i2Chip W3100

■ Description

The i2Chip W3100 is an LSI of hardware protocol stack that provides an easy, low-cost solution for high-speed Internet connectivity for digital devices by allowing simple installation of TCP/IP stack in the hardware.

The W3100 offers system designers a quick, easy way to add Ethernet networking functionality to any products. Implementing this LSI into a system can completely offload Internet connectivity and processing standard protocols from the system, thereby significantly reducing the software development cost.

The W3100 contains TCP/IP Protocol Stacks such as TCP, UDP, IP, DHCP, ARP and ICMP protocols, as well as Ethernet protocols such as Data Link Control and MAC protocol.

The W3100 offers a socket API (Application Programming Interface) that is similar to the windows socket API. The chip offers Intel and Motorola MCU (8051, i386, 6811 tested) bus interface for upper-layer and supports standard MII interface for under-layer Ethernet.

The W3100 can be applied to handheld devices including Internet phones, VoIP SOC chips, Internet MP3 players, handheld medical devices, LAN cards for Web servers, cellular phones and many other non-portable electronic devices such as large consumer electronic equipment.

■ Features

- Hardware Internet protocols included:
  TCP, IP Ver.4, UDP, ICMP, ARP, DHCP
- Hardware Ethernet protocols included:
  DLC, MAC
- Supports 4 independent connections simultaneously
- Protocol processing speed: full-duplex 4~5 Mbps
- Intel/Motorola MCU bus Interface
- Standard MII Interface for under-layer physical chip
- Socket API support for easy application programming
- Supports full-duplex mode
- Internal 24Kbytes Dual-port SRAM for data buffer
- 0.35 μm CMOS technology
- Wide operating voltage:
  3.3V internal operation, 5V tolerant 3.3V IOs
- Small 64 Pin LQFP Package

■ Block Diagram

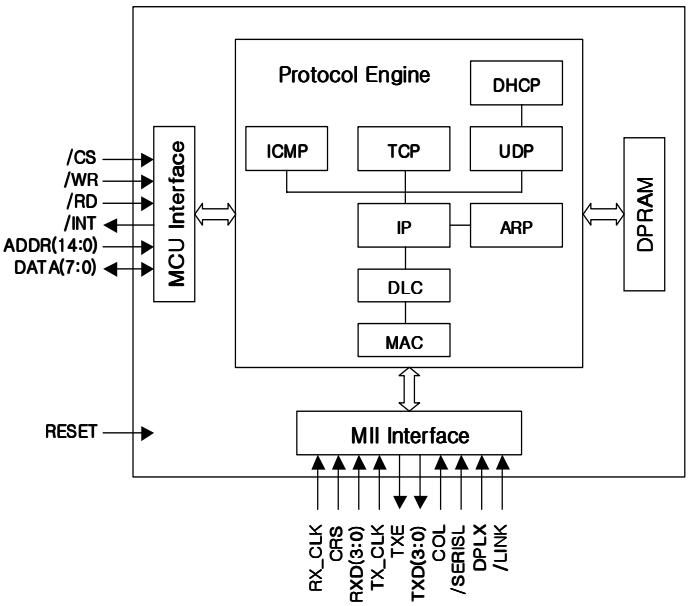
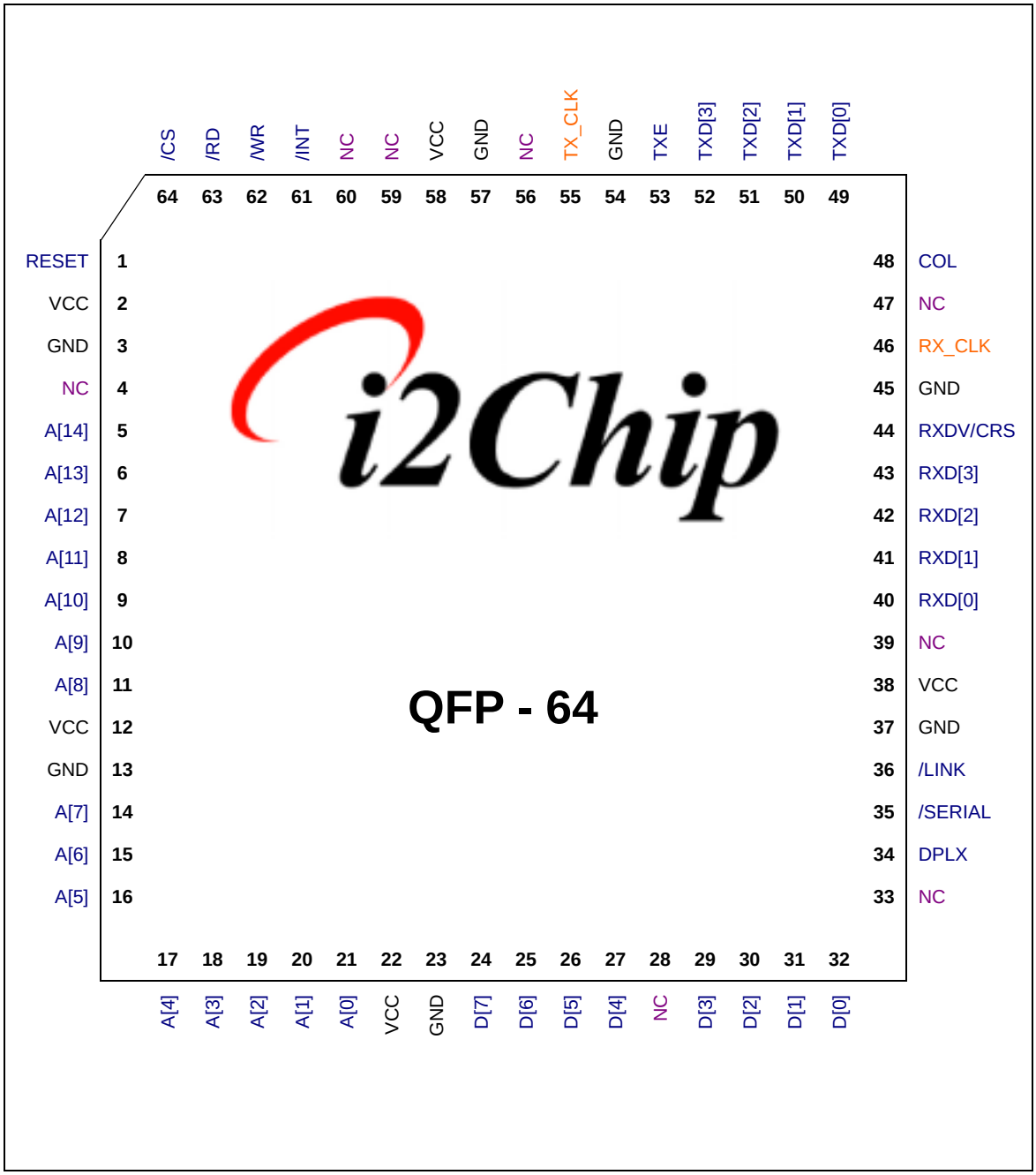


