>
<tr><th colspan="3">OP_MODE</th><th rowspan="2">Description</th></tr>
<tr><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Normal Operation Mode, Recommended
Auto-negotiation enable with all capabilities</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Auto-negotiation with 100 BASE-TX FDX/HDX ability</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Auto-negotiation with 10 BASE-T FDX/HDX ability</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Manual selection of 100 BASE-TX FDX</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Manual selection of 100 BASE-TX HDX</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Manual selection of 10 BASE-T FDX</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Manual selection of 10 BASE-T HDX</td></tr>
</table>
cf> FDX : Full-duplex, HDX : Half-duplex

The setting value is latched after hardware reset.</td></tr>
</table>

1.3 Host Interface Signals

Symbol	Type	Description
/RESET	IL	**RESET** Hardware Reset Signal. 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. Refer to RESET timing of “7 Electrical Specification” W5300 does not support Power-On-Reset. Therefore, it should be manually designed in the target system.
BIT16EN	IU	**16/8 BIT DATA BUS SELECT** High : 16 bit data bus Low : 8 bit data bus It determinates data bus width of W5300. At reset time, it is latched in 15th Bit(‘BW’)of Mode register(MR). 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.
ADDR9-0	I	**ADDRESS** System address bus. These are selected by host interface mode and data bus width of W5300. When using 16 bit data bus, ADDR0 is internally ignored. Refer to “6.External Interface”.
DATA[15:8]	IO	**DATA** System high data bus. These are used for read/write operation of W5300 register. In case of using 8 bit data bus, These are driven as High-Z.
DATA[7:0]	IO8	**DATA** System low data bus. These are used for read/write operation of W5300 register.
/CS	IL	**CHIP SELECT** Chip select signal.

		Host selects W5300 at the W5300 read/write operation. When /CS is de-asserted high, DATA[15:0] are driven as High-Z.
/WR	IL	**WRITE ENABLE** Write enable signal. Host writes W5300 register addressed by ADDR[9:0] to DATA[15:0]. DATA[15:0] are latched in the W5300 register according to the configuration of the Write-data-fetch-timing. Refer to 13-11th bit(WDF[2:0] of MR).
/RD	IL	**READ ENABLE** Read enable signal. Host reads W5300 register addressed by ADDR[9:0] through DATA[15:0].
/INT	OL	**INTERRUPT** 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).
BRDY[3:0]	O	**Buffer Ready Indicator** 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 & Pn_DPTHR in “4.3 COMMON Registers”.

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".

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

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 pail (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.

Symbol	Type	Description
/TXLED(**MII_TXEN**)	OMH	Transmit Act LED / **Transmit Enable** 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].
/RXLED(**MII_TXD3**)	OM	/RXLED,/COLLED,/LEDFDX,/SPDLED / **Transmit data output** The transmit packet is synchronized with MII_TXC clock and output to external PHY in nibble unit. MII_TXD3 is the Most Significant Bit (MSB).
/COLLED(**MII_TXD2**)		
/FDXLED(**MII_TXD1**)		
/SPDLED(**MII_TXD0**)		
MII_TXC	ID	**Transmit Clock Input** 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.
MII_CRS	IDH	**Carrier Sense** It is signal to notify the link traffic of the media. If carrier of media is not idle (carrier present), it is asserted high.
MII_COL	IDH	**Collision Detect** 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.

MII_RXD3 MII_RXD2 MII_RXD1 MII_RXD0	ID	**Receive Data Input** When MII_RXDV is high, the received packet is synchronized with MII_RXC and inputs in nibble unit. MII_RXD3 is MSB.
MII_RXDV	ID	**Receive Data Valid** 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.
MII_RXC	ID	**Receive Clock Input** 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.
/FDX	IDL	**Full-Duplex Select** 0 : Full-duplex 1 : Half-duplex 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.

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.

Symbol	Type	Description
LINKLED	OL	**Link LED** It indicates the link status of media(10/100M).
/TXLED(MII_TXEN)	OML	**Transmit activity LED**/Transmit Enable It notifies the output of transmit data through TXOP/TXON (Transmit Activity).
/RXLED(MII_TXD3)	OML	**Receive activity LED**/Transmit Data It notifies the input of receive data from RXIP/RXIN (Receive Activity) cf> By binding /TXLED and /RXLED signals with 'AND' gate, it can be used for network activity LED.
/COLLED(MII_TXD2)	OML	**Collision LED**/Transmit Data It notifies when collisions occur. It is valid at half-duplex, and is ignored at full-duplex.
/FDXLED(MII_TXD1)	OML	**Full duplex LED**/Transmit Data 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].
/SPDLED(MII_TXD0)	OML	**Link speed LED**/Transmit Data It is asserted low at the 100Mbps and high at the 10Mbps according to auto-negotiation or manual configuration of OP_MODE[2:0].

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.

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

1.8 Power Supply Signals

Symbol	Type	Description
VCC3A3	Power	**3.3V power supply for Analog part** Be sure to connect 10uF tantalum capacitor between VCC343 and GNDA in order to prevent power compensation.
VCC3V3	Power	**3.3V power supply for Digital part** 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.
VCC1A8	Power	**1.8V power supply for Analog part** Be sure to connect a 10uF tantalum capacitor and 0.1uF capacitor between VCC1A8 and GNDA for core power noise filtering.
VCC1V8	Power	**1.8V power supply for Digital part** Between each VCC and GND, 0.1uF decoupling capacitor can be selectively connected.
GNDA	Ground	**Analog ground** Make analogue ground plane as wide as possible when designing the PCB layout.
GND	Ground	**Digital ground** Make digital ground plane as wide as possible when designing the

		PCB layout.
1V8O	O	**1.8V regulator output voltage** 1.8V/150mA power created by internal power regulator, is used for core operation power (VCC1A8, VCC1V8). 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. <Notice> 1V8O is the power for W5300 core operation. It should not be connected to the power of other devices.

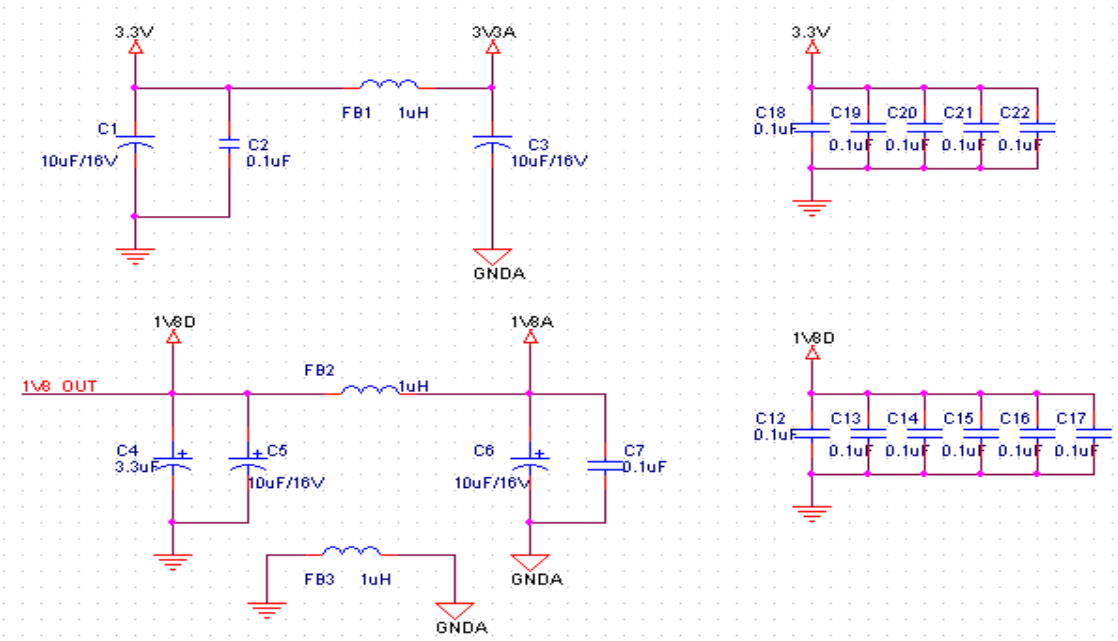


Fig 2. Power Design

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 2bytes 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 2Bytes 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 (0^{th} 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:

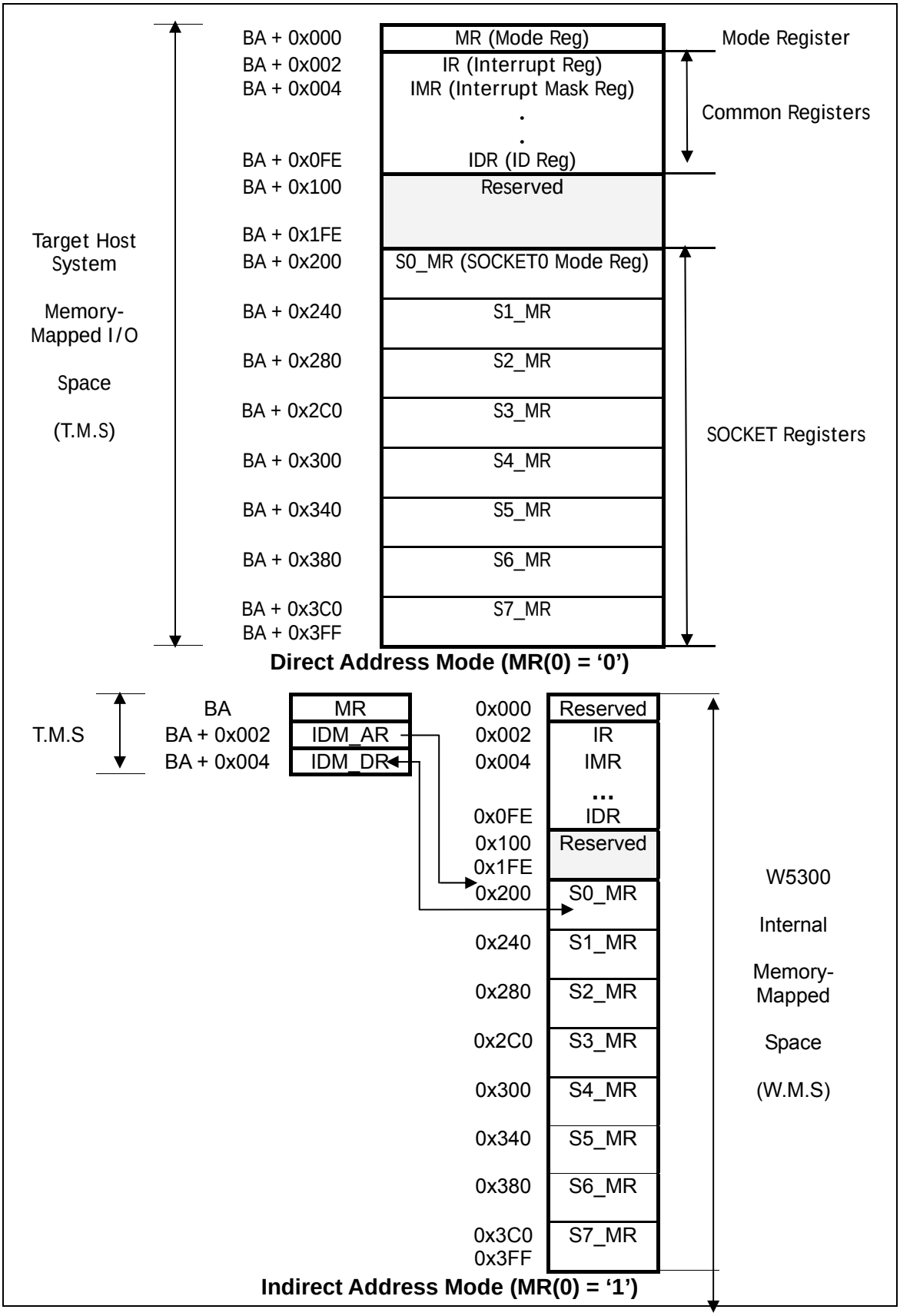


Fig 3. Memory Map

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.

Physical Address of W5300 Reg **= Base Address** of T.M.S **+ 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]

MR : MR register
MR0 : Low address register of MR (Address offset - 0x000), Most significant byte
MR1 : High address register of MR (Address offset – 0x001), Least significant byte
MR(15:5) : 11 bit (from 15th bit to 5th bit of MR register)
MR(0) : 0th bit of MR register, 0th bit of MR1
MR(13) : 13th bit of MR register, 5th bit of MR0
MR0(7) : 15th bit of MR register, Most significant bit of MR0
MR(DWB) : MR의 DWB bit (DWB : Bit Symbol)

SHAR : Source Hardware Address Register
SHAR0 : 1ST address register of SHAR (Address offset – 0x008)
SHAR1 : 2nd address register of SHAR (Address offset – 0x009)
SHAR2 : 3rd address register of SHAR (Address offset – 0x00A)
SHAR3 : 4th address register of SHAR (Address offset – 0x00B)
SHAR4 : 5th Address register of SHAR (Address offset – 0x00C)
SHAR5 : 6th address register of SHAR (Address offset – 0x00D)

3.1 Mode Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x000	0x000	MR	MR0	Mode Register
	0x001		MR1	

3.2 Indirect Mode Registers

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x002	0x002	IDM_AR	IDM_AR0	Indirect Mode Address Register
	0x003		IDM_AR1	
0x004	0x004	IDM_DR	IDM_DR0	Indirect Mode Data Register
	0x005		IDM_DR1	

3.3 COMMON registers

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x002	0x002	IR	IR0	Interrupt Register
	0x003		IR1	
0x004	0x004	IMR	IMR0	Interrupt Mask Register
	0x005		IRM1	
0x006	0x006			Reserved
	0x007			
0x008	0x008	SHAR	SHAR0	Source Hardware Address Register
	0x009		SHAR1	
0x00A	0x00A	SHAR2	SHAR2	
	0x00B		SHAR3	
0x00C	0x00C	SHAR4	SHAR4	
	0x00D		SHAR5	
0x00E	0x00E			Reserved
	0x00F			
0x010	0x010	GAR	GAR0	Gateway Address Register
	0x011		GAR1	
0x12	0x012	GAR2	GAR2	
	0x013		GAR3	

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x014	0x014	SUBR	SUBR0	Subnet Mask Register
	0x015		SUBR1	
0x016	0x016	SUBR2	SUBR2	
	0x017		SUBR3	
0x018	0x018	SIPR	SIPR0	Source IP Address Register
	0x019		SIPR1	
0x01A	0x01A	SIPR2	SIPR2	
	0x01B		SIPR3	
0x01C	0x01C	RTR	RTR0	Retransmission Timeout-value Register
	0x01D		RTR1	
0x01E	0x01E	RCR	RCR0	Reserved
	0x01F		RCR1	Retransmission Retry-count Register
0x020	0x020	TMS01R	TMSR0	Transmit Memory Size Register of SOCKET0
	0x021		TMSR1	Transmit Memory Size Register of SOCKET1
0x022	0x022	TMS23R	TMSR2	Transmit Memory Size Register of SOCKET2
	0x023		TMSR3	Transmit Memory Size Register of SOCKET3
0x24	0x024	TMS45R	TMSR4	Transmit Memory Size Register of SOCKET4
	0x025		TMSR5	Transmit Memory Size Register of SOCKET5
0x26	0x026	TMS67R	TMSR6	Transmit Memory Size Register of SOCKET7
	0x027		TMSR7	Transmit Memory Size Register of SOCKET 8
0x028	0x028	RMS01R	RMSR0	Receive Memory Size Register of SOCKET0
	0x029		RMSR1	Receive Memory Size Register of SOCKET1
0x02A	0x02A	RMS23R	RMSR2	Receive Memory Size Register of SOCKET2
	0x02B		RMSR3	Receive Memory Size Register of SOCKET3
0x02C	0x02C	RMS45R	RMSR4	Receive Memory Size Register of SOCKET4
	0x02D		RMSR5	Receive Memory Size Register of SOCKET5
0x02E	0x02E	RMS67R	RMSR6	Receive Memory Size Register of SOCKET6
	0x02F		RMSR7	Receive Memory Size Register of SOCKET7
0x030	0x030	MTYPER	MTYPER0	Memory Block Type Register
	0x031		MTYPER1	
0x032	0x032	PATR	PATR0	PPPoE Authentication Register
	0x033		PATR1	

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x034	0x034			Reserved
	0x035			
0x036	0x036	PTIMER	PTIMER0	Reserved
	0x037		PTIMER1	PPP LCP Request Time Register
0x038	0x038	PMAGICR	PMAGICR0	
	0x039		PMAGICR1	PPP LCP Magic Number Register
0x03A	0x03A			Reserved
	0x03B			
0x03C	0x03C	PSIDR	PSIDR0	PPP Session ID Register
	0x03D		PSIDR1	
0x03E	0x03E			Reserved
	0x03F			
0x040	0x040	PDHAR	PDHAR0	PPP Destination Hardware Address Register
	0x041		PDHAR1	
0x042	0x042	PDHAR2	PDHAR2	
	0x043		PDHAR3	
0x044	0x044	PDHAR4	PDHAR4	
	0x045		PDHAR5	
0x046	0x046			Reserved
	0x047			
0x048	0x048	UIPR	UIPR0	Unreachable IP Address Register
	0x049		UIPR1	
0x04A	0x04A	UIPR2	UIPR2	
	0x04B		UIPR3	
0x04C	0x04C	UPORTR	UPORT0	Unreachable Port Number Register
	0x04D		UPORT1	
0x04E	0x04E	FMTUR	FMTUR0	Fragment MTU Register
	0x04F		FMTUR1	
0x050	0x050			Reserved
	0x051			
:				
:				
0x5E	0x05E			Reserved
	0x060			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x060	0x060	P0_BRDYR	P0_BRDYR0	Reserved
	0x061		P0_BRDYR1	PIN "BRDY0" Configure Register
0x062	0x062	P0_BDPTHR	P0_BDPTHR0	PIN "BRDY0" Buffer Depth Register
	0x063		P0_BDPTHR1	
0x064	0x064	P1_BRDYR	P1_BRDYR0	Reserved
	0x065		P1_BRDYR1	PIN "BRDY1" Configure Register
0x066	0x066	P1_BDPTHR	P1_BDPTHR0	PIN "BRDY1" Buffer Depth Register
	0x067		P1_BDPTHR1	
0x068	0x068	P2_BRDYR	P1_BRDYR0	Reserved
	0x069		P2_BRDYR1	PIN "BRDY2" Configure Register
0x06A	0x06A	P2_BDPTHR	P2_BDPTHR0	PIN "BRDY2" Buffer Depth Register
	0x06B		P2_BDPTHR1	
0x06C	0x06C	P3_BRDYR	P3_BRDYR0	Reserved
	0x06D		P3_BRDYR1	PIN "BRDY3" Configure Register
0x06E	0x06E	P3_BDPTHR	P3_BDPTHR0	PIN "BRDY3" Buffer Depth Register
	0x06F		P3_BDPTHR1	
0x070	0x070			Reserved
	0x071			
:				
:				
0xFC	0x0FC			Reserved
	0x0FD			
0xFE	0x0FE	IDR	IDR0	W5300 ID Register
	0x0FF		IDR1	