Table of Contents

■ Pin Assignment

Figure 1 : 64-Pin QFP Pin Assignments



■ Signal Description

Table 1: W3100 MII Signal Description

PIN#	Signal	I/O	Description
52 51 50 49	TXD[3] TXD[2] TXD[1] TXD[0]	O	**TRANSMIT DATA:** Nibble/Serial NRZ data output to the ENDEC that is valid on the rising edge of TX_CLK. In Serial mode, the TXD[0] pin is used as the serial data pin, and TXD[3:1] are ignored.
53	TXE	O	**TRANSMIT ENABLE:** becomes active when the first nibble/serial data of the packet is valid on TXD[3:0] and goes low after the last nibble/serial data of the packet is clocked out of TXD[3:0]. This signal connects directly to the ENDEC (PHY device). This signal is active high.
55	TX_CLK	I	**TRANSMIT CLOCK:** TX_CLK is sourced by the PHY. TX_CLK is 2.5 MHz in 10BASE-T Nibble mode, and 25 MHz in 100BASE-T Nibble mode. The PHY 25Mhz clock input is required to be synchronized with main MCU clock.
43 42 41 40	RXD[3] RXD[2] RXD[1] RXD[0]	I	**RECEIVE DATA:** Nibble wide receive data (synchronous to RX_CLK) that must be driven on the falling edge of RX_CLK. In 10 Mb/s serial mode, the RXD[0] pin is used as the data input pin which is also clocked in on the falling edge of RX_CLK. During 10 Mb/s serial mode, RXD[3:1] pins become don't cares.
44	RXDV/CRS	I	**CARRIER SENSE:** signal provided by the ENDEC and indicates that carrier is present. This signal is active high.
46	RX_CLK	I	**RECEIVE CLOCK:** Re-synchronized clock from the ENDEC and indicates that carrier is present.
48	COL	I	**COLLISION DETECT:** becomes active when a collision has been detected in Half Duplex modes. This signal is asynchronous, active high and ignored during full-duplex operation.

Table 2: W3100 Bus Interface Signal Description

PIN#	Signal	I/O	Description
5-11 14-21	A[14-8] A[7-0]	I	**ADDRESS PINS**
24-27 29-32	D[7-4] D[3-0]	I/O	**DATA PINS**
61	/INT	O	**INTERRUPT:** Indicates that the W3100 requires CPU attention after reception or transmission. The interrupt is cleared by writing to the ISR (Interrupt Status Register). All interrupts are maskable by writing IMG (Interrupt Mask Register). This signal is active low.
64	/CS	I	**CHIP SELECT:** This signal is active low.
62	/WR	I	**WRITE ENABLE:** This signal is active low.
63	/RD	I	**READ ENABLE:** This signal is active low.

Table 3: W3100 Miscellaneous Signal Description

PIN#	Signal	I/O	Description
1	RESET	I	**RESET:** Active High input that initializes or reinitializes the W3100. Asserting this pin will force a reset process to occur which will result in all internal registers reinitializing to their default states as specified for each bit in section x.x, and all strapping options are reinitialized. Refer to section x.x for further detail regarding reset.
36	/LINK	I	**LINK:** This is the signal generated by Ethernet PHY to indicate the PHY is connected to the Ethernet HUB device or other peer device. This is active low and internally full-down. W3100 can knows the status of physical line connection with this /LINK input. If /LINK is high, W3100 interprets the physical line is disconnected. It result in TCP timeout and connection close. In special PHY case, LINK signal is vary in time, then you can bound it to ground.
35	/SERIAL	I	**10BASE-T SERIAL/NIBBLE SELECT:** With the selection of this active low input, transmit and receive data are exchanged serially at a 10 MHz clock rate on the least significant bits of the nibble-wide MII data buses, pins TXD[0] and RXD[0], respectively. This mode is intended for use with the W3100 connected to a PHY using a 10 Mb/s serial interface. There is an internal pull-up resister for this pin. If this pin is left floated externally, then the device will be configured to normal mode. This pin must be externally pulled low (typically x kΩ) in order to configure the W3100 for Serial MII operation.
34	DPLX	I	**FULL/HALF DUPLEX SELECT:** This input pin selects Half/Full Duplex operation. There is an internal pull-up resister for this pin. If this pin is left floated externally, then the device will be configured to Full Duplex mode. This pin must be externally pulled low (typically x kΩ) in order to configure the W3100 for Half Duplex operation. 0 = Full Duplex 1 = Half Duplex
4, 28, 33, 39, 47, 56, 59, 60	NC		**NO CONNECT:** These pins are reserved for future use. Leave them unconnected (floating).

Table 4: W3100 Power Supply Signal Description

PIN#	Signal	I/O	Description
2, 12, 22, 38, 58	VCC		**POSITIVE 3.3V SUPPLY PINS**
3, 13, 23,	GND		**NEGATIVE (GROUND) SUPPLY PINS:** A decoupling capacitor is

37, 45, 54, 57			suggested to be connected between the Vcc and GND pins

■ Description of Functions

Overview

The i2Chip W3100 is an LSI of hardware protocol stack that provides an easy, low-cost solution for high-speed Internet connectivity for digital devices by allowing simple installation of TCP/IP stack in the hardware. The W3100 offers a socket API (Application Programming Interface) that is similar to the windows socket API. Table 5-1 contains an overview of the functions provided by the W3100.

Table 5: Overview of W3100 Functions

Functions	Operation
Sysinit	Initializes i2chip W3100
Socket	Creates a socket
Connect	Attempts to open a TCP connection
Listen	Accepts a TCP connection
Send	Transmits data through a TCP socket
Recv	Receives data through a TCP socket.
sendto	Transmits data through a UDP or RAW socket
recvfrom	Receives data through a UDP or RAW socket
Close	Closes a socket

Through the initialization process, the W3100 enters into an operating mode, which is largely categorized into 3 different modes depending on the type of protocols applied. When TCP protocol is in use, data can be transmitted or received after the connecting process is complete. TCP protocol provides active mode, which attempts to connect, and passive mode, which stands by for connection, and data transfer is made after the connection is made. Socket can be closed after completion of the data transfer. UDP and RAW protocols allow data transfer following the initialization, without the connecting process, and the socket can be closed after completion of the data transfer. The W3100 provides 4 sockets, each of which can be used as a different protocol.

Initialization

Internal register must be initialized for normal operation of the W3100. The internal register includes MAC address and IP address for the W3100, as well as the subnet mask and gateway to set the subnet.

Table 6: Overview of Initialization Functions

Functions	Operation
Setgateway	Sets gateway
setsubmask	Sets subnet mask
SetIP	Sets IP address of the W3100
setMACAddr	Sets MAC address for the W3100

Example)

```
struct sockaddr_in server;

server.sin_addr.S_un.S_un_b.s_b1 = 211;
server.sin_addr.S_un.S_un_b.s_b2 = 171;
server.sin_addr.S_un.S_un_b.s_b3 = 137;
server.sin_addr.S_un.S_un_b.s_b4 = 1;                    /* gateway address */
setgateway((struct sockaddr *) &server);

server.sin_addr.S_un.S_un_b.s_b1 = 255;
server.sin_addr.S_un.S_un_b.s_b2 = 255;
server.sin_addr.S_un.S_un_b.s_b3 = 255;
server.sin_addr.S_un.S_un_b.s_b4 = 0;                    /* subnet mask */
setsubmask((struct sockaddr *) &server);

server.sin_addr.S_un.S_un_b.s_b1 = 211;
server.sin_addr.S_un.S_un_b.s_b2 = 171;
server.sin_addr.S_un.S_un_b.s_b3 = 137;
server.sin_addr.S_un.S_un_b.s_b4 = 27;                   /* ip address */
setIP((struct sockaddr *) &server);

MACAddr[0] = 0x10;
MACAddr[1] = 0x20;
MACAddr[2] = 0x30;
MACAddr[3] = 0x40;
MACAddr[4] = 0x50;
MACAddr[5] = 0x60;                                       /* MAC address */
setMACAddr(MACAddr);

SysInit();                                               /* initialization */
```

TCP Protocol

TCP (transmission control protocol) is the most common transport layer protocol used on the Internet. TCP protocol is further described in RFC793. TCP is built on top of Internet Protocol (IP) and is nearly always found in the combination of TCP/IP (TCP over IP). It adds reliable communication, flow-control, multiplexing and connection oriented communication. It also provides full-duplex, process-to-process connections as well as reliable connection-oriented or stream-oriented connections over the Internet. After initializing the W3100, a socket is created and connection is made. Data transfer is allowed after the connection is made, and the socket can be closed after the data transfer is complete.

1) Socket creation

Socket function: creates a socket, and the port number to be used with the SOCK_STREAM is transferred to the parameter of the socket() function when TCP protocol is in use. The data is transmitted or received by the socket handle created during this process.

Prototype	socket_descriptor socket(UCHAR protocol, UNIT port)

Input	protocol: SOCK_STREAM when TCP protocol is in use port: port number used at local address
Return	socket_descriptor: the socket handle created by the socket function when no error has occurred, or –1 in case error has occurred

2) Connection setting

Connect function: attempts connect with the designated port of the designated host

Prototype	ret_val connect(SOCKET socket, struct sockaddr * dest)
Input	socket: socket handle to be used dest: IP address and port number of the host to be connected
Return	ret_val: 1 when connected, or 0 in case error has occurred

Example)

```
SOCKET fd;
struct sockaddr_in server;

fd = socket(SOCK_STREAM, 5000);

server.sin_addr.S_un.S_un_b.s_b1 = 211;
server.sin_addr.S_un.S_un_b.s_b2 = 171;
server.sin_addr.S_un.S_un_b.s_b3 = 137;
server.sin_addr.S_un.S_un_b.s_b4 = 26;
server.sin_port = 3000;

connect(fd, (struct sockaddr *) &server);
```

Listen function: receives TCP connection coming into the designated port when creating a socket

Prototype	ret_val listen(SOCKET socket, struct sockaddr * dest)
Input	socket: socket handle to be used
Output	dest: IP address and port number of the connected host
Return	ret_val: 1 when connected, or 0 in case error has occurred

Example:

```
fd = socket(SOCK_STREAM, 5000);
listen(fd, (struct sockaddr *) &server);
```

3) Data transmission, reception

Send function: as a function that transmits the data, it transfers the following to the parameter: socket descriptor, pointer for data buffer and the size of the data to be transmitted.

Prototype	UINT send(SOCKET s, UCHAR far * buf, UINT len)

Input	socket: socket handle to be used buf: pointer for the data to be transmitted len: size of the data to be transmitted
Return	Size of transmitted data

Example:

```
unsigned UCHAR far sock_buf[2048];
send(fd, sock_buf, 2048);
```

Recv function: as a function that receives the data, it transfers the following to the parameter: socket descriptor, pointer for data buffer and the size of the data.

Prototype	UINT recv(SOCKET s, UCHAR far * buf, UINT len)
Input	socket: socket handle to be used buf: pointer for the buffer where received data is to be saved len: size of the buffer where received data is to be saved
Return	Size of received data

Example:

```
unsigned UCHAR far sock_buf[2048];
int size;

size = recv(fd, sock_buf, 2048);
```

4) Socket return

Close function: closes the socket that was used when no further data is transmitted or received

Prototype	UINT close(SOCKET s)
Input	socket: socket handle
Return	1 when successfully closed, or 0 in case error has occurred

Example)

```
close(fd);
```

UDP Protocol

UDP (user datagram protocol) provides simple but unreliable datagram services. UDP protocol is further described in RFC768. It is a connectionless protocol which, like TCP, is layered on top of IP. UDP neither guarantees delivery nor does it require a connection. As a result, it is lightweight and efficient, but all error processing and retransmission must be taken care of by the application protocol. First, the socket is created for data transmission and reception, and the socket is closed after completion of the data transfer.

1) Socket creation

Socket function: uses SOCK_DGRAM to call on the socket function when UDP protocol is in use.

Example:

```
fd = socket(SOCK_DGRAM, 5000);
```

2) Transmitting and receiving data

Sendto function: used for transmitting data when UDP protocol is in use. Other than the socket handle, buffer point and buffer size used by the send function, information on the transmitting host is used as the element for the sendto function.

Prototype	UINT sendto(SOCKET s, const UCHAR far * buf, UINT len, struct sockaddr * dest)
Input	socket: socket handle to be used buf: pointer for the data to be transmitted len: size of the data to be transmitted dest: IP address and port number of the host to transmit the data
Return	Size of transmitted data

Example)

```
unsigned char sock_buf[1024];

fd = socket(SOCK_DGRAM, 3000);

dest.sin_addr.S_un.S_un_b.s_b1 = 211;
dest.sin_addr.S_un.S_un_b.s_b2 = 171;
dest.sin_addr.S_un.S_un_b.s_b3 = 137;
dest.sin_addr.S_un.S_un_b.s_b4 = 26;
dest.sin_port = 3000;

sendto(fd, sio_buf, 1024, (struct sockaddr *) &dest);
```

Recvfrom function: used for reception of data when UDP protocol is in use, it receives the data coming into the port designated while creating the socket. Information can be obtained on the host that transmitted the data.

Prototype	UINT recvfrom(SOCKET s, const UCHAR far * buf, UINT len, struct sockaddr * dest)
Input	socket: socket handle to be used buf: pointer for the buffer where transmitted data is to be saved len: size of the buffer where transmitted data is to be saved
output	dest: IP address and port number of the host that transmitted the data
Return	Size of transmitted data

Example)

```
unsigned char sock_buf[1024];
```

```
fd = socket(SOCK_DGRAM, 3000);
len = recvfrom(fd, &sock_buf, 1024, (struct sockaddr *) &server);
```

3) Socket return

Close function: closes the socket created by the UDP protocol.

RAW Mode

RAW Mode is used when using the IP protocol without using the TCP/UDP protocol. IP protocol must be set to use the RAW mode, and the value is 1 in case of ICMP (Internet Control Message Protocol). Sento and recvfrom functions used by UDP are used for data transmission and reception.

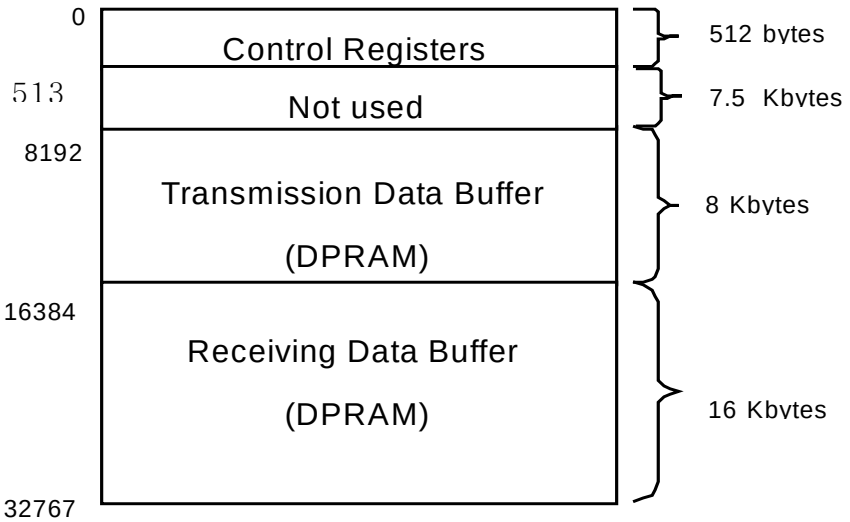
Prototype	UINT set_ip_protocol(UINT ip_protocol)
Input	ip_protocol: IP protocol
Return	1 when successfully closed, or 0 in case error has occurred

Example:

```
fd = socket(SOCK_RAWM, 5000);
set_ip_protocol(1);
```

■ Internal Memory and Registers

W3100 Top level Memory Map



W3100 uses 8bit x 512 address space for Control Registers and 8bit x 24K address space for Data Buffers;