3.4 SOCKET registers

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x200	0x200	S0_MR	S0_MR0	SOCKET0 Mode Register
	0x201		S0_MR1	
0x202	0x202	S0_CR	S0_CR0	Reserved
	0x203		S0_CR1	SOCKET0 Command Register
0x204	0x204	S0_IMR	S0_IMR0	Reserved
	0x205		S0_IMR1	SOCKET0 Interrupt Mask Register
0x206	0x206	S0_IR	S0_IR0	Reserved
	0x207		S0_IR1	SOCKET0 Interrupt Register
0x208	0x208	S0_SSR	S0_SSR0	Reserved
	0x209		S0_SSR1	SOCKET0 SOCKET Status Register
0x20A	0x20A	S0_PORTR	S0_PORTR0	SOCKET0 Source Port Register
	0x20B		S0_PORTR1	
0x20C	0x20C	S0_DHAR	S0_DHAR0	SOCKET0 Destination Hardware Address Register
	0x20D		S0_DHAR1	
0x20E	0x20E	S0_DHAR2	S0_DHAR2	
	0x20F		S0_DHAR3	
0x210	0x210	S0_DHAR4	S0_DHAR4	
	0x211		S0_DHAR5	
0x212	0x212	S0_DPORTR	S0_DPORTR0	SOCKET0 Destination Port Register
	0x213		S0_DPORTR1	
0x214	0x214	S0_DIPR	S0_DIPR0	SOCKET0 Destination IP Address Register
	0x215		S0_DIPR1	
0x216	0x216	S0_DIPR2	S0_DIPR2	
	0x217		S0_DIPR3	
0x218	0x218	S0_MSSR	S0_MSSR0	SOCKET0 Maximum Segment Size Register
	0x219		S0_MSSR1	
0x21A	0x21A	S0_PORTOR	S0_KPALVTR	SOCKET0 Keep Alive Time Register
	0x21B		S0_PROTOR	SOCKET0 Protocol Number Register
0x21C	0x21C	S0_TOSR	S0_TOSR0	Reserved
	0x21D		S0_TOSR1	SOCKET0 TOS Register
0x21E	0x21E	S0_TTLR	S0_TTLR0	Reserved
	0x21F		S0_TTLR1	SOCKET0 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x220	0x220	S0_TX_WRSR	S0_TX_WRSR0	Reserved
	0x221		S0_TX_WRSR1	SOCKET0 TX Write Size Register
0x222	0x222	S0_TX_WRSR2	S0_TX_WRSR2	
	0x223		S0_TX_WRSR3	
0x224	0x224	S0_TX_FSR	S0_TX_FSR0	Reserved
	0x225		S0_TX_FSR1	SOCKET0 TX Free Size Register
0x226	0x226	S0_TX_FSR2	S0_TX_FSR2	
	0x227		S0_TX_FSR3	
0x228	0x228	S0_RX_RSR	S0_RX_RSR0	Reserved
	0x229		S0_RX_RSR1	SOCKET0 RX Receive Size Register
0x22A	0x22A	S0_RX_RSR2	S0_RX_RSR2	
	0x22B		S0_RX_RSR3	
0x22C	0x22C	S0_FRAGR	S0_FRAGR0	Reserved
	0x22D		S0_FRAGR1	SOCKET0 FLAG Register
0x22E	0x22E	S0_TX_FIFOR	S0_TX_FIFOR0	SOCKET0 TX FIFO Register
	0x22F		S0_TX_FIFOR1	
0x230	0x230	S0_RX_FIFOR	S0_RX_FIFOR0	SOCKET0 RX FIFO Register
	0x231		S0_RX_FIFOR1	
0x232	0x232			Reserved
	0x233			
:				:
:				:
0x23E	0x23E			Reserved
	0x23F			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x240	0x240	S1_MR	S1_MR0	SOCKET1 Mode Register
	0x241		S1_MR1	
0x242	0x242	S1_CR	S1_CR0	Reserved
	0x243		S1_CR1	SOCKET1 Command Register
0x244	0x244	S1_IMR	S1_IMR0	Reserved
	0x245		S1_IMR1	SOCKET1 Interrupt Mask Register
0x246	0x246	S1_IR	S1_IR0	Reserved
	0x247		S1_IR1	SOCKET1 Interrupt Register
0x248	0x248	S1_SSR	S1_SSR0	Reserved
	0x249		S1_SSR1	SOCKET1 SOCKET Status Register
0x24A	0x24A	S1_PORTR	S1_PORTR0	SOCKET1 Source Port Register
	0x24B		S1_PORTR1	
0x24C	0x24C	S1_DHAR	S1_DHAR0	SOCKET1 Destination Hardware Address Register
	0x24D		S1_DHAR1	
0x24E	0x24E	S1_DHAR2	S1_DHAR2	
	0x24F		S1_DHAR3	
0x250	0x250	S1_DHAR4	S1_DHAR4	
	0x251		S1_DHAR5	
0x252	0x252	S1_DPORTR	S1_DPORTR0	SOCKET1 Destination Port Register
	0x253		S1_DPORTR1	
0x254	0x254	S1_DIPR	S1_DIPR0	SOCKET1 Destination IP Address Register
	0x255		S1_DIPR1	
0x256	0x256	S1_DIPR2	S1_DIPR2	
	0x257		S1_DIPR3	
0x258	0x258	S1_MSSR	S1_MSSR0	SOCKET1 Maximum Segment Size Register
	0x259		S1_MSSR1	
0x25A	0x25A	S1_PORTOR	S1_KPALVTR	SOCKET1 Keep Alive Time Register
	0x25B		S1_PROTOR	SOCKET1 Protocol Number Register
0x25C	0x25C	S1_TOSR	S1_TOSR0	Reserved
	0x25D		S1_TOSR1	SOCKET1 TOS Register
0x25E	0x25E	S1_TTLR	S1_TTLR0	Reserved
	0x25F		S1_TTLR1	SOCKET1 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x260	0x260	S1_TX_WRSR	S1_TX_WRSR0	Reserved
	0x261		S1_TX_WRSR1	SOCKET1 TX Write Size Register
0x262	0x262	S1_TX_WRSR2	S1_TX_WRSR2	
	0x263		S1_TX_WRSR3	
0x264	0x264	S1_TX_FSR	S1_TX_FSR0	Reserved
	0x265		S1_TX_FSR1	SOCKET1 TX Free Size Register
0x266	0x266	S1_TX_FSR2	S1_TX_FSR2	
	0x267		S1_TX_FSR3	
0x268	0x268	S1_RX_RSR	S1_RX_RSR0	Reserved
	0x269		S1_RX_RSR1	SOCKET1 RX Receive Size Register
0x26A	0x26A	S1_RX_RSR2	S1_RX_RSR2	
	0x26B		S1_RX_RSR3	
0x26C	0x26C	S1_FRAGR	S1_FRAGR0	Reserved
	0x26D		S1_FRAGR1	SOCKET1 IP FLAG Field Register
0x26E	0x26E	S1_TX_FIFOR	S1_TX_FIFOR0	SOCKET1 TX FIFO Register
	0x26F		S1_TX_FIFOR1	
0x270	0x270	S1_RX_FIFOR	S1_RX_FIFOR0	SOCKET1 RX FIFO Register
	0x271		S1_RX_FIFOR1	
0x272	0x272			Reserved
	0x273			
:				:
:				:
0x27E	0x27E			Reserved
	0x27F			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x280	0x280	S2_MR	S2_MR0	SOCKET2 Mode Register
	0x281		S2_MR1	
0x282	0x282	S2_CR	S2_CR0	Reserved
	0x283		S2_CR1	SOCKET2 Command Register
0x284	0x284	S2_IMR	S2_IMR0	Reserved
	0x285		S2_IMR1	SOCKET2 Interrupt Mask Register
0x286	0x286	S2_IR	S2_IR0	Reserved
	0x287		S2_IR1	SOCKET2 Interrupt Register
0x288	0x288	S2_SSR	S2_SSR0	Reserved
	0x289		S2_SSR1	SOCKET2 SOCKET Status Register
0x28A	0x28A	S2_PORTR	S2_PORTR0	SOCKET2 Source Port Register
	0x28B		S2_PORTR1	
0x28C	0x28C	S2_DHAR	S2_DHAR0	SOCKET2 Destination Hardware Address Register
	0x28D		S2_DHAR1	
0x28E	0x28E	S2_DHAR2	S2_DHAR2	
	0x28F		S2_DHAR3	
0x290	0x290	S2_DHAR4	S2_DHAR4	
	0x291		S2_DHAR5	
0x292	0x292	S2_DPORTR	S2_DPORTR0	SOCKET2 Destination Port Register
	0x293		S2_DPORTR1	
0x294	0x294	S2_DIPR	S2_DIPR0	SOCKET2 Destination IP Address Register
	0x295		S2_DIPR1	
0x296	0x296	S2_DIPR2	S2_DIPR2	
	0x297		S2_DIPR3	
0x298	0x298	S2_MSSR	S2_MSSR0	SOCKET2 Maximum Segment Size Register
	0x299		S2_MSSR1	
0x29A	0x29A	S2_PORTOR	S2_KPALVTR	SOCKET2 Keep Alive Time Register
	0x29B		S2_PROTOR	SOCKET2 Protocol Number Register
0x29C	0x29C	S2_TOSR	S2_TOSR0	Reserved
	0x29D		S2_TOSR1	SOCKET2 TOS Register
0x29E	0x29E	S2_TTLR	S2_TTLR0	Reserved
	0x29F		S2_TTLR1	SOCKET2 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x2A0	0x2A0	S2_TX_WRSR	S2_TX_WRSR0	Reserved
	0x2A1		S2_TX_WRSR1	SOCKET2 TX Write Size Register
0x2A2	0x2A2	S2_TX_WRSR2	S2_TX_WRSR2	
	0x2A3		S2_TX_WRSR3	
0x2A4	0x2A4	S2_TX_FSR	S2_TX_FSR0	Reserved
	0x2A5		S2_TX_FSR1	SOCKET2 TX Free Size Register
0x2A6	0x2A6	S2_TX_FSR2	S2_TX_FSR2	
	0x2A7		S2_TX_FSR3	
0x2A8	0x2A8	S2_RX_RSR	S2_RX_RSR0	Reserved
	0x2A9		S2_RX_RSR1	SOCKET2 RX Receive Size Register
0x2AA	0x2AA	S2_RX_RSR2	S2_RX_RSR2	
	0x2AB		S2_RX_RSR3	
0x2AC	0x2AC	S2_FRAGR	S2_FRAGR0	Reserved
	0x2AD		S2_FRAGR1	SOCKET2 IP FLAG Field Register
0x2AE	0x2AE	S2_TX_FIFOR	S2_TX_FIFOR0	SOCKET2 TX FIFO Register
	0x2AF		S2_TX_FIFOR1	
0x2B0	0x2B0	S2_RX_FIFOR	S2_RX_FIFOR0	SOCKET2 RX FIFO Register
	0x2B1		S2_RX_FIFOR1	
0x2B2	0x2B2			Reserved
	0x2B3			
:				:
:				:
0x2BE	0x2BE			Reserved
	0x2BF			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x2C0	0x2C0	S3_MR	S3_MR0	SOCKET3 Mode Register
	0x2C1		S3_MR1	
0x2C2	0x2C2	S3_CR	S3_CR0	Reserved
	0x2C3		S3_CR1	SOCKET3 Command Register
0x2C4	0x2C4	S3_IMR	S3_IMR0	Reserved
	0x2C5		S3_IMR1	SOCKET3 Interrupt Mask Register
0x2C6	0x2C6	S3_IR	S3_IR0	Reserved
	0x2C7		S3_IR1	SOCKET3 Interrupt Register
0x2C8	0x2C8	S3_SSR	S3_SSR0	Reserved
	0x2C9		S3_SSR1	SOCKET3 SOCKET Status Register
0x2CA	0x2CA	S3_PORTR	S3_PORTR0	SOCKET3 Source Port Register
	0x2CB		S3_PORTR1	
0x2CC	0x2CC	S3_DHAR	S3_DHAR0	SOCKET3 Destination Hardware Address Register
	0x2CD		S3_DHAR1	
0x2CE	0x2CE	S3_DHAR2	S3_DHAR2	
	0x2CF		S3_DHAR3	
0x2D0	0x2D0	S3_DHAR4	S3_DHAR4	
	0x2D1		S3_DHAR5	
0x2D2	0x2D2	S3_DPORTR	S3_DPORTR0	SOCKET3 Destination Port Register
	0x2D3		S3_DPORTR1	
0x2D4	0x2D4	S3_DIPR	S3_DIPR0	SOCKET3 Destination IP Address Register
	0x2D5		S3_DIPR1	
0x2D6	0x2D6	S3_DIPR2	S3_DIPR2	
	0x2D7		S3_DIPR3	
0x2D8	0x2D8	S3_MSSR	S3_MSSR0	SOCKET3 Maximum Segment Size Register
	0x2D9		S3_MSSR1	
0x2DA	0x2DA	S3_PORTOR	S3_KPALVTR	SOCKET3 Keep Alive Time Register
	0x2DB		S3_PROTOR	SOCKET3 Protocol Number Register
0x2DC	0x2DC	S3_TOSR	S3_TOSR0	Reserved
	0x2DD		S3_TOSR1	SOCKET3 TOS Register
0x2DE	0x2DE	S3_TTLR	S3_TTLR0	Reserved
	0x2DF		S3_TTLR1	SOCKET3 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x2E0	0x2E0	S3_TX_WRSR	S3_TX_WRSR0	Reserved
	0x2E1		S3_TX_WRSR1	SOCKET3 TX Write Size Register
0x2E2	0x2E2	S3_TX_WRSR2	S3_TX_WRSR2	
	0x2E3		S3_TX_WRSR3	
0x2E4	0x2E4	S3_TX_FSR	S3_TX_FSR0	Reserved
	0x2E5		S3_TX_FSR1	SOCKET3 TX Free Size Register
0x2E6	0x2E6	S3_TX_FSR2	S3_TX_FSR2	
	0x2E7		S3_TX_FSR3	
0x2E8	0x2E8	S3_RX_RSR	S3_RX_RSR0	Reserved
	0x2E9		S3_RX_RSR1	SOCKET3 RX Receive Size Register
0x2EA	0x2EA	S3_RX_RSR2	S3_RX_RSR2	
	0x2EB		S3_RX_RSR3	
0x2EC	0x2EC	S3_FRAGR	S3_FRAGR0	Reserved
	0x2ED		S3_FRAGR1	SOCKET3 IP FLAG Field Register
0x2EE	0x2EE	S3_TX_FIFOR	S3_TX_FIFOR0	SOCKET3 TX FIFO Register
	0x2EF		S3_TX_FIFOR1	
0x2F0	0x2F0	S3_RX_FIFOR	S3_RX_FIFOR0	SOCKET3 RX FIFO Register
	0x2F1		S3_RX_FIFOR1	
0x2F2	0x2F2			Reserved
	0x2F3			
:				:
:				:
0x2FE	0x2FE			Reserved
	0x2FF			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x300	0x300	S4_MR	S4_MR0	SOCKET4 Mode Register
	0x301		S4_MR1	
0x302	0x302	S4_CR	S4_CR0	Reserved
	0x303		S4_CR1	SOCKET4 Command Register
0x304	0x304	S4_IMR	S4_IMR0	Reserved
	0x305		S4_IMR1	SOCKET4 Interrupt Mask Register
0x306	0x306	S4_IR	S4_IR0	Reserved
	0x307		S4_IR1	SOCKET4 Interrupt Register
0x308	0x308	S4_SSR	S4_SSR0	Reserved
	0x309		S4_SSR1	SOCKET4 SOCKET Status Register
0x30A	0x30A	S4_PORTR	S4_PORTR0	SOCKET4 Source Port Register
	0x30B		S4_PORTR1	
0x30C	0x30C	S4_DHAR	S4_DHAR0	SOCKET4 Destination Hardware Address Register
	0x30D		S4_DHAR1	
0x30E	0x30E	S4_DHAR2	S4_DHAR2	
	0x30F		S4_DHAR3	
0x310	0x310	S4_DHAR4	S4_DHAR4	
	0x311		S4_DHAR5	
0x312	0x312	S4_DPORTR	S4_DPORTR0	SOCKET4 Destination Port Register
	0x313		S4_DPORTR1	
0x314	0x314	S4_DIPR	S4_DIPR0	SOCKET4 Destination IP Address Register
	0x315		S4_DIPR1	
0x316	0x316	S4_DIPR2	S4_DIPR2	
	0x317		S4_DIPR3	
0x318	0x318	S4_MSSR	S4_MSSR0	SOCKET4 Maximum Segment Size Register
	0x319		S4_MSSR1	
0x31A	0x31A	S4_PORTOR	S4_KPALVTR	SOCKET4 Keep Alive Time Register
	0x31B		S4_PROTOR	SOCKET4 Protocol Number Register
0x31C	0x31C	S4_TOSR	S4_TOSR0	Reserved
	0x31D		S4_TOSR1	SOCKET4 TOS Register
0x31E	0x31E	S4_TTLR	S4_TTLR0	Reserved
	0x31F		S4_TTLR1	SOCKET4 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x320	0x320	S4_TX_WRSR	S4_TX_WRSR0	Reserved
	0x321		S4_TX_WRSR1	SOCKET4 TX Write Size Register
0x322	0x322	S4_TX_WRSR2	S4_TX_WRSR2	
	0x323		S4_TX_WRSR3	
0x324	0x324	S4_TX_FSR	S4_TX_FSR0	Reserved
	0x325		S4_TX_FSR1	SOCKET4 TX Free Size Register
0x326	0x326	S4_TX_FSR2	S4_TX_FSR2	
	0x327		S4_TX_FSR3	
0x328	0x328	S4_RX_RSR	S4_RX_RSR0	Reserved
	0x329		S4_RX_RSR1	SOCKET4 RX Receive Size Register
0x32A	0x32A	S4_RX_RSR2	S4_RX_RSR2	
	0x32B		S4_RX_RSR3	
0x32C	0x32C	S4_FRAGR	S4_FRAGR0	Reserved
	0x32D		S4_FRAGR1	SOCKET4 IP FLAG Field Register
0x32E	0x32E	S4_TX_FIFOR	S4_TX_FIFOR0	SOCKET4 TX FIFO Register
	0x32F		S4_TX_FIFOR1	
0x330	0x330	S4_RX_FIFOR	S4_RX_FIFOR0	SOCKET4 RX FIFO Register
	0x331		S4_RX_FIFOR1	
0x332	0x332			Reserved
	0x333			
:				:
:				:
0x33E	0x33E			Reserved
	0x33F			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x340	0x340	S5_MR	S5_MR0	SOCKET5 Mode Register
	0x341		S5_MR1	
0x342	0x342	S5_CR	S5_CR0	Reserved
	0x343		S5_CR1	SOCKET5 Command Register
0x344	0x344	S5_IMR	S5_IMR0	Reserved
	0x345		S5_IMR1	SOCKET5 Interrupt Mask Register
0x346	0x346	S5_IR	S5_IR0	Reserved
	0x347		S5_IR1	SOCKET5 Interrupt Register
0x348	0x348	S5_SSR	S5_SSR0	Reserved
	0x349		S5_SSR1	SOCKET5 SOCKET Status Register
0x34A	0x34A	S5_PORTR	S5_PORTR0	SOCKET5 Source Port Register
	0x34B		S5_PORTR1	
0x34C	0x34C	S5_DHAR	S5_DHAR0	SOCKET5 Destination Hardware Address Register
	0x34D		S5_DHAR1	
0x34E	0x34E	S5_DHAR2	S5_DHAR2	
	0x34F		S5_DHAR3	
0x350	0x350	S5_DHAR4	S5_DHAR4	
	0x351		S5_DHAR5	
0x352	0x352	S5_DPORTR	S5_DPORTR0	SOCKET5 Destination Port Register
	0x353		S5_DPORTR1	
0x354	0x354	S5_DIPR	S5_DIPR0	SOCKET5 Destination IP Address Register
	0x355		S5_DIPR1	
0x356	0x356	S5_DIPR2	S5_DIPR2	
	0x357		S5_DIPR3	
0x358	0x358	S5_MSSR	S5_MSSR0	SOCKET5 Maximum Segment Size Register
	0x359		S5_MSSR1	
0x35A	0x35A	S5_PORTOR	S5_KPALVTR	SOCKET5 Keep Alive Time Register
	0x35B		S5_PROTOR	SOCKET5 Protocol Number Register
0x35C	0x35C	S5_TOSR	S5_TOSR0	Reserved
	0x35D		S5_TOSR1	SOCKET5 TOS Register
0x35E	0x35E	S5_TTLR	S5_TTLR0	Reserved
	0x35F		S5_TTLR1	SOCKET5 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x360	0x360	S5_TX_WRSR	S5_TX_WRSR0	Reserved
	0x361		S5_TX_WRSR1	SOCKET5 TX Write Size Register
0x362	0x362	S5_TX_WRSR2	S5_TX_WRSR2	
	0x363		S5_TX_WRSR3	
0x364	0x364	S5_TX_FSR	S5_TX_FSR0	Reserved
	0x365		S5_TX_FSR1	SOCKET5 TX Free Size Register
0x366	0x366	S5_TX_FSR2	S5_TX_FSR2	
	0x367		S5_TX_FSR3	
0x368	0x368	S5_RX_RSR	S5_RX_RSR0	Reserved
	0x369		S5_RX_RSR1	SOCKET5 RX Receive Size Register
0x36A	0x36A	S5_RX_RSR2	S5_RX_RSR2	
	0x36B		S5_RX_RSR3	
0x36C	0x36C	S5_FRAGR	S5_FRAGR0	Reserved
	0x36D		S5_FRAGR1	SOCKET5 IP FLAG Field Register
0x36E	0x36E	S5_TX_FIFOR	S5_TX_FIFOR0	SOCKET5 TX FIFO Register
	0x36F		S5_TX_FIFOR1	
0x370	0x370	S5_RX_FIFOR	S5_RX_FIFOR0	SOCKET5 RX FIFO Register
	0x371		S5_RX_FIFOR1	
0x372	0x372			Reserved
	0x373			
:				:
:				:
0x37E	0x37E			Reserved
	0x37F			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x380	0x380	S6_MR	S6_MR0	SOCKET6 Mode Register
	0x381		S6_MR1	
0x382	0x382	S6_CR	S6_CR0	Reserved
	0x383		S6_CR1	SOCKET6 Command Register
0x384	0x384	S6_IMR	S6_IMR0	Reserved
	0x385		S6_IMR1	SOCKET6 Interrupt Mask Register
0x386	0x386	S6_IR	S6_IR0	Reserved
	0x387		S6_IR1	SOCKET6 Interrupt Register
0x388	0x388	S6_SSR	S6_SSR0	Reserved
	0x389		S6_SSR1	SOCKET6 SOCKET Status Register
0x38A	0x38A	S6_PORTR	S6_PORTR0	SOCKET6 Source Port Register
	0x38B		S6_PORTR1	
0x38C	0x38C	S6_DHAR	S6_DHAR0	SOCKET6 Destination Hardware Address Register
	0x38D		S6_DHAR1	
0x38E	0x38E	S6_DHAR2	S6_DHAR2	
	0x38F		S6_DHAR3	
0x390	0x390	S6_DHAR4	S6_DHAR4	
	0x391		S6_DHAR5	
0x392	0x392	S6_DPORTR	S6_DPORTR0	SOCKET6 Destination Port Register
	0x393		S6_DPORTR1	
0x394	0x394	S6_DIPR	S6_DIPR0	SOCKET6 Destination IP Address Register
	0x395		S6_DIPR1	
0x396	0x396	S6_DIPR2	S6_DIPR2	
	0x397		S6_DIPR3	
0x398	0x398	S6_MSSR	S6_MSSR0	SOCKET6 Maximum Segment Size Register
	0x399		S6_MSSR1	
0x39A	0x39A	S6_PORTOR	S6_KPALVTR	SOCKET6 Keep Alive Time Register
	0x39B		S6_PROTOR	SOCKET6 Protocol Number Register
0x39C	0x39C	S6_TOSR	S6_TOSR0	Reserved
	0x39D		S6_TOSR1	SOCKET6 TOS Register
0x39E	0x39E	S6_TTLR	S6_TTLR0	Reserved
	0x39F		S6_TTLR1	SOCKET6 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x3A0	0x3A0	S6_TX_WRSR	S6_TX_WRSR0	Reserved
	0x3A1		S6_TX_WRSR1	SOCKET6 TX Write Size Register
0x3A2	0x3A2	S6_TX_WRSR2	S6_TX_WRSR2	
	0x3A3		S6_TX_WRSR3	
0x3A4	0x3A4	S6_TX_FSR	S6_TX_FSR0	Reserved
	0x3A5		S6_TX_FSR1	SOCKET6 TX Free Size Register
0x3A6	0x3A6	S6_TX_FSR2	S6_TX_FSR2	
	0x3A7		S6_TX_FSR3	
0x3A8	0x3A8	S6_RX_RSR	S6_RX_RSR0	Reserved
	0x3A9		S6_RX_RSR1	SOCKET6 RX Receive Size Register
0x3AA	0x3AA	S6_RX_RSR2	S6_RX_RSR2	
	0x3AB		S6_RX_RSR3	
0x3AC	0x3AC	S6_FRAGR	S6_FRAGR0	Reserved
	0x3AD		S6_FRAGR1	SOCKET6 IP FLAG Field Register
0x3AE	0x3AE	S6_TX_FIFOR	S6_TX_FIFOR0	SOCKET6 TX FIFO Register
	0x3AF		S6_TX_FIFOR1	
0x3B0	0x3B0	S6_RX_FIFOR	S6_RX_FIFOR0	SOCKET6 RX FIFO Register
	0x3B1		S6_RX_FIFOR1	
0x3B2	0x3B2			Reserved
	0x3B3			
:				:
:				:
0x3BE	0x3BE			Reserved
	0x3BF			