- 0~255 : Space for Control Registers
- 256~512 : Space for Shadow Registers
- 513 ~ 8191 : Not used (This space can be used by other devices)
- 8192~16383 : Data Buffer for Transmission
- 16384~32767 : Data Buffer for Receiving

Table 7: W3100 Registers Map

Address	Register	Bit Definitions							
0x00	C0_Command	S/W Reset	Recv	Send	Close	Listen	Connect	Sock_Init	Sys_Init
0x01	C1_Command		Recv	Send	Close	Listen	Connect	Sock_Init	
0x02	C2_Command		Recv	Send	Close	Listen	Connect	Sock_Init	
0x03	C3_Command		Recv	Send	Close	Listen	Connect	Sock_Init	
0x04	C0_INT_Status		Recv_OK	Send_OK	Timeout	Closed	Established	SInit_OK	Init_OK
0x05	C1_INT_Status		Recv_OK	Send_OK	Timeout	Closed	Established	SInit_OK	
0x06	C2_INT_Status		Recv_OK	Send_OK	Timeout	Closed	Established	SInit_OK	
0x07	C3_INT_Status		Recv_OK	Send_OK	Timeout	Closed	Established	SInit_OK	
0x08	INT REG	C3R	C2R	C1R	C0R	C3	C2	C1	C0
0x09	INT Mask REG	IM_C3R	IM_C2R	IM_C1R	IM_C0R	IM_C3	IM_C2	IM_C1	IM_C0
0x0A – 0x0F	Reserved								
0x10 – 0x13	C0_Rx_Wr_Ptr	Channel 0 Rx Write Pointer							
0x14 – 0x17	C0_Rx_Rd_Ptr	Channel 0 Rx Read Pointer							
0x18 – 0x1B	C0_Tx_Ack_Ptr	Channel 0 Tx ACK Pointer							
0x1C – 0x1F	C1_Rx_Wr_Ptr	Channel 1 Rx Write Pointer							
0x20 – 0x23	C1_Rx_Rd_Ptr	Channel 1 Rx Read Pointer							
0x24 – 0x27	C1_Tx_Ack_Ptr	Channel 1 Tx ACK Pointer							
0x28 – 0x2B	C2_Rx_Wr_Ptr	Channel 2 Rx Write Pointer							
0x2C – 0x2F	C2_Rx_Rd_Ptr	Channel 2 Rx Read Pointer							
0x30 – 0x33	C2_Tx_Ack_Ptr	Channel 2 Tx ACK Pointer							
0x34 – 0x37	C3_Rx_Wr_Ptr	Channel 3 Rx Write Pointer							
0x38 – 0x3B	C3_Rx_Rd_Ptr	Channel 3 Rx Read Pointer							
0x3C – 0x3F	C3_Tx_Ack_Ptr	Channel 3 Tx ACK Pointer							
0x40 – 0x43	C0_Tx_Wr_Ptr	Channel 0 Tx Write Pointer							

Table 7: W3100 Registers Map (Continued)

Address	Register	Bit Definitions							
0x44 – 0x47	C0_Tx_Rd_Ptr	Channel 0 Tx Read Pointer							
0x48 – 0x4B	Reserved								
0x4C – 0x4F	C1_Tx_Wr_Ptr	Channel 1 Tx Write Pointer							
0x50 – 0x53	C1_Tx_Rd_Ptr	Channel 1 Tx Read Pointer							
0x54 – 0x57	Reserved								
0x58 – 0x5B	C2_Tx_Wr_Ptr	Channel 2 Tx Write Pointer							
0x5C – 0x5F	C2_Tx_Rd_Ptr	Channel 2 Tx Read Pointer							
0x60 – 0x63	Reserved								
0x64 – 0x67	C3_Tx_Wr_Ptr	Channel 3 Tx Write Pointer							
0x68 – 0x6B	C3_Tx_Rd_Ptr	Channel 3 Tx Read Pointer							
0x6C – 0x7F	Reserved								
0x80 – 0x83	Gateway Addr	Gateway Address							
0x84 – 0x87	Subnet Mask	Subnet Mask							
0x88 – 0x8D	Source HA	Source Hardware Address							
0x8E – 0x91	Source IP	Source IP Address							
0x92 – 0x93	Timeout Val	Timeout Value							
0x94	Timeout Count	Timeout Count							
0x95 – 0x9F	Reserved								
0xA0	C0_S_Status	Channel 0 Socket Status							
0xA1	C0_Opt_Protocol	Broadcast	NDTimeout	NDAck				Protocol	Protocol

Table 7: W3100 Registers Map(Continued)

Address	Register	Bit Definitions							
0xA2 – 0xA7	Reserved								
0xA8 – 0xAB	C0_Dest_IP	Channel 0 Destination IP Address							
0xAC – 0xAD	C0_Dest_Port	Channel 0 Destination Port							
0xAE – 0xAF	C0_Source_Port	Channel 0 Source Port							
0xB0	C0_IP_Protocol	Channel 0 IP Protocol for raw							
0xB1	C0_TOS	Channel 0 TOS							
0xB2 – 0xB3	C0_MSS	Channel 0 MSS							
0xB4 – 0xB7	Reserved								
0xB8	C1_S_Status	Channel 1 Socket Status							
0xB9	C1_Opt_Protocol	Broadcast	NDTimeout	NDAck				Protocol	Protocol
0xBA – 0xBF	Reserved								
0xC0 – 0xC3	C1_Dest_IP	Channel 1 Destination IP Address							
0xC4 – 0xC5	C1_Dest_Port	Channel 1 Destination Port							
0xC6 – 0xC7	C1_Source_Port	Channel 1 Source Port							
0xC8	C1_IP_Protocol	Channel 1 IP Protocol for raw							
0xC9	C1_TOS	Channel 1 TOS							
0xCA – 0xCB	C1_MSS	Channel 1 MSS							
0xCC – 0xCF	Reserved								
0xD0	C2_S_Status	Channel 2 Socket Status							
0xD1	C2_Opt_Protocol	Broadcast	NDTimeout	NDAck				Protocol	Protocol
0xD2 – 0xD7	Reserved								

Table 7: W3100 Registers Map(Continued)

Address	Register	Bit Definitions							
0xD8 – 0xDB	C2_Dest_IP	Channel 2 Destination IP Address							
0xDC – 0xDD	C2_Dest_Port	Channel 2 Destination Port							
0xDE – 0xDF	C2_Source_Port	Channel 2 Source Port							
0xE0	C2_IP_Protocol	Channel 2 IP Protocol for raw							
0xE1	C2_TOS	Channel 2 TOS							
0xE2 – 0xE3	C2_MSS	Channel 2 MSS							
0xE4 – 0xE7	Reserved								
0xE8	C3_S_Status	Channel 3 Socket Status							
0xE9	C3_Opt_Protocol	Broadcast	NDTimeout	NDAck				Protocol	Protocol
0xEA – 0xEF	Reserved								
0xF0 – 0xF3	C3_Dest_IP	Channel 3 Destination IP Address							
0xF4 – 0xF5	C3_Dest_Port	Channel 3 Destination Port							
0xF6 – 0xF7	C3_Source_Port	Channel 3 Source Port							
0xF8	C3_IP_Protocol	Channel 3 IP Protocol for raw							
0xF9	C3_TOS	Channel 3 TOS							
0xFA – 0xFB	C3_MSS	Channel 3 MSS							
0xFC – 0xFF	Reserved								

■ Register Definitions

Register sets are categorized into (i) control registers related to command, status and interrupt, (ii) system registers for gateway address, subnet mask, source IP, source HA (Hardware Address) and timeout value, and (iii) channel registers that control operation of each channel. R/W access to reserved register is not allowed, and also, writing on read-only register is not allowed.

Control Registers

Channel 0 Command Register(R/W, 0x00)

This register commands Channel 0 socket to initialize, connect, close, transmit and receive data. Sys_Init command is used to set the gateway, subnet mask, source IP and source HA. The same command is used to close the socket in all channels.