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x3C0	0x3C0	S7_MR	S7_MR0	SOCKET7 Mode Register
	0x3C1		S7_MR1	
0x3C2	0x3C2	S7_CR	S7_CR0	Reserved
	0x3C3		S7_CR1	SOCKET7 Command Register
0x3C4	0x3C4	S7_IMR	S7_IMR0	Reserved
	0x3C5		S7_IMR1	SOCKET7 Interrupt Mask Register
0x3C6	0x3C6	S7_IR	S7_IR0	Reserved
	0x3C7		S7_IR1	SOCKET7 Interrupt Register
0x3C8	0x3C8	S7_SSR	S7_SSR0	Reserved
	0x3C9		S7_SSR1	SOCKET7 SOCKET Status Register
0x3CA	0x3CA	S7_PORTR	S7_PORTR0	SOCKET7 Source Port Register
	0x3CB		S7_PORTR1	
0x3CC	0x3CC	S7_DHAR	S7_DHAR0	SOCKET7 Destination Hardware Address Register
	0x3CD		S7_DHAR1	
0x3CE	0x3CE	S7_DHAR2	S7_DHAR2	
	0x3CF		S7_DHAR3	
0x3D0	0x3D0	S7_DHAR4	S7_DHAR4	
	0x3D1		S7_DHAR5	
0x3D2	0x3D2	S7_DPORTR	S7_DPORTR0	SOCKET7 Destination Port Register
	0x3D3		S7_DPORTR1	
0x3D4	0x3D4	S7_DIPR	S7_DIPR0	SOCKET7 Destination IP Address Register
	0x3D5		S7_DIPR1	
0x3D6	0x3D6	S7_DIPR2	S7_DIPR2	
	0x3D7		S7_DIPR3	
0x3D8	0x3D8	S7_MSSR	S7_MSSR0	SOCKET7 Maximum Segment Size Register
	0x3D9		S7_MSSR1	
0x3DA	0x3DA	S7_PORTOR	S7_KPALVTR	SOCKET7 Keep Alive Time Register
	0x3DB		S7_PROTOR	SOCKET7 Protocol Number Register
0x3DC	0x3DC	S7_TOSR	S7_TOSR0	Reserved
	0x3DD		S7_TOSR1	SOCKET7 TOS Register
0x3DE	0x3DE	S7_TTLR	S7_TTLR0	Reserved
	0x3DF		S7_TTLR1	SOCKET7 TTL Register

Address offset		Symbol		Description
16Bit	8Bit	16Bit	8Bit	
0x3E0	0x3E0	S7_TX_WRSR	S7_TX_WRSR0	Reserved
	0x3E1		S7_TX_WRSR1	SOCKET7 TX Write Size Register
0x3E2	0x3E2	S7_TX_WRSR2	S7_TX_WRSR2	
	0x3E3		S7_TX_WRSR3	
0x3E4	0x3E4	S7_TX_FSR	S7_TX_FSR0	Reserved
	0x3E5		S7_TX_FSR1	SOCKET7 TX Free Size Register
0x3E6	0x3E6	S7_TX_FSR2	S7_TX_FSR2	
	0x3E7		S7_TX_FSR3	
0x3E8	0x3E8	S7_RX_RSR	S7_RX_RSR0	Reserved
	0x3E9		S7_RX_RSR1	SOCKET7 RX Receive Size Register
0x3EA	0x3EA	S7_RX_RSR2	S7_RX_RSR2	
	0x3EB		S7_RX_RSR3	
0x3EC	0x3EC	S7_FRAGR	S7_FRAGR0	Reserved
	0x3ED		S7_FRAGR1	SOCKET7 IP FLAG Field Register
0x3EE	0x3EE	S7_TX_FIFOR	S7_TX_FIFOR0	SOCKET7 TX FIFO Register
	0x3EF		S7_TX_FIFOR1	
0x3F0	0x3F0	S7_RX_FIFOR	S7_RX_FIFOR0	SOCKET7 RX FIFO Register
	0x3F1		S7_RX_FIFOR1	
0x3F2	0x3F2			Reserved
	0x3F3			
:				:
:				:
0x3FE	0x3FE			Reserved
	0x3FF			

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	X	U(R)	0	0
Symbol of high address Reg.	Bit 7	6	5	4	3	2	1	0
Physical Address	Symbol	-	-	-	-	-	-	-
Address offset	Reset Value	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

16 bit Register Symbol(AO1/AO2)	
8bit Register Symbol (AO1/AO2)	8bit Register Symbol (AO1/AO2)
MSB(Value)	LSB(Value)

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.

MR0	15	14	13	12	11	10	9	8
0x08000	DBW	MPF	WDF2	WDF1	WDF0	RDH	-	FS
----	U(R)	0(R)	1	1	1	0	0	0
MR1	7	6	5	4	3	2	1	0
0x08001	RST	-	MT	PB	PPPoE	DBS	-	IND
----	0	0	0	0	0	0	0	0

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

Bit	Symbol	Description
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** When Host-Write operation, since '/CS' is asserted low, W5300 fetches Write-Data after WRF X PLL_CLK. 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.
12	WDF1	
11	WDF0	
10	RDH	**Read Data Hold Time**

		0 : No use data hold time 1 : Use data hold time (2 X PLL_CLK) When Host-Read operation, W5300 holds the Read-Data during 2 X PLL_CLK after Host-Read operation is finished ('/CS' is de-asserted high). In this case, be careful of collision of data bus.
9	-	Reserved
8	FS	**FIFO Swap Bit** 0 : Disable swap 1 : Enable swap 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_FIFOR/Sn_RX_FIFOR by setting this bit as '1', and use it as like little-endian.

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

Bit	Symbol	Description
7	RST	**S/W Reset** If it's set as '1', W5300 is reset. This bit is automatically cleared after reset.
6	-	Reserved
5	MT	**Memory Test Bit** 0 : Disable internal TX/RX memory test 1 : Enable internal TX/RX memory test Basically, internal TX memory of W5300 supports Host-Write operation through Sn_TX_FIFOR, and internal RX memory does Host-Read operation through Sn_RX_FIFOR. However if this bit is set as '1', internal TX/RX memory supports both of Host-Read and Host-Write operation through Sn_TX_FIFOR/Sn_RX_FIFOR , 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".*
4	PB	**Ping Block Mode**

		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_PROTOR1=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.

IDM_AR0 = MSB (0x03) of address offset of S4_RX_FIFOR

IDM_AR1 = LSB (0x30) of address offset of S4_RX_FIFOR

IDM_AR(0x08002/----)	
IDM_AR0(0x08002/----)	IDM_AR1(0x08003/----)
0x03	0x30

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,

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')
IDM_AR = 0x0002 IDM_DR = 0x80F0	IDM_AR0 = 0x00 IDM_AR1 = 0x02 IDM_DR0 = 0x80 IDM_DR1 = 0xF0

Ex2) When the host reads IR(0x0FE) and saves it in variable 'val',

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')
IDM_AR = 0x0002 val = IDM_DR	IDM_AR0 = 0x00 IDM_AR1 = 0x02 val = IDM_DR0 val = (val << 8) + IDM_DR1

IDM_AR(0x08002/----)	
IDM_AR0(0x08002/----)	IDM_AR1(0x08003/----)
0x00	0x02

IDM_DR(0x08004/----)	
IDM_DR0(0x08004/----)	IDM_DR1(0x08005/----)
MSB(IR0) of IR	LSB(IR1) of IR

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.

IR0	15	14	13	12	11	10	9	8
0x08002	IPCF	DPUR	PPPT	FMTU	-	-	-	-
0x002	0	0	0	0	0	0	0	0
IR1	7	6	5	4	3	2	1	0
0x08003	S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT
0x003	0	0	0	0	0	0	0	0

IR(15:8)/IR0(7:0)

Bit	Symbol	Description
15	IPCF	**IP Conflict** 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.
14	DPUR	**Destination Port unreachable** It's set as '1' when receiving ICMP(Destination port unreachable) packet. Refer to UIPR and UPORTR.
13	PPPT	**PPPoE Terminate** When the connection with server is closed at the PPPoE mode, it is set as '1'.
12	FMTU	**Fragment MTU** When receiving ICMP (Fragment MTU) packet, it's set as '1' Refer to FMTUR.
11	-	Reserved
10	-	Reserved
9	-	Reserved
8	-	Reserved

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

Bit	Symbol	Description
7	S7_INT	**Occurrence of SOCKET7 Interrupt** 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.
6	S6_INT	**Occurrence of SOCKET6 Interrupt** When an interrupt occurs at the SOCKET6, it becomes '1'. This interrupt

		information is applied to S6_IR1. This bit is automatically cleared when S6_IR1 is cleared to 0x00 by host.
5	S5_INT	**Occurrence of SOCKET5 Interrupt** 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.
4	S4_INT	**Occurrence of SOCKET4 Interrupt** 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.
3	S3_INT	**Occurrence of SOCKET3 Interrupt** 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.
2	S2_INT	**Occurrence of SOCKET2 Interrupt** 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.
1	S1_INT	**Occurrence of SOCKET1 Interrupt** 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.
0	S0_INT	**Occurrence of SOCKET0 Interrupt** 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.

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'.