Sock_Init, connect, listen, close, send and recv are used for Channel 0 socket to initialize, connect, set, and close socket, and to transmit and receive data. Sock_Init command is used to open the relevant channel under TCP, UDP, RAW mode in accordance with the protocol value set by the Opt_Protocol register. S/W Reset is used by the uProcessor to initialize the internal setting value of the chip.

7	6	5	4	3	2	1	0
S/W Reset	Recv	Send	Close	Listen	Connect	Sock_Init	Sys_Init

Bit	Symbol	Description
D0	Sys_Init	Command to set Gateway, Subnet Mask, Source IP, Source HA.
D1	Sock_Init	Opens Channel 0 Socket after setting relevant Protocol according to the C0_Opt_Protocol REG.
D2	Connect	Command to operate Channel 0 Socket in Client Mode to set the connection to the Server.
D3	Listen	Command to standby for connection when Channel 0 Socket is in Server Mode
D4	Close	Command to close Channel 0 Socket and to close connection
D5	Send	Command to transmit data for Channel 0 Socket
D6	Recv	Command to receive data for Channel 0 Socket
D7	S/W Reset	Command to reset S/W

Channel 1, 2, 3 Command Register(R/W, 0x01, 0x02, 0x03)

This register commands each socket of Channels 1, 2, and 3 to initialize, connect, close and transmit and receive data. Sock_Init, connect, listen, close, send and recv are used for relevant socket to initialize, connect, set and close the socket, and to transmit and receive data.

7	6	5	4	3	2	1	0
	Recv	Send	Close	Listen	Connect	Sock_Init	

Bit	Symbol	Description
D0		Reserved
D1	Sock_Init	Opens relevant Channel Socket after setting relevant Protocol according to the Cx_Opt_Protocol REG

D2	Connect	Command to operate the relevant Channel Socket in Client Mode to set the connection to the Server.
D3	Listen	Command to standby for connection when relevant Channel Socket is in Server Mode
D4	Close	Command to close the relevant Channel Socket and to close connection
D5	Send	Command to transmit data for the relevant Channel Socket
D6	Recv	Command to receive data for the relevant Channel Socket
D7		Reserved

Channel 0 Interrupt Status Register(R, 0x04)

This register is used to indicate the outcome of the command for channel 0 socket. The register is cleared by read operation. Init_OK indicates completion of Sys_Init command. Established indicates completion of connect, listen commands to set connection. Timeout indicates that timeout has occurred while processing connect, listen or send commands. SInit_OK, closed, send_OK and recv_OK indicate completion of Sock_Init, close, send, and recv commands, respectively.

7	6	5	4	3	2	1	0
	Recv_OK	Send_OK	Timeout	Closed	Established	SInit_OK	Init_OK

Bit	Symbol	Description
D0	Init_OK	INT Status Bit indicating completion of Sys_Init Command
D1	SInit_OK	INT Status Bit indicating completion of Sock_Init Command for Channel 0 Socket
D2	Established	INT Status Bit indicating completion of setting of connection for Channel 0 Socket
D3	Closed	INT Status Bit indicating completion of closing of connection for Channel 0 Socket
D4	Timeout	INT Status Bit indicating timeout has occurred during setting of connection for Channel 0 Socket or data transfer
D5	Send_OK	INT Status Bit indicating completion of Send Command for Channel 0 Socket
D6	Recv_OK	INT Status Bit indicating completion of Recv Command for Channel 0 Socket
D7		Reserved

Channel 1, 2, 3 Interrupt Status Register(R, 0x05, 0x06, 0x07)

This register is used to indicate the outcome of the command for each socket of channels 1, 2 and 3. The register is cleared by read operation. Established indicates completion of connect, listen commands to set connection. Timeout indicates that timeout has occurred while processing connect, listen or send commands. SInit_OK, closed, send_OK and recv_OK indicate completion of Sock_Init, close, send, and recv commands, respectively.

7	6	5	4	3	2	1	0
	Recv_OK	Send_OK	Timeout	Closed	Established	SInit_OK	

Bit	Symbol	Description
D0		Reserved
D1	SInit_OK	INT Status Bit indicating completion of Sock_Init Command for the relevant Channel

		Socket
D2	Established	INT Status Bit indicating completion of setting of connection for the relevant Channel Socket
D3	Closed	INT Status Bit indicating completion of closing of connection for the relevant Channel Socket
D4	Timeout	INT Status Bit indicating timeout has occurred during setting of connection for the relevant Channel Socket or data transfer
D5	Send_OK	INT Status Bit indicating completion of Send Command for the relevant Channel Socket
D6	Recv_OK	INT Status Bit indicating completion of Recv Command for the relevant Channel Socket
D7		Reserved

Interrupt Register(R/W, 0x08)

This register is used to categorize the channel when interrupt has occurred. C0, C1, C2 and C3 Bit are used to indicate that interrupt has occurred at channels 0, 1, 2 and 3, respectively. uProcessor assesses the channel interrupt status register of the relevant channel to recognize that interrupt has occurred. C0R, C1R, C2R and C3R Bit indicate reception of data at channels 0, 1, 2 and 3, respectively. Interrupt signal can be cleared by writing '1' in the register.

7	6	5	4	3	2	1	0
C3R	C2R	C1R	C0R	C3	C2	C1	C0

Bit	Symbol	Description
D0	C0	Interrupt indication of socket 0
D1	C1	Interrupt indication of socket 1
D2	C2	Interrupt indication of socket 2
D3	C3	Interrupt indication of socket 3
D4	C0R	Data received indication of Socket 0
D5	C1R	Data received indication of Socket 1
D6	C2R	Data received indication of Socket 2
D7	C3R	Data received indication of Socket 3

Interrupt Mask Register(R/W, 0x09)

This register can mask the interrupt caused to occur by each bit of the relevant interrupt register. Interrupt can occur when the relevant bit of the interrupt register is set as '1'.

7	6	5	4	3	2	1	0
IM_C3R	IM_C2R	IM_C1R	IM_C0R	IM_C3	IM_C2	IM_C1	IM_C0

Bit	Symbol	Description
D0	IM_C0	1 : INT Enable for Channel 0 Socket 0 : INT Disable for Channel 0 Socket
D1	IM_C1	1 : INT Enable for Channel 1 Socket 0 : INT Disable for Channel 1 Socket
D2	IM_C2	1 : INT Enable for Channel 2 Socket 0 : INT Disable for Channel 2 Socket
D3	IM_C3	1 : INT Enable for Channel 3 Socket 0 : INT Disable for Channel 3 Socket
D4	IM_C0R	1 : INT Enable for data reception for Channel 0 Socket 0 : INT Disable for data reception for Channel 0 Socket
D5	IM_C1R	1 : INT Enable for data reception for Channel 1 Socket 0 : INT Disable for data reception for Channel 1 Socket
D6	IM_C2R	1 : INT Enable for data reception for Channel 2 Socket 0 : INT Disable for data reception for Channel 2 Socket
D7	IM_C3R	1 : INT Enable for data reception for Channel 3 Socket 0 : INT Disable for data reception for Channel 3 Socket

System Registers

Gateway Address Register(R/W, 0x80 – 0x83)

Register for setting the default gateway address to be used by the system, it must be set before starting the Sys_Init command.

Subnet Mask Register(R/W, 0x84 – 0x87)

Register for setting subnet mask to be used by the system, it must be set before starting the Sys_Init command.

Source HA Register(R/W, 0x88 – 0x8D)

Register for setting HA to be used by the system, it must be set before starting the Sys_Init command.

Source IP Register(R/W, 0x8E – 0x91)

Register for setting the IP to be used by the system, it must be set before starting the Sys_Init command.