IMR0	15	14	13	12	11	10	9	8
0x08004	IPCF	DPUR	PPPT	FMTU	-	-	-	-
0x004	0	0	0	0	0	0	0	0
IMR1	7	6	5	4	3	2	1	0
0x08005	S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT
0x005	0	0	0	0	0	0	0	0

IMR(15:8)/IMR0(7:0)

Bit	Symbol	Description
15	IPCF	**IR(IPCF) Interrupt Mask**
14	DPUR	**IR(DPUR) Interrupt Mask**
13	PPPT	**IR(PPPT) Interrupt Mask**
12	FMTU	**IR(FMTU) Interrupt Mask**
11	-	Reserved
10	-	Reserved
9	-	Reserved
8	-	Reserved

IMR(7:0)/IMR1(7:0)

Bit	Symbol	Description
7	S7_INT	**IR(S7_INT) Interrupt Mask**
6	S6_INT	**IR(S6_INT) Interrupt Mask**
5	S5_INT	**IR(S5_INT) Interrupt Mask**
4	S4_INT	**IR(S4_INT) Interrupt Mask**
3	S3_INT	**IR(S3_INT) Interrupt Mask**
2	S2_INT	**IR(S2_INT) Interrupt Mask**
1	S1_INT	**IR(S1_INT) Interrupt Mask**
0	S0_INT	**IR(S0_INT) Interrupt Mask**

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”

SHAR(0x08008/0x008)	
SHAR0(0x08008/0x008)	SHAR1(0x08009/0x009)
0x00	0x08
SHAR2(0x0800A/0x00A)	
SHAR2(0x0800A/0x00A)	SHAR3(0x0800B/0x00B)
0xDC	0x01
SHAR4(0x0800C/0x00C)	
SHAR4(0x0800C/0x00C)	SHAR5(0x0800D/0x00D)
0x02	0x03

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”

GAR(0x08010/0x010)		GAR2(0x08012/0x012)	
GAR0(0x08010/0x010)	GAR1(0x08011/0x011)	GAR2(0x08012/0x012)	GAR3(0x08013/0x013)
192(0xC0)	168(0xA8)	0(0x00)	1(0x01)

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”

SUBR(0x08014/0x014)		SUBR2(0x08016/0x016)	
SUBR0(0x08014/0x014)	SUBR1(0x08015/0x015)	SUBR2(0x08016/0x016)	SUBR3(0x08017/0x017)
255 (0xFF)	255 (0xFF)	255 (0xFF)	0 (0x00)

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”

SIPR(0x08018/0x018)		SIPR2(0x0801A/0x01A)	
SIPR0(0x08018/0x018	SIPR1(0x08019/0x019	SIPR2(0x0801A/0x01A	SIPR3(0x0801B/0x01B

)	)	)	)
192(0xC0)	168(0xA8)	0(0x00)	3(0x03)

RTR (Retransmission Timeout-period Register) [R/W] [0x0801C/0x01C] [0x07D0]

It configures 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)

RTR(0x0801C/0x01C)	
RTR0(0x0801C/0x01C)	RTR1(0x0801D/0x01D)
0x0F	0xA0

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

RCR(0x0801E/0x01E)	
RCR0(0x0801E/0x01C)	RCR1(0x0801F/0x01F)
Reserved	0x07

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) 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 re-transmitted. 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,

$$TCP_{TO} = \left(\sum_{N=0}^{M} (RTR \times 2^N) + ((RCR-M) \times RTR_{MAX}) \right) \times 0.1ms$$

N : Retransmission count, 0 <= N <= M
M : Minimum value when RTR X $2^{(M+1)}$ > 65535 and 0 <= M <= RCR
RTR_{MAX} : RTR X 2^M

Ex) When RTR = 2000(0x07D0), RCR = 8(0x0008),

ARP_{TO} = 2000 X 0.1ms X 9 = 1800ms = 1.8s

TCP_{TO} = (0x07D0 + 0x0FA0 + 0x1F40 + 0x3E80 + 0x7D00 + 0xFA00 + 0xFA00 + 0xFA00 + 0xFA00) X 0.1ms

= (2000 + 4000 + 8000 + 16000 + 32000 + ((8 - 4) X 64000)) X 0.1ms

= 318000 X 0.1ms = 31.8s

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

TMS01R(0x08020/0x020)	
TMSR0(0x08020/0x020)	TMSR1(0x08021/0x021)
4 (0x04)	16 (0x10)

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

TMS23R(0x08020/0x020)	
TMSR2(0x08022/0x022)	TMSR3(0x08023/0x023)
1 (0x01)	20 (0x14)

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

TMS45R(0x08024/0x024)	
TMSR4(0x08024/0x024)	TMSR5(0x08025/0x025)
0 (0x00)	7 (0x07)

TMS67R(TX Memory Size of SOCKET6/7 Register) [R/W] [0x08024/0x024] [0x0808]

It configures internal TX memory size of SOCKET6 and SOCKET7.

Ex4) SOCKET6 : 12KB, SOCKET7 : 12KB

TMS67R(0x08026/0x026)	
TMSR6(0x08026/0x026)	TMSR7(0x08027/0x027)
12 (0x0C)	12 (0x0C)

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]

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_{SUM} and TMS_{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

RMS01R(0x08028/0x028)	
RMSR0(0x08028/0x028)	RMSR1(0x08029/0x029)
17 (0x11)	3 (0x03)

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

RMS23R(0x0802A/0x02A)	
RMSR2(0x0802A/0x02A)	RMSR3(0x0802B/0x02B)
5 (0x05)	16 (0x10)

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

RMS45R(0x0802C/0x02C)	
RMSR4(0x0802C/0x02C)	RMSR5(0x0802D/0x02D)
3 (0x03)	4 (0x04)

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

RMS67R(0x0802E/0x02E)	
RMSR6(0x0802E/0x02E)	RMSR7(0x0802F/0x02F)
4 (0x04)	4 (0x04)

As shown above Ex1) ~ Ex8), RMS_{SUM}(RMSR0 + RMSR1 + RMSR2 + RMSR3 + RMSR4 + RMSR5 + RMSR6 + RMSR7) is set as 56. The sum of TMS_{SUM} and RMS_{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'.

MTYPER0	15	14	13	12	11	10	9	8
0x08030	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
0x030	0	0	0	0	0	0	0	0
MTYPER1	7	6	5	4	3	2	1	0
0x08031	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
0x031	1	1	1	1	1	1	1	1

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

Bit	Symbol	Description
15	MB15	**16th Memory Block Type**
14	MB14	**15th Memory Block Type**
13	MB13	**14th Memory Block Type**
12	MB12	**13th Memory Block Type**
11	MB11	**12th Memory Block Type**
10	MB10	**11th Memory Block Type**
9	MB9	**10th Memory Block Type**
8	MB8	**9th Memory Block Type**

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

Bit	Symbol	Description
7	MB7	**8th Memory Block Type**
6	MB6	**7th Memory Block Type**
5	MB5	**6th Memory Block Type**
4	MB4	**5th Memory Block Type**
3	MB3	**4th Memory Block Type**
2	MB2	**3rd Memory Block Type**
1	MB1	**2nd Memory Block Type**
0	MB0	**1st Memory Block Type**

Ex1) TMS_{SUM} = 72, RMS_{SUM} = 56

As 72 / 8 = 9, from MB0 to MB8 are set as TX memory.

MTYPER(0x08030/0x030)	
MTYPER0(0x08030/0x030)	MTYPER1(0x08031/0x031)
0x01	0xFF

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

MTYPER(0x08030/0x030)	
MTYPER0(0x08030/0x030)	MTYPER1(0x08031/0x031)
0xFF	0xFF

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

MTYPER(0x08030/0x030)	
MTYPER0(0x08030/0x030)	MTYPER1(0x08031/0x031)
0x00	0x00

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

It notifies authentication method negotiated with PPPoE server.

W5300 supports 2 types of authentication methods.

Value	Authentication method
0xC023	PAP
0xC223	CHAP

Ex) PATR = 'CHAP'

PATR(0x08032/0x032)	
PATR0(0x08032/0x032)	PATR1(0x08033/0x033)
0xC2	0x23

PTIMER(PPP Link Control Protocol Request Timer Register)[R/W][0x08036/0x036][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)

PTIMER(0x08036/0x037)	
PTIMER0(0x08036/0x036)	PTIMER1(0x08037/0x037)
Reserved	200 (0xC8)

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

PMAGICR(0x08036/0x037)	
PMAGICR0(0x08038/0x038)	PMAGICR1(0x08039/0x039)
Reserved	0x01

Magic number = 0x01010101

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

PSIDR(0x0803C/0x03C)	
PSIDR0(0x0803C/0x03C)	PSIDR1(0x0803D/0x03D)
0x00	0x17

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

PDHAR(0x08040/0x040)	
PDHAR0(0x08040/0x040)	PDHAR1(0x08041/0x041)
0x00	0x01
PDHAR2(0x08042/0x042)	
PDHAR2(0x08042/0x042)	PDHAR3(0x08043/0x043)
0x02	0x03
PDHAR4(0x08044/0x044)	
PDHAR4(0x08044/0x044)	PDHAR5(0x08045/0x045)
0x04	0x05

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

UIPR(0x08048/0x048)		UIPR2(0x0804A/0x04A)	
UIPR0(0x08048/0x048)	UIPR1(0x08049/0x049)	UIPR2(0x0804A/0x04A)	UIPR3(0x0804B/0x04B)
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

Ex2) UPORT = 5000(0x1388)

UPORTR(0x0804C/0x04C)	
UPORTR0(0x0804C/0x04C)	UPORTR1(0x0804D/0x04D)
0x13	0x18

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)

FMTUR(0x0804E/0x04E)	
FMTUR0(0x0804E/0x04E)	FMTUR1(0x0804F/0x04F)
0x02	0x00

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_BDPTHR, or received buffer size of RX memory is same or bigger than the Pn_BDPTHR, PIN "BRDYn" is signaled.

Pn_BRDYR0	15	14	13	12	11	10	9	8
0x08060 + 4n	-	-	-	-	-	-	-	-
0x060 + 4n	0	0	0	0	0	0	0	0
Pn_BRDYR1	7	6	5	4	3	2	1	0
0x08061	PEN	PMT	PPL	-	-	SN2	SN1	SN0
0x061	0	0	1	0	0	0	0	0

Pn_BRDYR(7:0)/Pn_BRDYR1(7:0)

Bit	Symbol	Description
7	PEN	**PIN "BRDYn" Enable** 0 : Disable BRDYn 1 : Enable BRDYn When using PIN "BRDYn", set it as "1".
6	PMT	**PIN Memory Type** 0 : RX memory 1 : TX memory It sets the type of SOCKET memory to monitor.
5	PPL	**PIN Polarity** 0 : Low sensitive 1 : High sensitive When Free/Received buffer size of TX/RX memory is same or bigger than Pn_DPTHR, set the logic level of PIN "BRDYn" to be signaled to the host.
4	-	Reserved
3	-	Reserved
2	SN2	**SOCKET Number** Set the SOCKET number to monitor through PIN "BRDYn".
1	SN1	
0	SN0	

	SN2	SN1	SN0		SN2	SN1	SN0
7	1	1	1	3	0	1	1
6	1	1	0	2	0	1	0
5	1	0	1	1	0	0	1
4	1	0	0	0	0	0	0

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,

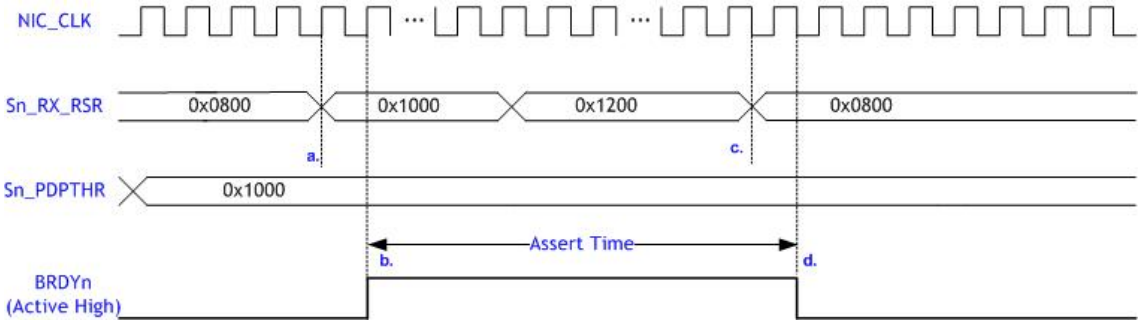
P3_BRDYR = 0x00E5

P3_BRDYR(0x0806C/0x06C)	
P3_BRDYR0(0x0806C/0x06C)	P3_BRDYR1(0x0806D/0x06D)
Reserved	0xE5

P3_BDPTHR = 2048(0x0800)

P3_BDPTHR(0x0806E/0x06E)	
P3_BDPTHR0(0x0806E/0x06E)	P3_BDPTHR1(0x0806F/0x06F)
0x08	0x00

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



a. Sn_RX_RSR > Sn_BDPTHR 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_BDPTHR" 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_BDPTHR" (at least 80ns).

Fig 4. 'BRDYn' Timing

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

It notifies W5300's ID value.

IDR(0x080FE/0x0FE)	
FMTUR0(0x080FE/0x0FE)	FMTUR1(0x080FF/0x0FF)
0x53	0x00

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.

Sn_MR0	15	14	13	12	11	10	9	8
0x08200 + 0x40n	-	-	-	-	-	-	-	ALIGN
0x200 + 0x40n	0	0	0	0	0	0	0	0
Sn_MR1	7	6	5	4	3	2	1	0
0x08201 + 0x40n	MULTI	-	ND/MC	-	P3	P2	P1	P0
0x201 + 0x40n	0	0	1	0	0	0	0	0

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

Bit	Symbol	Description
15	-	Reserved
14	-	Reserved
13	-	Reserved
12	-	Reserved
11	-	Reserved
10	-	Reserved
9	-	Reserved
8	ALIGN	**Alignment** 0 : No use alignment 1 : Use alignment 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)

Bit	Symbol	Description
7	MULTI	**Multicasting** 0 : Disable multicasting 1 : Enable multicasting

		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.
6	MF	**MAC Filter** 0 : Disable MAC filter 1 : Enable MAC filter 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.
5	ND/IGMPv	**Use No Delayed ACK** 0 : Disable no delayed ACK option 1 : Enable no delayed ACK option It is valid in TCP(P3~P0 : “0001”). In case that it is set as '1', ACK packet is transmitted right after receiving DATA packet from the peer. It is recommended to be set as '1' for TCP performance improvement. In case that it is set as '0', ACK packet is transmitted after the time set in RTR regardless of DATA packet receipt. **IGMP version** 0 : using IGMP version 2 1 : using IGMP version 1 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.
4	-	Reserved
3	P3	**Protocol** It configures communication protocol (TCP, UDP, IP RAW, MAC RAW) in each SOCKET or PPPoE SOCKET to operate with PPPoE server.

2	P2	
1	P1	
0	P0	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.

Symbol	P3	P2	P1	P0	Meaning
Sn_MR_CLOSE	0	0	0	0	Closed
Sn_MR_TCP	0	0	0	1	TCP
Sn_MR_UDP	0	0	1	0	UDP
Sn_MR_IPRAW	0	0	1	1	IP RAW
S0_MR_MACRAW	0	1	0	0	MAC RAW
S0_MR_PPPoE	0	1	0	1	

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.

Sn_CR(0x08202+0x40n/0x202+0x40n)	
Sn_CR0(0x08202+0x40n/0x202+0x40n)	Sn_CR1(0x08203+0x40n/0x203+0x40n)
Reserved	Command

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

Value	Command	Description
0x01	OPEN	It initializes SOCKETn and opens according to protocol type set in Sn_MR(P3:P0). Below is the value change of Sn_SSR according to Sn_MR(P3:P0)

Sn_MR(P3:P0)	Sn_SSR
Sn_MR_CLOSE	-
Sn_MR_TCP	SOCK_INIT
Sn_MR_UDP	SOCK_UDP
Sn_MR_IPRAW	SOCK_IPRAW
S0_MR_MACRAW	SOCK_MACRAW
S0_MR_PPPoE	SOCK_PPPoE

0x02	LISTEN	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.
0x04	CONNECT	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.
0x08	DISCON	Only valid in TCP mode Regardless of "TCP SERVER" or “TCP CLINET”, it performs disconnect-process. - Active close : it transmits disconnect-request(FIN packet) to the connected peer.

		– 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.
0x10	CLOSE	It closes SOCKETn. Sn_SSR is changed to SOCK_CLOSED.
0x20	SEND	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> 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.
0x21	SEND_MAC	Valid only in UDP (Sn_MR(P3:P0)=Sn_MR_UDP) or IPRAW((Sn_MR(P3:P0) = Sn_MR_IPRAW) mode.

		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.
0x22	SEND_KEEP	Valid only in TCP mode. In order to check TCP connection status with the peer, KEEP ALIVE(KA) packet can be transmitted. SEND_KEEP is available only in case of 'Sn_KPALVTR=0', but ignored in case of 'Sn_KPALVTR>0'. In case of 'Sn_KPALVTR > 0', KA packet is automatically transmitted if there is no data communication during the time of Sn_KPALVTR. 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. cf> KA packet can be transmitted after one or more data communication is processed.
0x40	RECV	It notifies that the host received the data packet of SOCKETn cf> Before RECV command, the host should copy receiving data packet from internal RX memory into the host memory through Sn_RX_FIFOR.
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".*		
0x23	PCON	PPPoE connection begins by transmitting PPPoE discovery packet.
0x24	PDISCON	Closes PPPoE connection.
0x25	PCR	In each phase, it transmits REQ message.
0x26	PCN	In each phase, it transmits NAK message.
0x27	PCJ	In each phase, it transmits REJECT message.

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.)

Sn_IMR0	15	14	13	12	11	10	9	8
0x08204 + 0x40n	-	-	-	-	-	-	-	-
0x204 + 0x40n	0	0	0	0	0	0	0	0
Sn_IMR1	7	6	5	4	3	2	1	0
0x08205 + 0x40n	PRECV	PFAIL	PNEXT	SENDOK	TIMEOUT	RECV	DISCON	CON
0x205 + 0x40n	1	1	1	1	1	1	1	1

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

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

Bit	Symbol	Description
7	PRECV	**Sn_IR(PRECV) Interrupt Mask** Valid only in case of 'SOCKET=0' & 'S0_MR(P3:P0)=S0_MR_PPPoE'
6	PFAIL	**Sn_IR(PFAIL) Interrupt Mask** Valid only in case of 'SOCKET=0' & 'S0_MR(P3:P0)=S0_MR_PPPoE'
5	PNEXT	**Sn_IR(PNEXT) Interrupt Mask** Valid only in case of 'SOCKET=0' & 'S0_MR(P3:P0)=S0_MR_PPPoE'
4	SENDOK	**Sn_IR(SENDOK) Interrupt Mask**
3	TIMEOUT	**Sn_IR(TIMEOUT) Interrupt Mask**
2	RECV	**Sn_IR(RECV) Interrupt Mask**
1	DISCON	**Sn_IR(DISCON) Interrupt Mask**
0	CON	**Sn_IR(CON) Interrupt Mask**

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 SOCKENTn to the host.

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

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.

Sn_IR0	15	14	13	12	11	10	9	8
0x08206 + 0x40n	-	-	-	-	-	-	-	-
0x206 + 0x40n	0	0	0	0	0	0	0	0
Sn_IR1	7	6	5	4	3	2	1	0
0x08207 + 0x40n	PRECV	PFAIL	PNEXT	SENDOK	TIMEOUT	RECV	DISCON	CON
0x207 + 0x40n	0	0	0	0	0	0	0	0

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

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

Bit	Symbol	Description
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(0x08208+0x40n/0x208+0x40n)	
Sn_SSR0(0x08208+0x40n/0x208+0x40n)	Sn_SSR1(0x08209+0x40n/0x209+0x40n)
Reserved	SOCKET Status

Sn_SSR(15:8)/Sn_SSR0(7:0) : All Reserved

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

Value	Symbol	Description
0x00	SOCK_CLOSED	It is the status that resource of SOCKETn is released When DISCON or CLOSE command is performed, or ARP_{TO}, or TCP_{TO} occurs, it is changed to SOCK_CLOSED regardless of previous value.
0x13	SOCK_INIT	It is the status that SOCKETn is open as TCP mode. 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. It is possible to perform LISTEN command at the "TCP SERVER" mode and CONNECT command at the "TCP CLIENT".
0x14	SOCK_LISTEN	It is the status that SOCKETn operates as "TCP SERVER" and waits for connection-request (SYN packet) from "TCP CLIENT". When LISTEN command is performed, it is changed to SOCK_LISTEN. When connect-request(SYN packet) from "TCP CLIENT" is successfully processed, SOCK_LISTEN is changed to SOCK_ESTABLISHED. If it is failed, TCP_{TO} occurs(Sn_IR(TIME OUT)='1') and changed to SOCK_CLOSED.
0x17	SOCK_ESTABLISHED	It is the status that TCP connection is established. It is changed to SOCK_ESTABLISHED when SYN packet from "TCP CLIENT" is successfully processed at the SOCK_LISTEN, or CONNECT command is successfully performed. At this status, DATA packet can be transferred,

		that is, SEND or RECV command can be performed.
0x1C	SOCK_CLOSE_WAIT	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.
0x22	SOCK_UDP	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.
0x32	SOCK_IPRAW	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.
0x42	SOCK_MACRAW	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.
0x5F	SOCK_PPPoE	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"*.