Timeout Value Register(R/W, 0x92 – 0x93)

Register for the user to set the timeout value.

Formula of timeout value :

TR : Timeout Register

TC : Timeout Counter

TR * 50us = start timeout second

Total timeout value until give-up = (TR * 50us) *(2^{TC}-1)

So, first timeout start with TR * 50us initial start value,

and then doubles the initial value and increment the timeout number when timeout value is expired. When the timeout number equals TC, then give up and reset the connection.

Example)

Give-up Timeout(Sec)	TC	TR
10	10	195
30	10	586
60	10	1173
120	10	2346

Timeout Count Register(R/W, 0x94)

Register for designating the number of retry when timeout occurs, it creates timeout interrupt when number of timeout occurred exceeds the number of retry.

Pointer Registers.

In order to read the pointer registers, relevant pointer of the shadow registers needs to be read in advance. Regarding write, no access is needed for the shadow register.

Shadow Registers	Address	해당 Pointer Registers
C0_Sd_Rx_Wr_Ptr	0x1E0	C0_ Rx_Wr_Ptr
C0_Sd_Rx_Rd_Ptr	0x1E1	C0_ Rx_Rd_Ptr
C0_Sd_Tx_Ack_Ptr	0x1E2	C0_ Tx_Ack_Ptr
C1_Sd_Rx_Wr_Ptr	0x1E3	C1_ Rx_Wr_Ptr
C1_Sd_Rx_Rd_Ptr	0x1E4	C1_ Rx_Rd_Ptr
C1_Sd_Tx_Ack_Ptr	0x1E5	C1_ Tx_Ack_Ptr
C2_Sd_Rx_Wr_Ptr	0x1E6	C2_ Rx_Wr_Ptr
C2_Sd_Rx_Rd_Ptr	0x1E7	C2_ Rx_Rd_Ptr
C2_Sd_Tx_Ack_Ptr	0x1E8	C2_ Tx_Ack_Ptr
C3_Sd_Rx_Wr_Ptr	0x1E9	C3_ Rx_Wr_Ptr
C3_Sd_Rx_Rd_Ptr	0x1EA	C3_ Rx_Rd_Ptr
C3_Sd_Tx_Ack_Ptr	0x1EB	C3_ Tx_Ack_Ptr
C0_Sd_Tx_Wr_Ptr	0x1F0	C0_ Tx_Wr_Ptr
C0_Sd_Tx_Rd_Ptr	0x1F1	C0_ Tx_Rd_Ptr
C1_Sd_Tx_Wr_Ptr	0x1F3	C1_ Tx_Wr_Ptr
C1_Sd_Tx_Rd_Ptr	0x1F4	C1_ Tx_Rd_Ptr
C2_Sd_Tx_Wr_Ptr	0x1F6	C2_ Tx_Wr_Ptr
C2_Sd_Tx_Rd_Ptr	0x1F7	C2_ Tx_Rd_Ptr
C3_Sd_Tx_Wr_Ptr	0x1F9	C3_ Tx_Wr_Ptr
C3_Sd_Tx_Rd_Ptr	0x1FA	C3_ Tx_Rd_Ptr

Rx Write Pointer Register(R, C0 : 0x10 – 0x13, C1 : 0x1C – 0x1F, C2 : 0x28 – 0x2B, C3 : 0x34 – 0x37)

This register exists for each channel, and it indicates the end pointer of the received data when the data is being received. uProcessor receives and processes the data from Rx read pointer to Rx writer pointer of the relevant channel.

Rx Read Pointer Register(R/W, C0 : 0x14 – 0x17, C1 : 0x20 – 0x23, C2 : 0x2C – 0x2F, C3 : 0x38 – 0x3B)

This register exists for each channel, and it indicates the start pointer of the received data when the data is being received. For the uProcessor, when processing of the received data is complete, the recv command needs to be made after the write by the data pointer that completed the processing of the Rx read pointer.

Tx Write Pointer Register(R/W, C0 : 0x40 – 0x43, C1 : 0x4C – 0x4F, C2 : 0x58 – 0x5B, C3 : 0x64 – 0x67)

This register exists for each channel, and it indicates the end pointer of the data to be transmitted when the data is being transmitted. uProcessor starts the write from the Tx write pointer of the relevant channel for the data to be transmitted and updates the Tx write pointer with a new value after copying the data. Finally, the data is transmitted by providing the send command.

Tx Read Pointer Register(R, C0 : 0x44 – 0x47, C1 : 0x50 – 0x53, C2 : 0x5C – 0x5F, C3 : 0x68 – 0x6B)

This register exists for each channel, and it indicates the current working pointer of the data to be transmitted when the data is being transmitted. This register, which is used internally within the chip, indicates the pointer to start transmitting when the transmission is being ordered by the send command.

Tx Ack Pointer Register(R, C0 : 0x18 – 0x1B, C1 : 0x24 – 0x27, C2 : 0x30 – 0x33, C3 : 0x3C – 0x3F)

This register exists for each channel, and it indicates the start pointer of the data to be transmitted when the data is being transmitted. Driver uses this pointer and the Tx write pointer to calculate the free size of the Tx buffer. In other words, the difference between the value of Tx write pointer and Tx Ack pointer is applicable to the size of buffer currently in use.

Channel Registers

Socket State Register(R, C0 : 0xA0, C1 : 0x B8, C2 : 0x D0, C3 : 0x E8)

Indicates the socket status of the relevant channel.

Value	Status	Indication
0x00	SOCK_CLOSED	Socket is not in use
0x01	SOCK_ARPSENT	ARP Request is sent, standing by for Reply
0x02	SOCK_LISTEN	Operating in Passive Mode, standing by for setting of connection to the Client
0x03	SOCK_SYNSENT	When operating in Active Mode, standby for SYN,ACK Packet after transmitting SYN Packet for setting of connection
0x04	SOCK_SYNSENT_ACK	In Active Mode, transmits SYN,ACK Packet after receiving ACK Packet and completes connection setting
0x05	SOCK_SYNRECV	In Passive Mode, under Listen status, SYN,ACK Packet is

		being transmitted after receiving SYN Packet from the opposing Client
0x06	SOCK_ESTABLISHED	Connection setting is complete in Active, Passive Modes
0x07	SOCK_CLOSE_WAIT	Connection is being closed
0X08	SOCK_LAST_ACK	Connection is being closed
0X09	SOCK_FIN_WAIT1	Connection is being closed
0X0A	SOCK_CLOSE_WAIT_ACK	Connection is being closed
0X0B	SOCK_CLOSING	Connection is being closed
0X0C	SOCK_TIME_WAIT	Connection is being closed
0X0D	SOCK_RESET	Connection is being closed after receiving Reset Packet from Peer
0X0E	SOCK_INIT	Initialization of Socket
0X0F	SOCK_UDP	Relevant Channel is being operated in UDP Mode
0X10	SOCK_RAW	Relevant Channel is being operated in RAW Mode
0X11	SOCK_UDP_ARPSENT	Standing by for Reply after ARP Request Packet is transmitted to the Destination to transmit UDP.
0X12	SOCK_UDP_DATA	Data is sent in UDP or RAW Mode
0X13	SOCK_SYNSENT_RST	Transmission of RESET Packet is caused by operating error of the Peer while setting the connection in Active Mode

Socket Option and Protocol Register(R/W, C0 : 0xA1, C1 : 0x B9, C2 : 0x D1, C3 : 0x E9)

Socket Option or Protocol is set for the relevant Channel.