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

0x15	SOCK_SYNSENT	It is the status that connect-request(SYN packet) is transmitted to "TCP SERVER".

		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_ESTBLISHED. 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	
0X1D	SOCK_LAST_ACK	
0x01	SOCK_ARP	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.

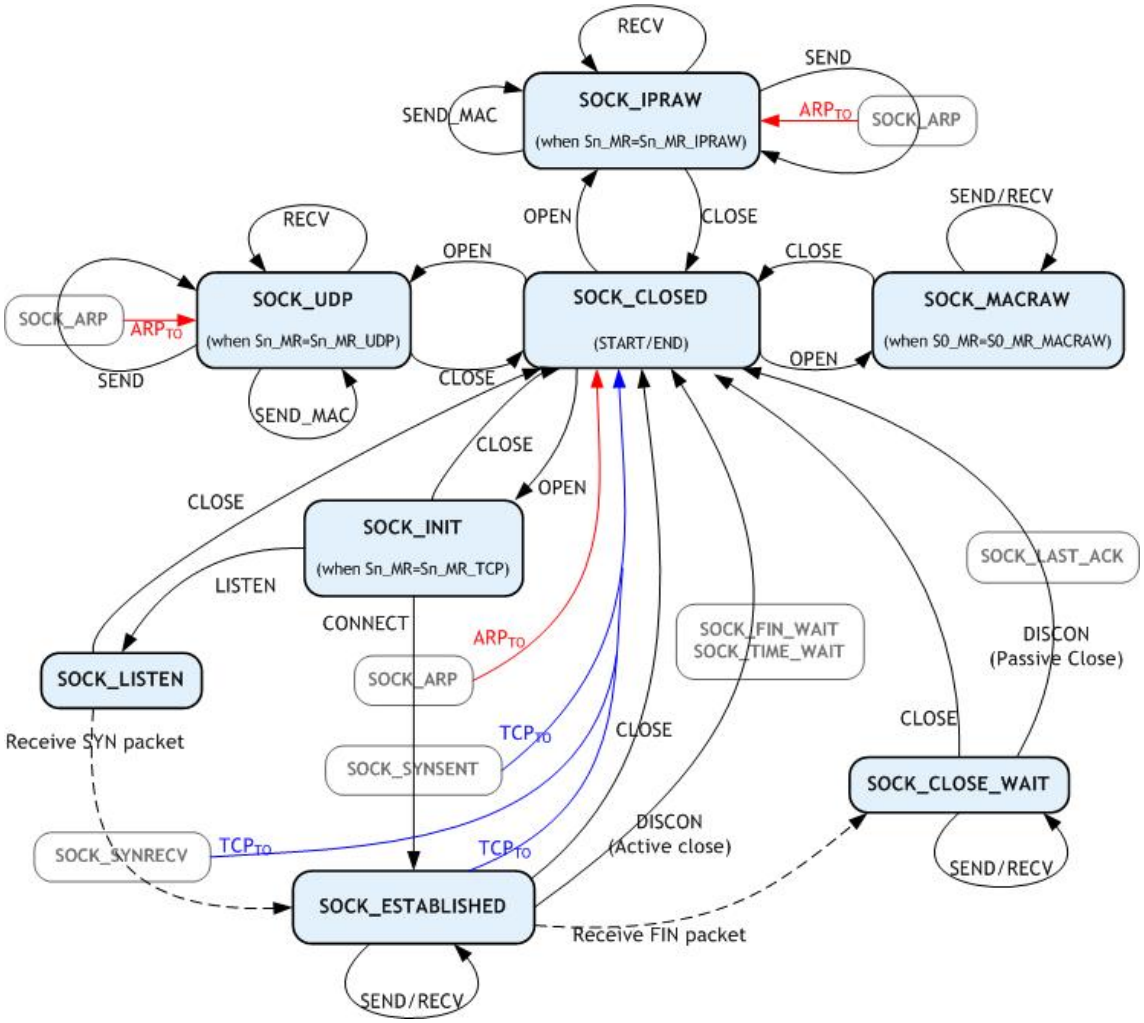


Fig 5. SOCKETn Status Transition

Sn_PORTR(SOCKETn Source Port Register)[R/W]

[0x0820A+0x40n/0x20A+0x40A] [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)

Sn_PORTR(0x0820A+0x40n/0x20A+0x40n)	
Sn_PORTR0(0x0820A+0x40n/0x20A+0x40n)	Sn_PORTR1(0x0820B+0x40n/0x20B+0x40n)
0x13	0x88

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

Sn_DHAR(0x0820C+0x40n/0x20C+0x040n)	
Sn_DHAR0(0x0820C+0x40n/0x20C+0x040n)	Sn_DHAR1(0x0820D+0x40n/0x20D+0x040n)
0x00	0x08
Sn_DHAR2(0x0820E+0x40n/0x20E+0x040n)	
Sn_DHAR2(0x0820E+0x40n/0x20E+0x040n)	Sn_DHAR3(0x0820F+0x40n/0x20F+0x040n)
0xDC	0x01
Sn_DHAR4(0x08210+0x40n/0x210+0x040n)	
Sn_DHAR4(0x08210+0x40n/0x210+0x040n)	Sn_DHAR5(0x08211+0x40n/0x211+0x040n)
0x02	0x10

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)

Sn_PORTR(0x08212+0x40n/0x212+0x40n)	
Sn_PORTR0(0x08212+0x40n/0x212+0x40n)	Sn_PORTR1(0x08213+0x40n/0x213+0x40n)
0x13	0x88

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

Sn_DIPR(0x08214+0x40n/0x214+0x040n)	
Sn_DIPR0(0x08214+0x40n/0x214+0x040n)	Sn_DIPR1(0x08215+0x40n/0x215+0x040n)
192 (0xC0)	168 (0xA8)
Sn_DHAR2(0x08216+0x40n/0x216+0x040n)	
Sn_DIPR2(0x08216+0x40n/0x216+0x040n)	Sn_DIPR3(0x08217+0x40n/0x217+0x040n)
0 (0x00)	11 (0x0B)

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.

Mode	**Normal** (MR(PPPoE)='0')		**PPPoE** (MR(PPPoE)='1')	
	Default MTU	Range	Default MTU	Range
TCP	1460	1 ~ 1460	1452	1 ~ 1452
UDP	1472	1 ~ 1472	1464	1 ~ 1464
IPRAW	1480		1472	
MACRAW	1514			

Ex) Sn_MSSR = 1460 (0x05B4)

Sn_MSSR(0x08218+0x40n/0x218+0x040n)	
Sn_MSSR0(0x08218+0x40n/0x218+0x040n)	Sn_MSSR1(0x08219+0x40n/0x219+0x040n)
0x05	0xB4

Sn_KPALVTR(SOCKETn Keep Alive Time Register)[R/W]

[0x0821A+40n/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.

Sn_PROTOR(0x0821A+0x40n/0x21A+0x040n)	
Sn_KPALVTR(0x0821A+0x40n/0x21A+0x040n)	Sn_PROTOR (0x0821B+0x40n/0x21B+0x040n)
10 (0x0A)	Sn_PROTOR

Sn_PROTOR (SOCKETn Protocol Number Register)[R/W]

[0x0821B+40n/0x21B+0x40n] [0x00]

It is 1 byte register that sets protocol number field of IP header at the IP layer.

It is valid only in IPRAW mode, and ignored in other modes. Sn_PROTOR is set before OPEN command. SOCKETn opened as IPRAW mode, transmits and receives the data of protocol number set in Sn_PROTOR. Sn_PROTOR can be assigned in the range of 0x00 ~ 0xFF, but W5300 does not support TCP(0x06) and UDP(0x11) protocol number

Protocol number is defined in IANA(Internet assigned numbers authority). For the detail, refer to online document (http://www.iana.org/assignments/protocol-numbers).

Ex) Sn_PROTOR = 0x01 (ICMP)

Sn_PROTOR(0x0821A+0x40n/0x21A+0x040n)	
Sn_KPALVTR(0x0821A+0x40n/0x21A+0x040n)	Sn_PROTOR (0x0821B+0x40n/0x21B+0x040n)
Sn_KPALVTR	0x01

Sn_TOSR (SOCKETn TOS Register) [R/W] [0x0821C+40n/0x21C+40n] [0x00]

It sets TOS(Type of Service) 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_TOSR = 0x00

Sn_TOSR(0x0821C+0x40n/0x21C+0x040n)	
Sn_TOSR0(0x0821C+0x40n/0x21C+0x040n)	Sn_TOSR1(0x0821D+0x40n/0x21D+0x040n)
Reserved	0x00

Sn_TTLR (SOCKETn TTL Register) [R/W] [0x0821E+40n/0x21E+40n] [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)

Sn_TTLR(0x0821E+0x40n/0x21E+0x040n)	
Sn_TTLR0(0x0821E+0x40n/0x21E+0x040n)	Sn_TTLR1(0x0821F+0x40n/0x21F+0x040n)
Reserved	0x80

Sn_TX_WRSR (SOCKETn TX Write Size Register) [R/W]

[0x08220+40n/0x220+40n] [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

Sn_TX_WRSR(0x08220+0x40n/0x220+0x040n)								
Sn_TX_WRSR0(0x08220+0x40n/0x220+0x040n)	Sn_TX_WRSR1(0x08221+0x40n/0x221+0x040n)							
Reserved	-	-	-	-	-	-	-	'1'
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)								
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)	Sn_TX_WRSR3(0x08223+0x40n/0x21D+0x040n)							
0x00	0x00							

Ex2) Sn_TX_WRSR = 2017 = 0x000007E1

Sn_TX_WRSR(0x08220+0x40n/0x220+0x040n)								
Sn_TX_WRSR0(0x08220+0x40n/0x220+0x040n)	Sn_TX_WRSR1(0x08221+0x40n/0x221+0x040n)							
Reserved	-	-	-	-	-	-	-	'0'
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)								
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)	Sn_TX_WRSR3(0x08223+0x40n/0x223+0x040n)							
0x07	0xE1							

Sn_TX_FSR (SOCKETn TX Free Size Register) [R]

[0x08224+40n/0x224+40n] [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

Sn_TX_FSR(0x08224+0x40n/0x224+0x040n)								
Sn_TX_FSR0(0x08224+0x40n/0x214+0x040n)	Sn_TX_FSR1(0x08225+0x40n/0x225+0x040n)							
Reserved	-	-	-	-	-	-	-	'1'
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)								
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)	Sn_TX_FSR3(0x08227+0x40n/0x227+0x040n)							
0x00	0x00							

Ex2) Sn_TX_FSR = 33332 = 0x00008234

Sn_TX_FSR(0x08224+0x40n/0x224+0x040n)								
Sn_TX_FSR0(0x08224+0x40n/0x224+0x040n)	Sn_TX_FSR1(0x08225+0x40n/0x225+0x040n)							
Reserved	-	-	-	-	-	-	-	'0'
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)								
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)	Sn_TX_FSR3(0x08227+0x40n/0x227+0x040n)							
0x82	0x34							

Sn_RX_RSR (SOCKETn RX Received Size Register) [R]

[0x08228+40n/0x228+40n] [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 2bytes 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

Sn_RX_RSR(0x08228+0x40n/0x228+0x040n)								
Sn_RX_RSR0(0x08228+0x40n/0x21C+0x040n)	Sn_RX_RSR1(0x08229+0x40n/0x229+0x040n)							
Reserved	-	-	-	-	-	-	-	'1'
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)								
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)	Sn_RX_RSR3(0x0822B+0x40n/0x22B+0x040n)							
0x00	0x00							

Ex2) Sn_RX_RSR = 3800 = 0x00000ED8

Sn_RX_RSR(0x08228+0x40n/0x228+0x040n)								
Sn_RX_RSR0(0x08228+0x40n/0x21C+0x040n)	Sn_RX_RSR1(0x08229+0x40n/0x229+0x040n)							
Reserved	-	-	-	-	-	-	-	'0'
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)								
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)	Sn_RX_RSR3(0x0822B+0x40n/0x22B+0x040n)							
0x0E	0xD8							

Sn_FRAGR (SOCKETn Fragment Register) [R/W] [0x0822C+40n/0x22C+40n] [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)

Sn_FRAGR(0x0822C+0x40n/0x22C+0x040n)	
Sn_FRAGR0(0x0822C+0x40n/0x22C+0x040n)	Sn_FRAGR1(0x0822D+0x40n/0x22D+0x040n)
Reserved	0x40

Sn_TX_FIFOR (SOCKETn TX FIFO Register) [R/W] [0x0822E+40n/0x22E+40n] [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

Sn_TX_FIFOR(0x0822E+0x40n/0x22E+0x040n)	
Sn_TX_FIFOR0(0x0822E+0x40n/0x22E+0x040n)	Sn_TX_FIFOR1(0x0822F+0x40n/0x22F+0x040n)
0x11	0x22

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

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')
Sn_TX_FIFOR = 0x6162 Sn_TX_FIFOR = 0x6364 Sn_TX_FIFOR = 0x6500 Sn_TX_WRSR0 = 0x0000 Sn_TX_WRSR1 = 0x0005 Sn_CR = 0x0020 (SEND command)	Sn_TX_FIFOR0 = 0x61 Sn_TX_FIFOR1 = 0x62 Sn_TX_FIFOR0 = 0x63 Sn_TX_FIFOR1 = 0x64 Sn_TX_FIFOR0 = 0x65 Sn_TX_FIFOR1 = 0x00 Sn_TX_WRSR0 = 0x00 Sn_TX_WRSR1 = 0x00 Sn_TX_WRSR2 = 0x00 Sn_TX_WRSR2 = 0x05 Sn_CR1 = 0x20 (SEND command)

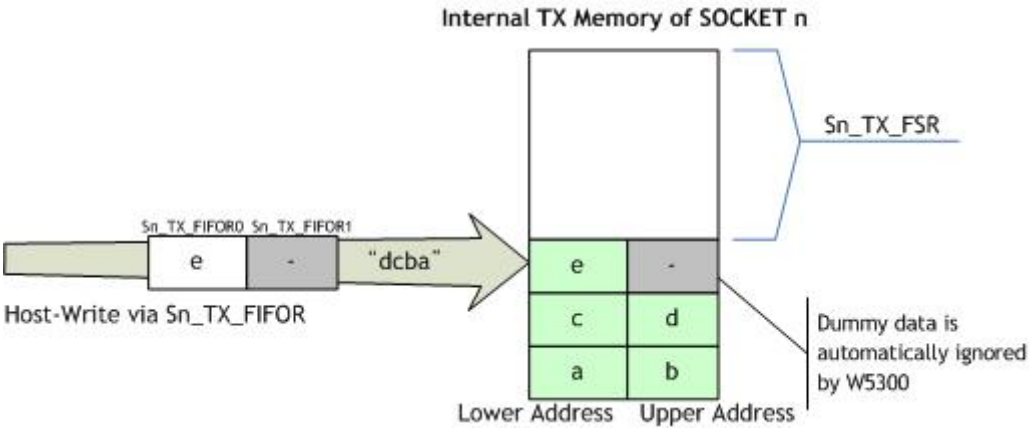


Fig 6. Access to Internal TX Memory

Sn_RX_FIFOR (SOCKETn RX FIFO Register) [R/W] [0x08230+40n/0x230+40n] [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

Sn_RX_FIFOR(0x08230+0x40n/0x230+0x040n)	
Sn_RX_FIFOR0(0x08230+0x40n/0x230+0x040n)	Sn_RX_FIFOR1(0x08231+0x40n/0x231+0x040n)
0x33	0x44

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

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')

INT16 pack_size, idx,temp
INT8 str[5]
pack_size = Sn_RX_FIFOR
idx = 0
LOOP pack_size/2
temp = Sn_RX_FIFOR
str[idx] = (INT8)(temp >> 8)
idx = idx + 1
str[idx] = (INT8)(temp & 0x00FF)
idx = idx + 1
END LOOP
IF pack_size is odd ? **THEN**
temp = Sn_RX_FIFOR
str[idx] = (INT8)(temp >> 8)
END IF
Sn_CR = 0x0040 (RECV command)
	INT16 pack_size, idx,temp
	INT8 str[5], dummy
	pack_size = Sn_RX_FIFOR0
	pack_size = (pack_size << 8)
	pack_size = pack_size + Sn_RX_FIFOR1
	idx = 0
	LOOP pack_size/2
	str[idx] = Sn_RX_FIFOR0
	idx = idx + 1
	str[idx] = Sn_RX_FIFOR1
	idx = idx + 1
	END LOOP
	IF pack_size is odd ? **THEN**
	str[idx] = Sn_RX_FIFOR0
	dummy = Sn_RX_FIFOR1
	END IF
	Sn_CR1 = 0x40 (RECV command)

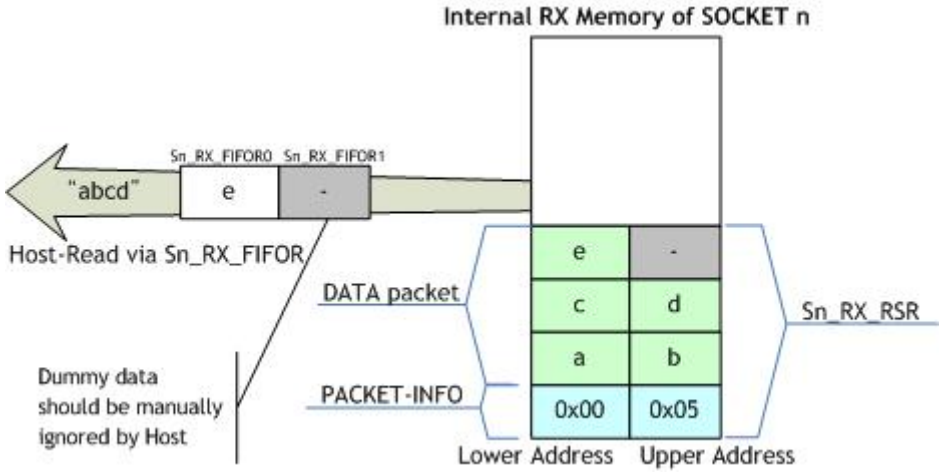


Fig 7. Access to Internal RX Memory

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.

Refer to http://www.ieee.org/, http://standards.ieee.org/regauth/oui/index.shtml

- STEP 3 : Allocation internal TX/RX memory for SOCKETn

1. Defining internal TX/RX memory size (Refer to MTYPER)
2. Defining TX/RX memory of SOCKETn (Refer to TMSR & RMSR)

W5300 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.

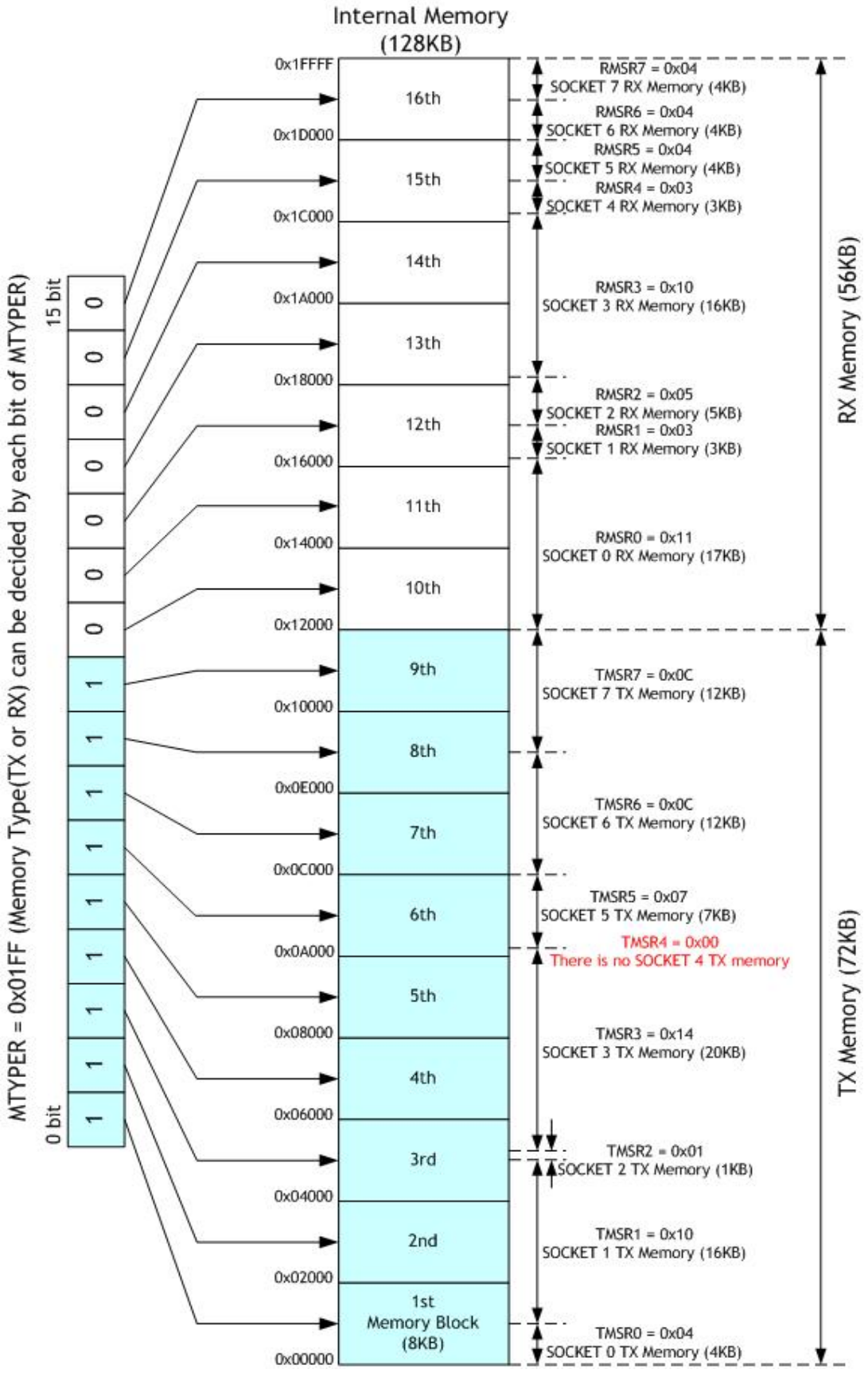


Fig 8. Allocation Internal TX/RX memory of SOCKETn

When the 3 initialization-steps are successfully processed, W5300 is available for data communication through Ethernet. From this time, W5300 can transmit the Ping-reply to the Ping-request packet (Auto-ping-reply)

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).

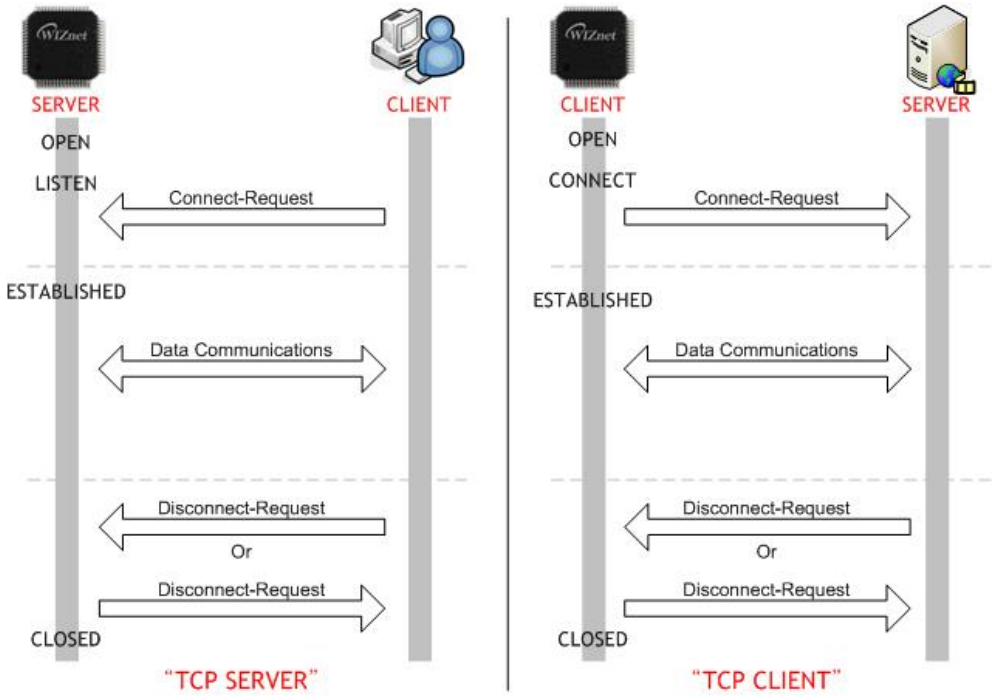


Fig 9. "TCP SERVER" & "TCP CLIENT"

5.2.1.1 TCP SERVER

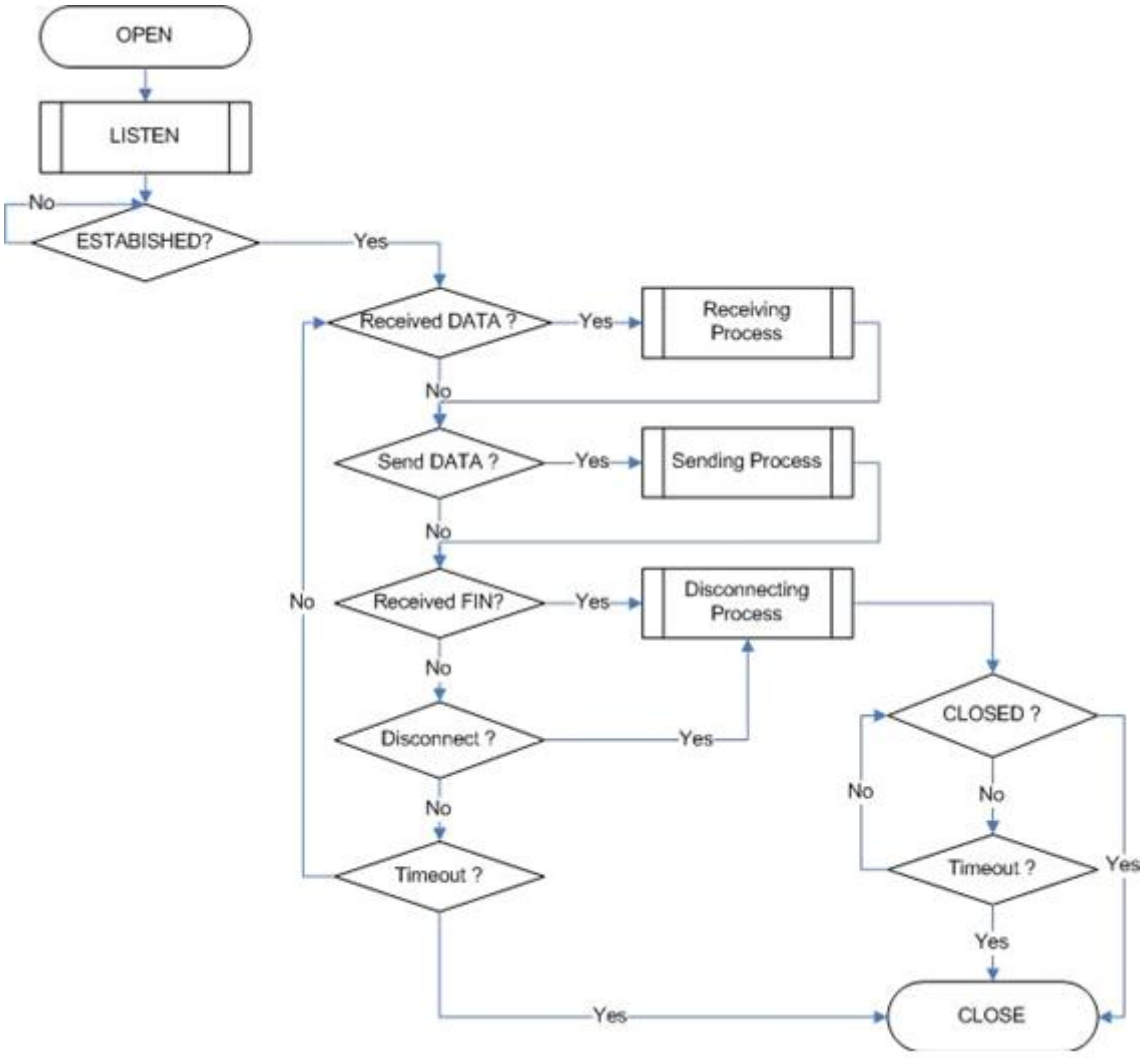


Fig 10. "TCP SERVER" Operation Flow

■ 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.

```
{
START:
   Sn_MR = 0x0001;                  /* sets TCP mode */
   Sn_PORTR = source_port;          /* sets source port number */
   Sn_CR = OPEN;                              /* sets OPEN command */
   /* wait until Sn_SSR is changed to SOCK_INIT */
   if (Sn_SSR != SOCK_INIT) Sn_CR = CLOSE; goto START;
}
```

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.

```
{
    /* 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 :
{
    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 :
{
```

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

■ ESTABLISHED : Received Data ?

It checks if TCP data is received from the peer.

```
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;
}
```

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.

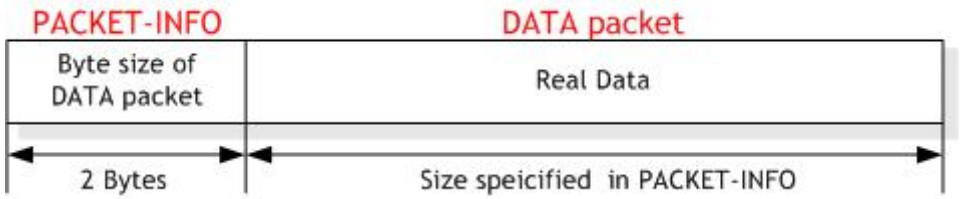


Fig 11. The received TCP data format

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.

```
{
    /* 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 */
}
```

■ ESTABLISHED : Send DATA ? / Sending Process

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.

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.

```
{
    /* 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.

```
{
    /* 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_{TO}. When the host wants for SOCKETn to be just closed without disconnect-process, it is also used.

```
{
    /* 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”.

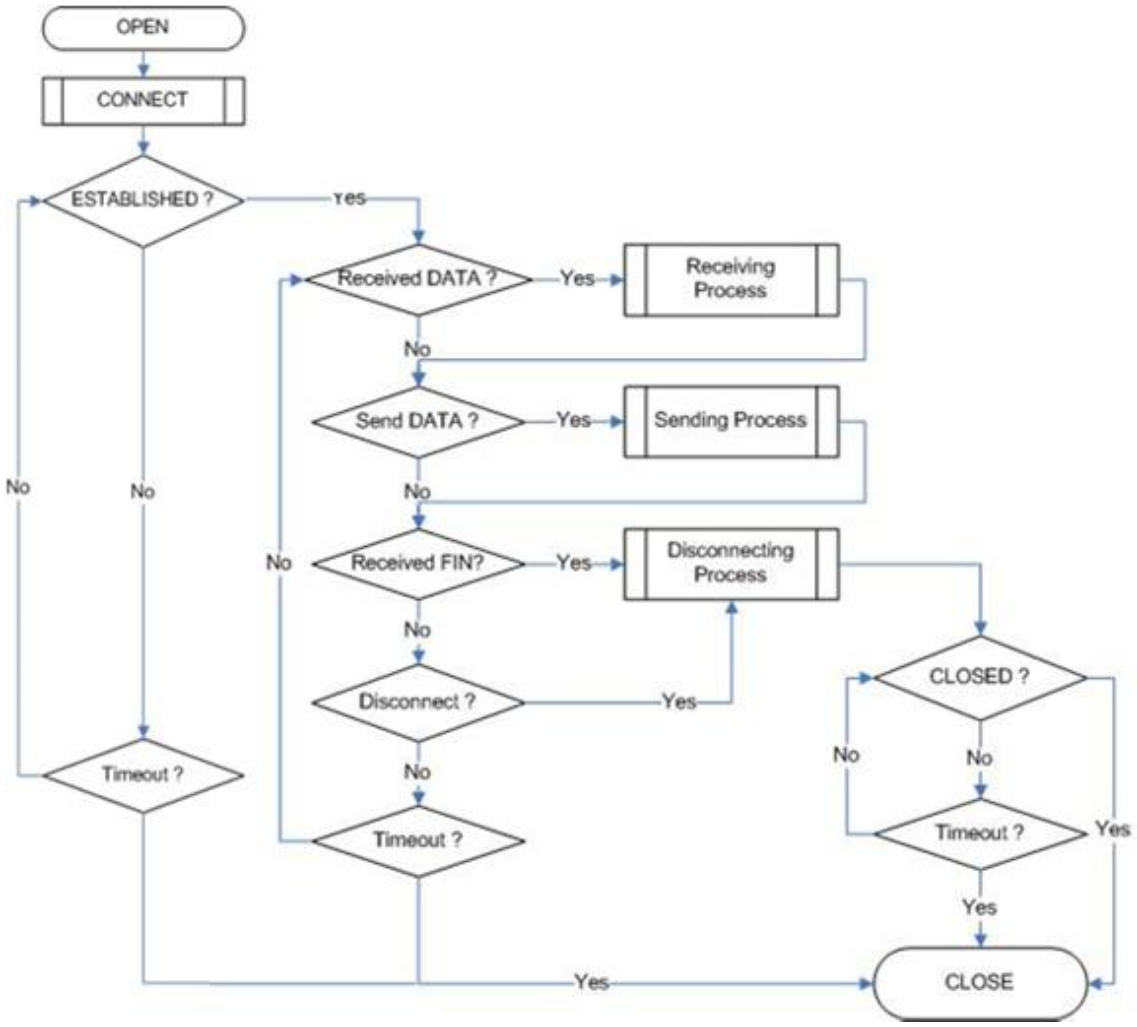


Fig 12. "TCP CLIENT" Operation Flow

■ 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.

```
{
    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:

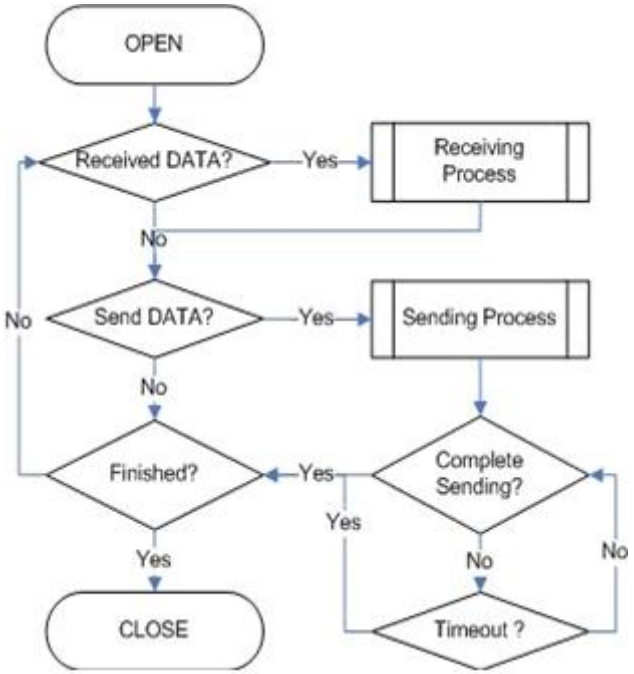


Fig 13. UDP Operation Flow

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_{TO} can occur in the ARP-process to acquire destination hardware address of A, B, C. It is not possible to transmit the data to the peer of ARP_{TO}. 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_{TO} 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.

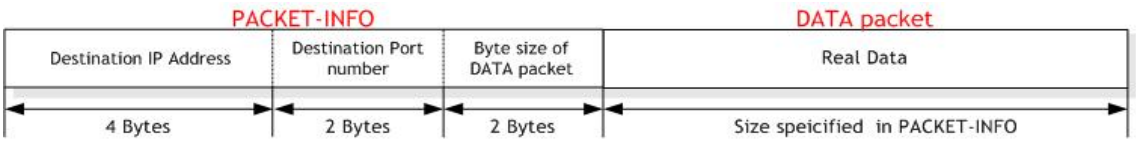


Fig 14. The received UDP data format

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.

```
{
    /* 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".