7	6	5	4	3	2	1	0
Broadcast	NDTimeout	NDAck				Protocol	Protocol

Bit	Symbol	Description
D0, D1	Protocol	Operation of the relevant Channel is set in TCP, UDP or RAW Mode Value / Indication 00 Closed 01 SOCK_STREAM(TCP) 10 SOCK_DGRAM(UDP) 11 SOCK_RAW(RAW)
D5	NDAck	No Delayed ACK '0' : Delayed ACK is used. '1' : Delayed ACK is not in use. ACK is transmitted immediately when Data Packet is received.
D6	NDTimeout	No Dynamic Timeout '0' : Dynamic Timeout is used to configure the Timeout value during the Operation regardless of the set value. '1' : Operated by using the Timeout value as set by the Timeout Value.

D7	Broadcast	Broadcasting Packet is received in RAW mode.

Destination IP Register(R/W, C0 : 0xA8 – 0xAB, C1 : 0xC0 – 0xC3, C2 : 0xD8 – 0xDB, C3 : 0xF0 – 0xF3)

Used to set the destination IP address for connection setting, it must be set before starting the connect command when operated in active mode. When in passive mode, IP of the peer is set internally after the connection is set.

Destination Port Register(R/W, C0 : 0xAC – 0xAD, C1 : 0xC4 – 0xC5, C2 : 0xDC – 0xDD, C3 : 0xF4 – 0xF5)

Used to set the destination port for connection setting, it must be set before starting the connect command when operated in active mode. When in passive mode, IP of the peer is set internally after the connection is set.

Source Port Register(R/W, C0 : 0xAE – 0xAF, C1 : 0xC6 – 0xC7, C2 : 0xDE – 0xDF, C3 : 0xF6 – 0xF7)

Used to set the source port for the relevant channel, it must be set before starting the Sock_Init command.

IP Protocol for RAW Register(R/W, C0 : 0xB0, C1 : 0xC8, C2 : 0xE0, C3 : 0xF8)

Register to be set at the IP protocol for the protocol field of the IP header when operated in RAW mode, it must be set before starting the Sock_Init command.

TOS Register(R/W, C0 : 0xB1, C1 : 0xC9, C2 : 0xE1, C3 : 0xF9)

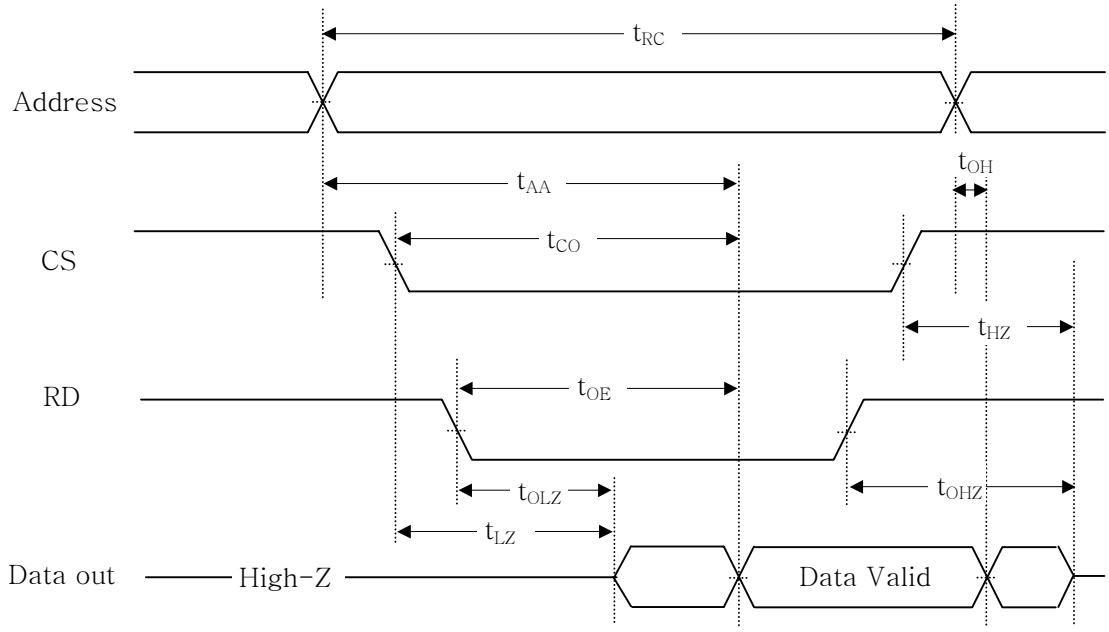
Register to be set at the TOS (type of service) field of the IP header, it must be set before starting the Sock_Init command.

MSS Register(R, C0 : B2 – 0xB3, C1 : 0xCA – 0xCB, C2 : 0xE2 – 0xE3, C3 : 0xFA – 0xFB)

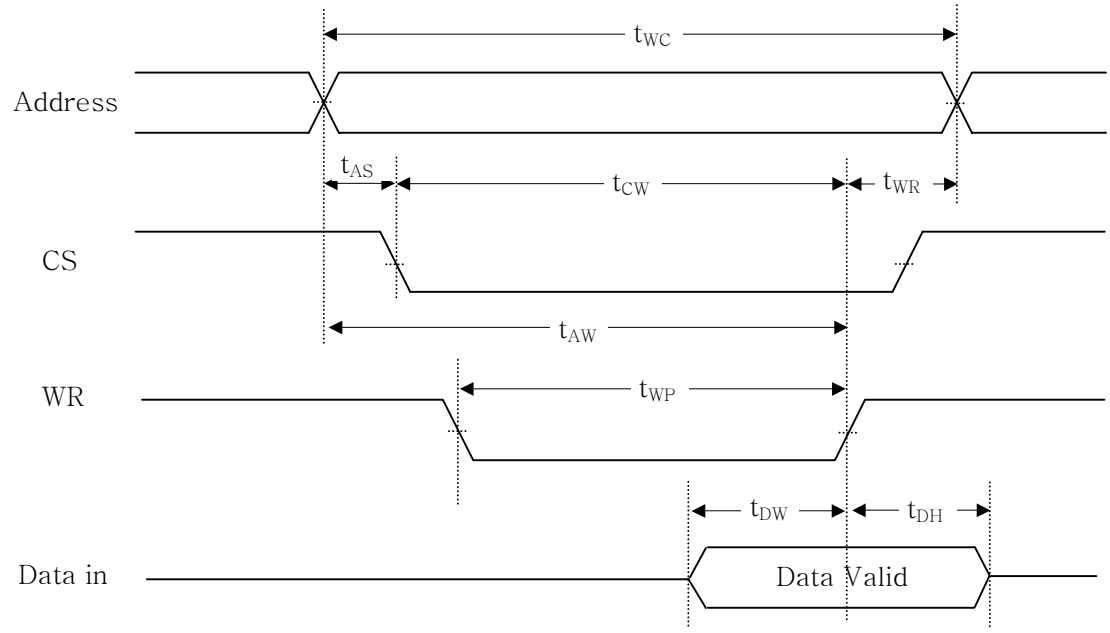
Register for MSS (maximum segment size), it indicates the MSS set by the peer when operated in passive mode of TCP.