```
{
    /* 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.

```
{
  /* check SEND command completion */
  while(Sn_IR(SENDOK)==‘0’)   /* wait interrupt of SEND completion */
  {
    /* check ARPTO */
    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.

```
{
    /* 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".

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.

```
{
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_DIRP3 = 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_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

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 */
```

```
  Sn_IR(SENDOK) = '1';          /* clear interrupt of SEND completion */
}
```

- Finished? / SOCKET Close

Refer to "5.2.2.1 Unicast & Broadcast".

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.

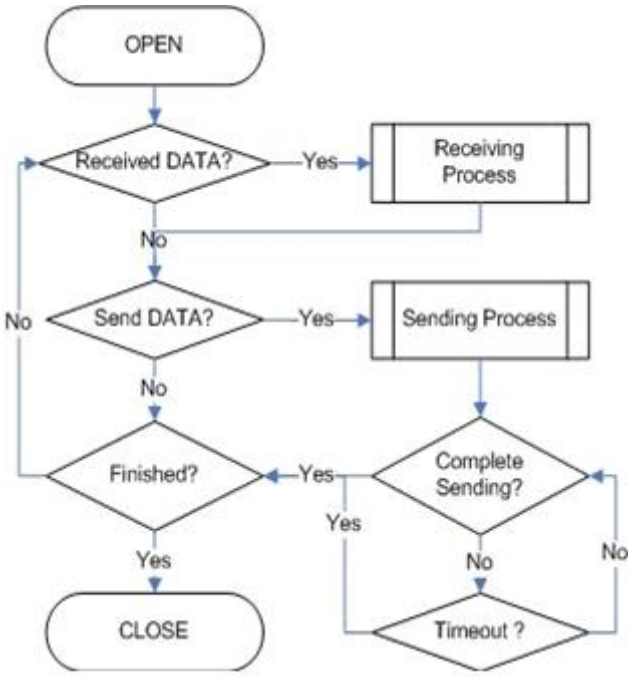


Fig 15. IPRAW Operation Flow

- 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. */
    Sn_PROTO = protocol_num;
    /* sets IP raw mode */
    Sn_MR = 0x03;
    /* sets OPEN command */
    Sn_CR = OPEN;
   /* wait until Sn_SSR is changed to SOCK_IPRAW */
    if (Sn_SSR != SOCK_IPRAW) Sn_CR = 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.

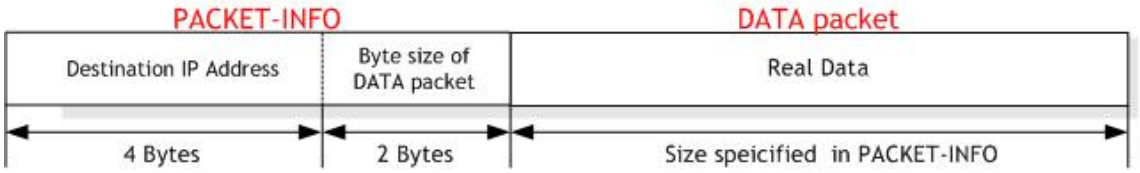


Fig 16. The received IPRAW data format

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.

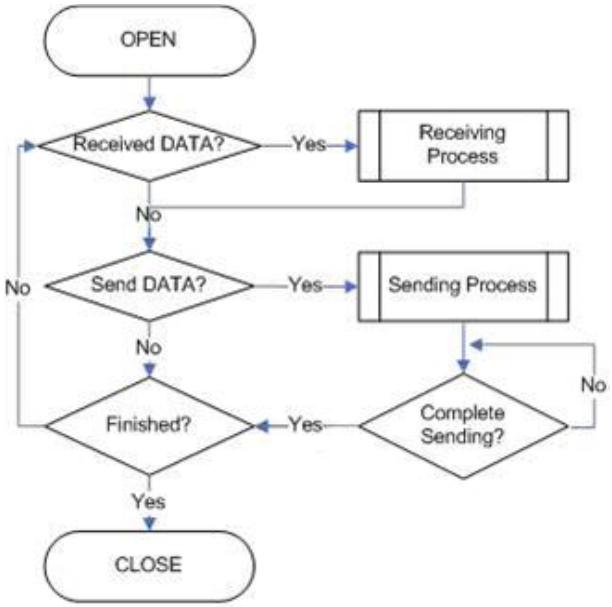


Fig 17. MACRAW Operation Flow

■ 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.

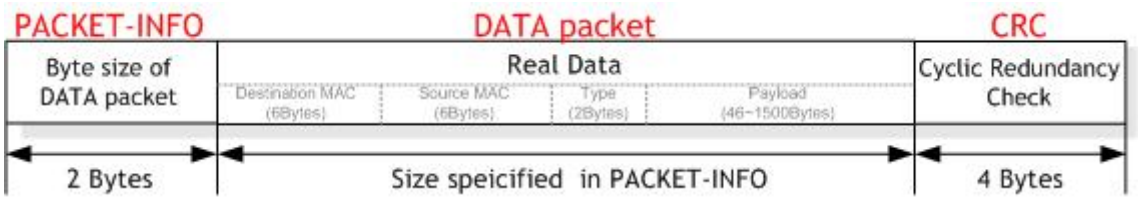


Fig 18. The received MACRAW data format

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.

The CRC of MACRAW data should be read by the host through S0_RX_FIFOR and ignored.

```
{
  /* 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;

  /* 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.

```
{
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*/
          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”.

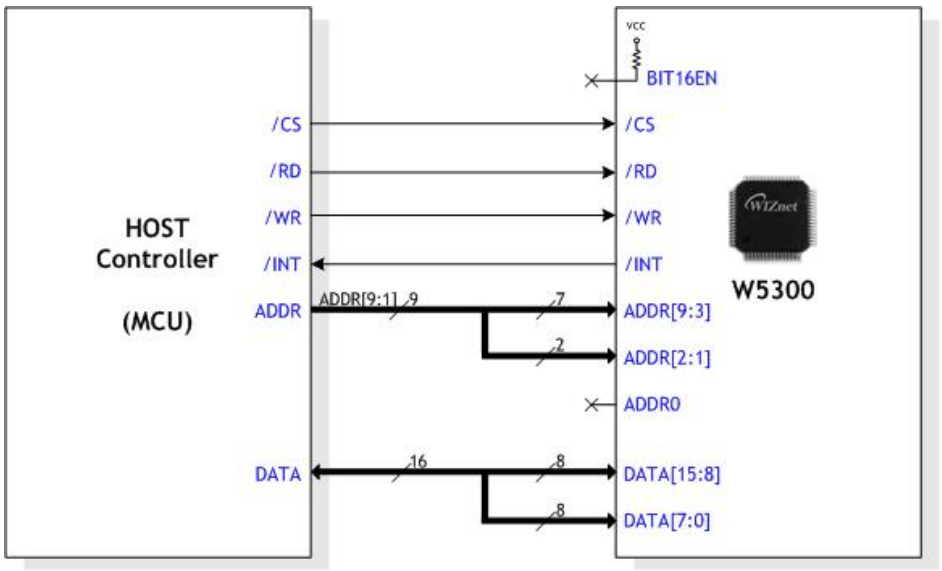
6. External Interface

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

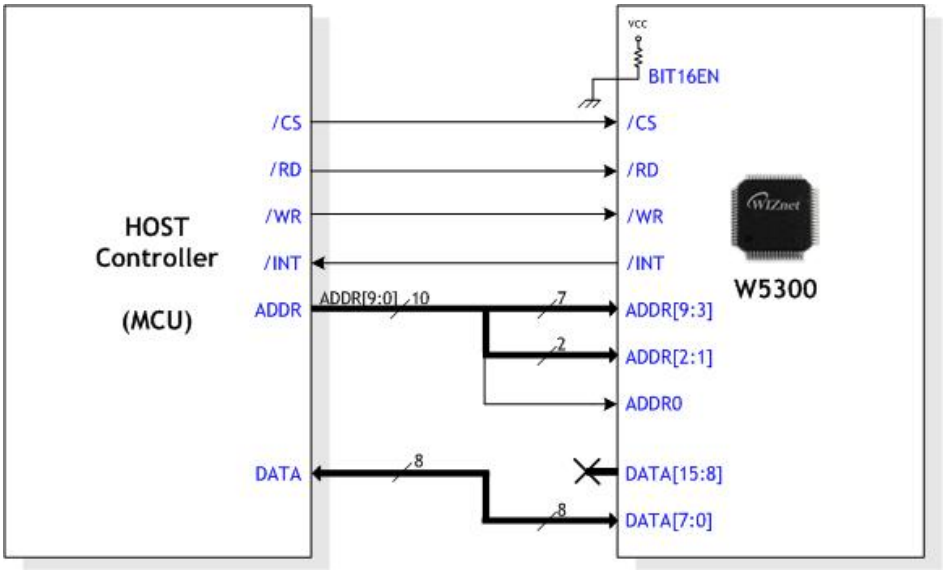
6.1.1 16 Bit Data Bus Width

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

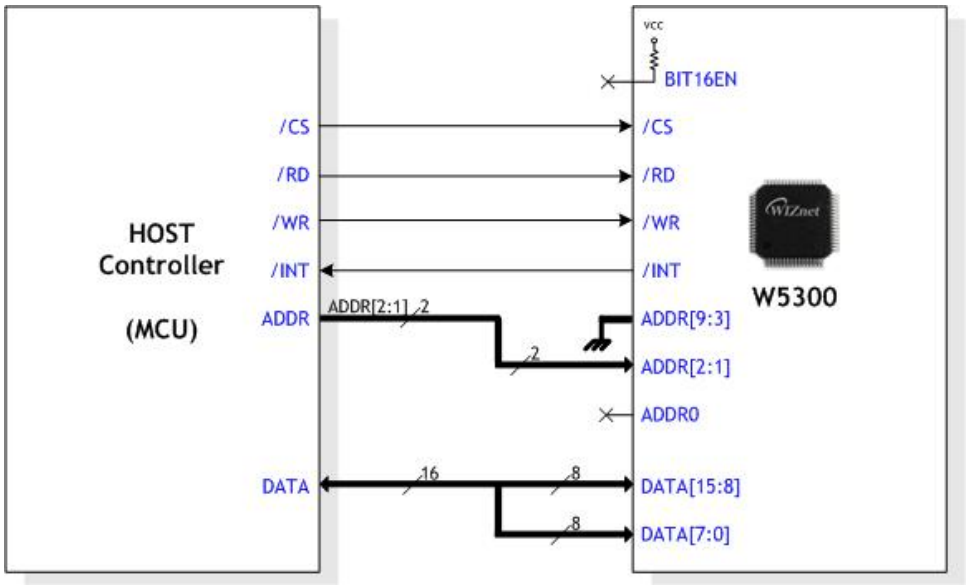
In the case of using an 8bit data bus width, ADDR[9:0] is used. 'BIT16EN' should be logical LOW (ground). Let the unused DATA[15:8] float.



6.2 Indirect Address Mode

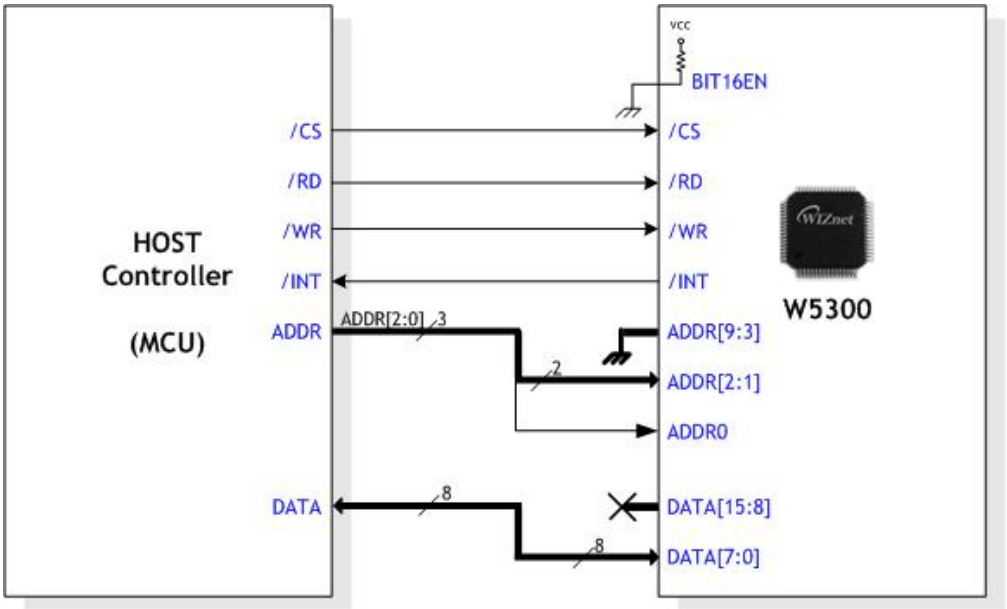
6.2.1 16 Bit Data Bus Width

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 un-used 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".

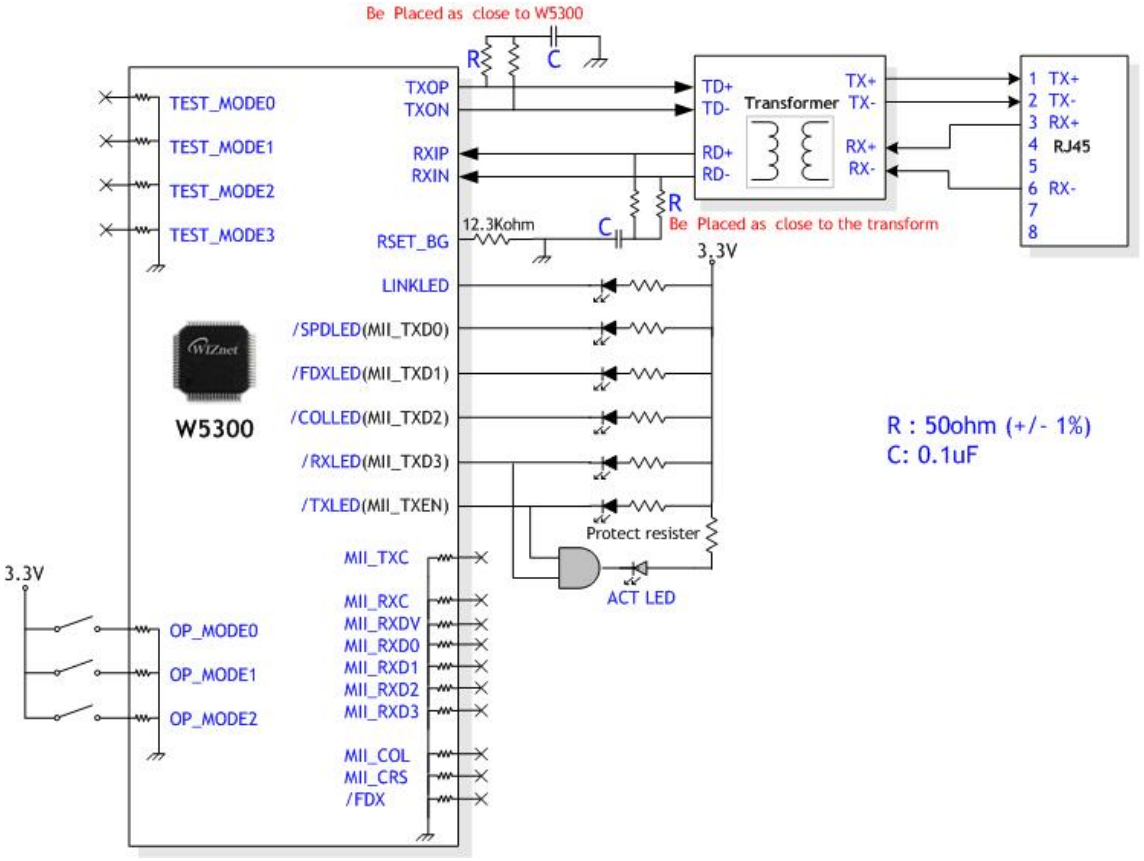


Fig 19. Internal PHY & LED Signals

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.

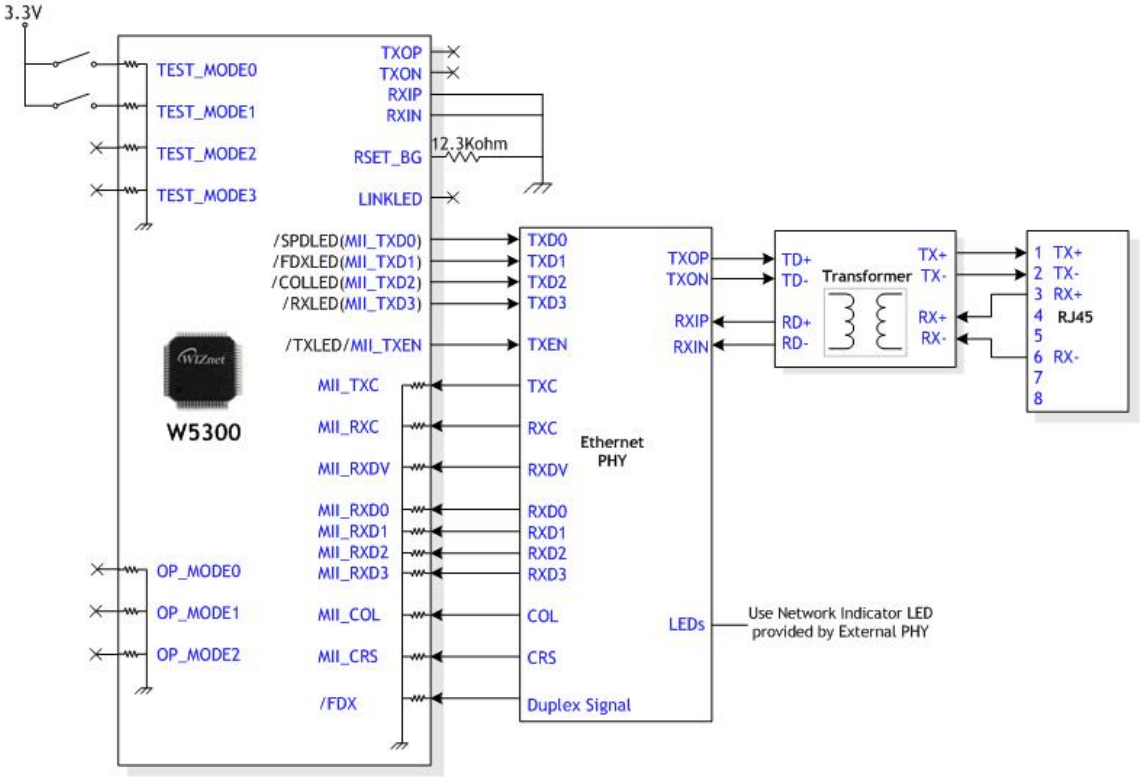


Fig 20. External PHY Interface with MII

7. Electrical Specifications

Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	-0.5 to 3.6	V
V_{IN}	DC input voltage	-0.5 to 5.5 (5V tolerant)	V
V_{OUT}	DC output voltage	-0.5 to 3.6	V
I_{IN}	DC input current	±5	mA
I_{OUT}	DC output current	2 to 8	mA
T_{OP}	Operating temperature	-40 to 80 [1]	°C
T_{STG}	Storage temperature	-55 to 125	°C

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

[1] : Please refer our Qualification Report in our website(search in http://www.wiznet.co.kr or http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/KOLAS_Test_Report_QRTC-D-0808-169_W5300[0].pdf)

DC Characteristics

Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
V_{DD}	DC Supply voltage	Junction temperature is from -55°C to 125°C	3.0	3.3	3.6	V
V_{IH}	High level input voltage		2.0		5.5	V
V_{IL}	Low level input voltage		- 0.5		0.8	V
V_{OH}	High level output voltage	I_{OH} = 2 ~ 16 mA	2.4			V
V_{OL}	Low level output voltage	I_{OL} = -2 ~ -12 mA			0.4	V
I_I	Input Current	$V_{IN} = V_{DD}$			±5	μA
I_O	Output Current	$V_{OUT} = V_{DD}$	2		8	mA

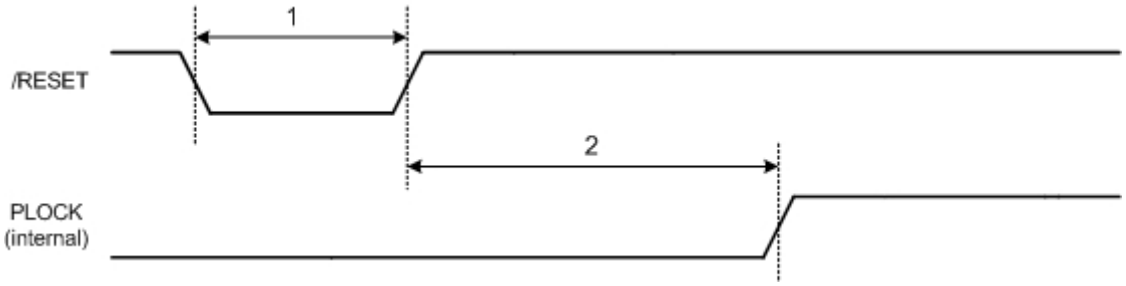
POWER DISSIPATION

Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
P_{IA}	Power consumption when using the auto-negotiation	Vcc 3.3V Temperature 25°C	-	180	250	mA

	of internal PHY mode					
P_{IM}	Power consumption when using manual configuration of internal PHY mode	Vcc 3.3V Temperature 25°C	-	175	210	mA
P_E	Power consumption when using external PHY mode	Vcc 3.3V Temperature 25°C		65	150	mA

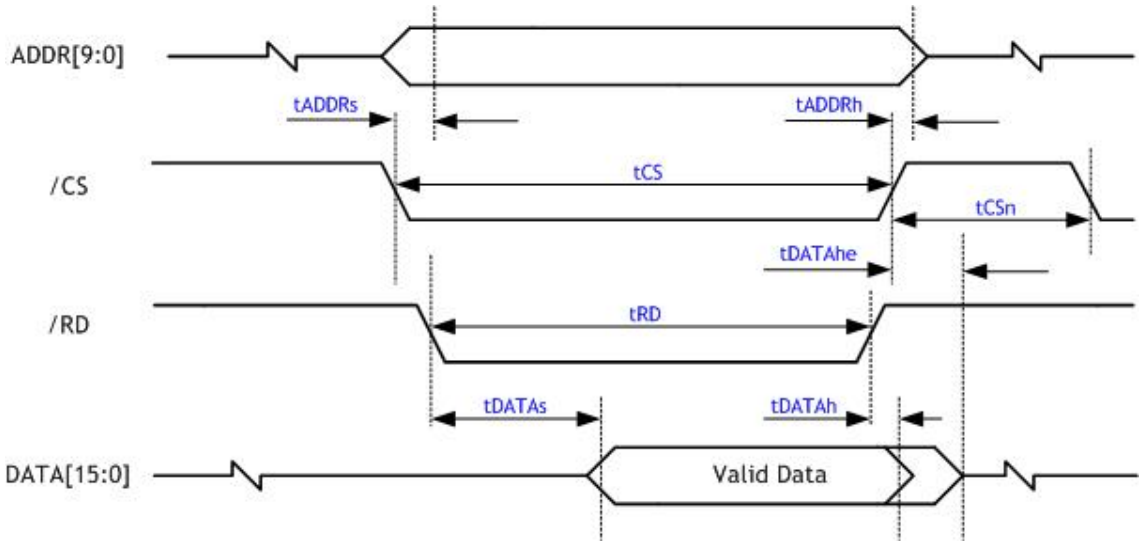
AC Characteristics

Reset Timing



	Description	Min	Max
1	Reset Cycle Time	2 us	-
2	PLL Lock-in Time	50 us	10 ms

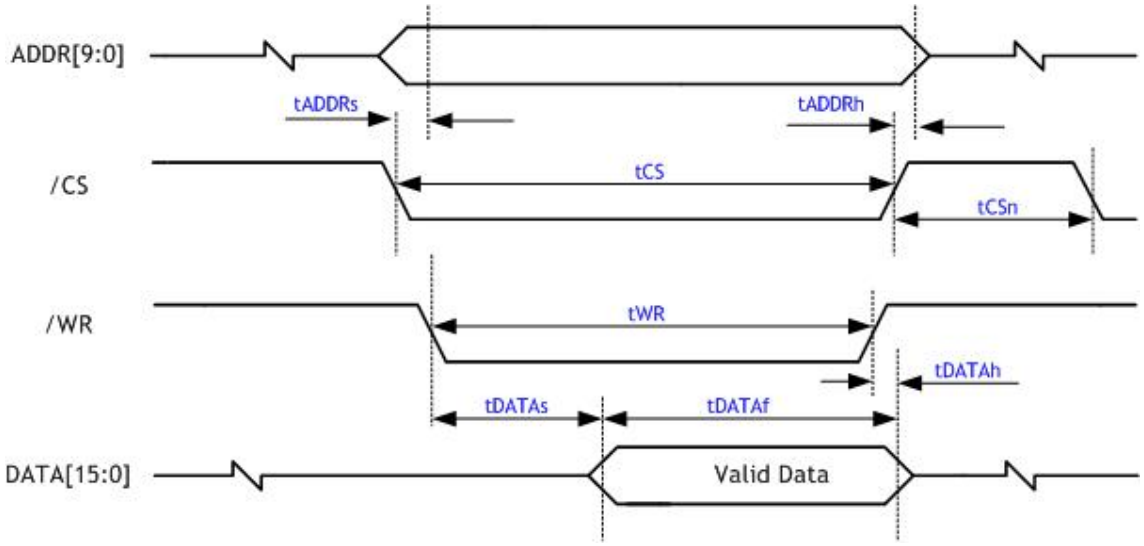
Register READ Timing



Description		Min	Max
tADDRs	Address Setup Time after /CS and /RD low	-	7 ns
tADDRh	Address Hold Time after /CS or /RD high	-	-
tCS	/CS Low Time	65 ns	-
tCSn	/CS Next Assert Time	28 ns	-
tRD	/RC Low Time	65 ns	-
tDATAs	DATA Setup Time after /RD low	42 ns	-
tDATAh	DATA Hold Time after /RD and /CS high	-	7 ns
tDATAhe	DATA Hold Extension Time after /CS high	-	2XPLL_CLK

<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



Description		Min	Max
tADDRs	Address Setup Time after /CS and /WR low	-	7 ns
tADDRh	Address Hold Time after /CS or /RD high	-	-
tCS	/CS low Time	50 ns	-
tCSn	/CS next Assert Time	28 ns	
tWR	/WR low time	50 ns	
tDATAs	Data Setup Time after /WR low	7 ns	7ns + 7XPLL_CLK

tDATAf	Data Fetch Time	14 ns	tWR-tDATAs
tDATAh	Data Hold Time after /WR high	7 ns	-

<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'.

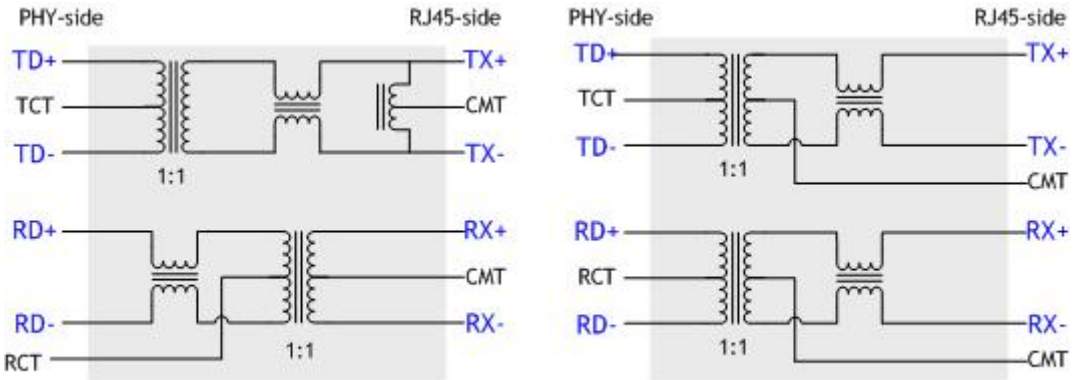
In order to fetch the valid data at this time, the host should guarantee 'tDATAh'.

Crystal Characteristics

Parameter	Range
Frequency	25 MHz
Frequency Tolerance (at 25℃)	±30 ppm
Shunt Capacitance	7pF Max
Drive Level	1 ~ 500uW (100uW typical)
Load Capacitance	27pF
Aging (at 25℃)	±3ppm / year Max

Transformer Characteristics

Parameter	Transmit End	Receive End
Turn Ratio	1:1	1:1
Inductance	350 uH	350 uH



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

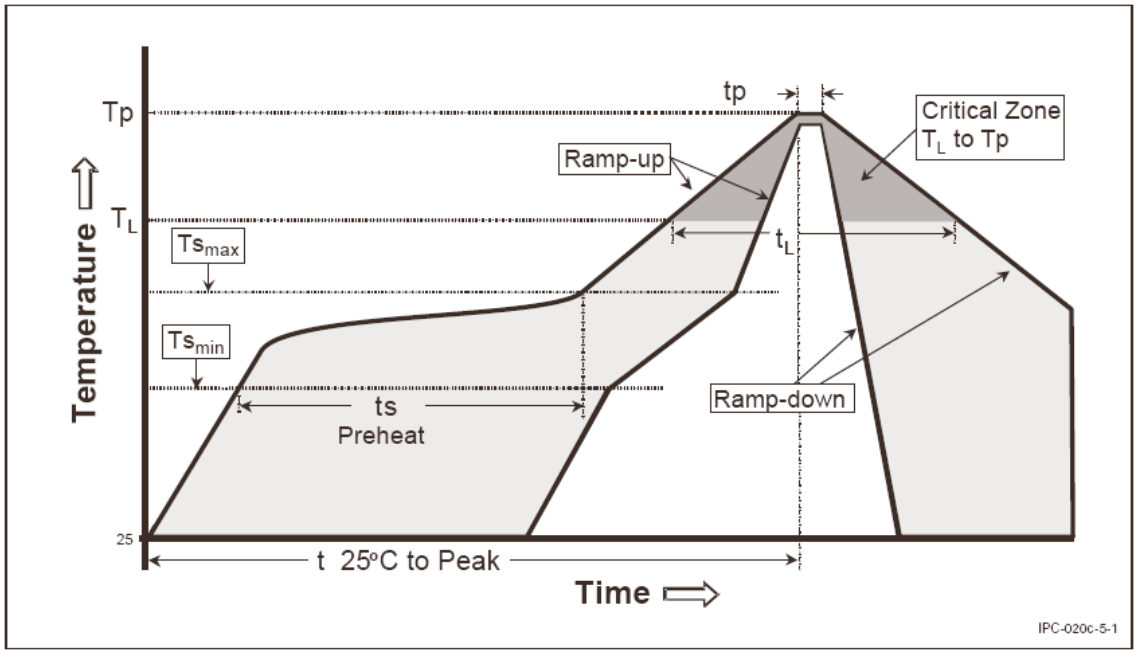
In case of using External PHY mode, use the transform which is suitable for external PHY specification.

8. IR Reflow Temperature Profile (Lead-Free)

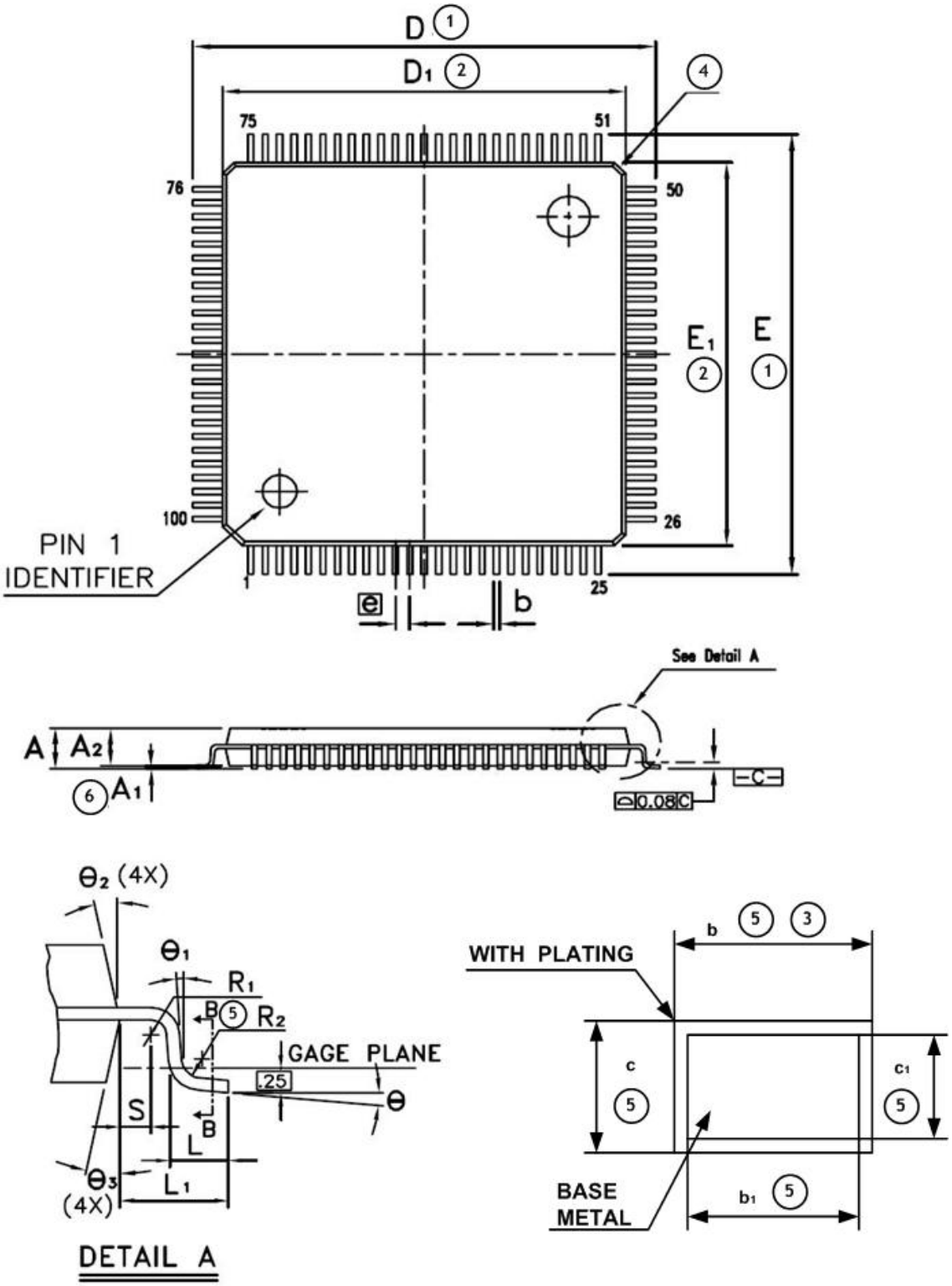
Moisture Sensitivity Level : 3

Dry Pack Required : Yes

Average Ramp-Up Rate (Ts_{max} to Tp)	3° C/second max.
Preheat – Temperature Min (Ts_{min}) – Temperature Max (Ts_{max}) – Time (ts_{min} to ts_{max})	 150 °C 200 °C 60-180 seconds
Time maintained above: – Temperature (TL) – Time (tL)	 217 °C 60-150 seconds
Peak/Classification Temperature (Tp)	260 + 0 °C
Time within 5 °C of actual Peak Temperature (tp)	20-40 seconds
Ramp-Down Rate	6 °C/second max.
Time 25 °C to Peak Temperature	8 minutes max.



9. Package Descriptions



SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	-	-	1.60	-	-	0.063
A_1	0.05	-	0.15	0.002	-	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.17	0.22	0.27	0.007	0.009	0.011
b_1	0.17	0.20	0.23	0.007	0.008	0.009
c	0.09	-	0.20	0.004	-	0.008
c_1	0.09	-	0.16	0.004	-	0.006
D	15.85	16.00	16.15	0.624	0.630	0.636
D_1	13.90	14.00	14.10	0.547	0.551	0.555
E	15.85	16.00	16.15	0.624	0.630	0.636
E_1	13.90	14.00	14.10	0.547	0.551	0.555
e		0.50 BSC			0.020 BSC	
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1		1.00 REF			0.039 REF	
R_1	0.08	-	-	0.003	-	-
R_2	0.08	-	0.20	0.003	-	0.008
S	0.20	-	-	0.008	-	-
θ	0°	3.5°	7°	0°	3.5°	7°
$θ_1$	0°	-	-	0°	-	-
$θ_2$		12° TYP			12° TYP	
$θ_3$		12° TYP			12° TYP	

<NOTE> ① To be determined at seating plane -C-.

② Dimensions 'D_1' and 'E_1' do not include mold protrusion.
D_1' and 'E_1' are maxium plastic body size dimensions including mold mismatch.

③ Dimension 'b' does not include dambar protrusion.
Dambar can not be located on the lower radius or the foot.

④ Exact shape of each corner is optional

⑤ These Dimensions apply to the flat section of the lead between 0.10mm and 0.25mm from the lead tip.

⑥ A_1 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.

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View W5300 on WIN SOURCE
- WIZnet Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management