■ Timing Diagrams

TIMING WAVEFORM OF Register/Memory READ CYCLE



TIMING WAVEFORM OF Register/Memory WRITE CYCLE



■ AC Characteristics

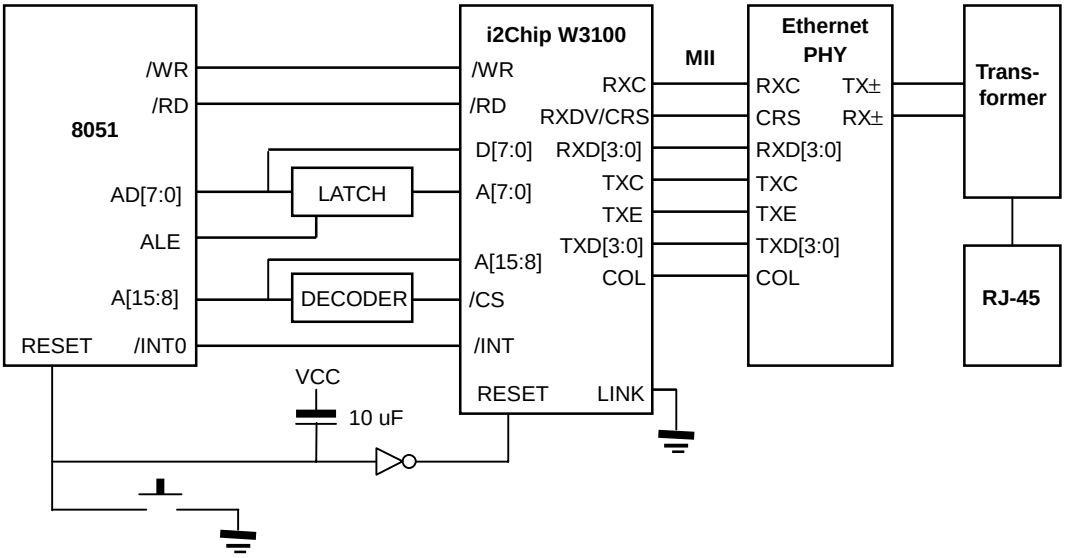
		Symbol	Speed		Units
			Min	Max	
Read	Read cycle time	t_{RC}	22	-	ns
	Address access time	t_{AA}	-	22	ns
	Chip select to output	t_{CO}	-	22	ns
	Output enable to valid output	t_{OE}	-	22	ns
	Chip select to low-Z output	t_{LZ}	5	-	ns
	Output enable to low-Z output	t_{OLZ}	5	-	ns
	Chip disable to high-Z output	t_{HZ}	5	15	ns
	Output disable to high-Z output	t_{OHZ}	5	15	ns
	Output hold from address change	t_{OH}	5	-	ns
Write	Write cycle time	t_{WC}	10	-	ns
	Chip select to end of write	t_{CW}	10	-	ns
	Address set-up time	t_{AS}	0	-	ns
	Address valid to end of write	t_{AW}	10	-	ns
	Write pulse width	t_{WP}	10	-	ns
	Write recovery time	t_{WR}	0	-	ns
	Data to write time overlap	t_{DW}	5	-	ns
	Data hold from write time	t_{DH}	5	-	ns

■ Power Consumption

- Typical value : 8.8mA

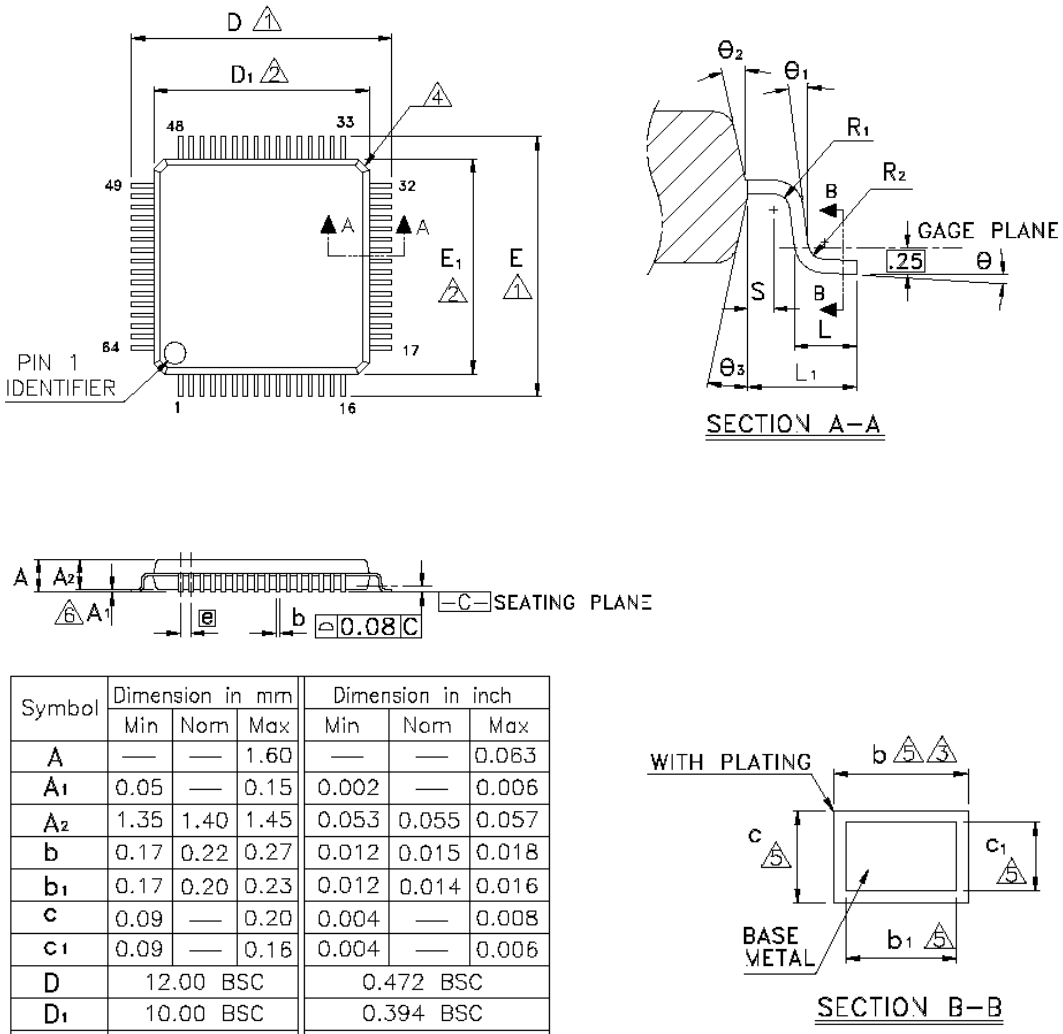
■ Application Information

Simplified Application Diagram



■ Package Description

Figure 1 : W3100 LQFP Package Specifications



Symbol	Dimension in mm			Dimension in 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	0.012	0.015	0.018
b_1	0.17	0.20	0.23	0.012	0.014	0.016
c	0.09	—	0.20	0.004	—	0.008
c_1	0.09	—	0.16	0.004	—	0.006
D	12.00 BSC			0.472 BSC		
D_1	10.00 BSC			0.394 BSC		
E	12.00 BSC			0.472 BSC		
E_1	10.00 BSC			0.394 BSC		
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		

■ W3100 Temperature/Humidity Test Results

1 Test Environment

1.1 Tested: i386 EVB with embedded W3100 i2Chip

1.2 Test period: 7 days

1.3 Test Performed: Normal operation through variation in temp. and humidity

Duration(H)	Temp.(℃)	Humidity(%)	Comments
12	0	0	
4	0 -> 50	0	Raised temp. to 50℃ over 4-hour period
4	50 -> 0	0	Lowered temp. to 0℃ over 4-hour period
20	0	0	Extended low-temp. with 0% humidity
8	0 -> 50	0 -> 70	Raised temp. and humidity over 8-hour period
20	50	70	Extended high-temp. and high-humidity
8	50 -> 0	70 -> 0	Lowered temp. and humidity over 8-hour period
24	0	0	Extended low-temp. with 0% humidity
8	0 -> 50	0 -> 70	Raised temp. and humidity over 8-hour period
24	50	70	Extended high-temp. and high-humidity
4	50 -> 25	70 -> 120	Lowered temp. and raised humidity over 4-hour period
20	25	120	Extended high-humidity with constant temp.
4	25 -> -30	120 -> 0	Lowered temp. and humidity over 4-hour period
4	-30 -> 70	0 -> 70	Raised temp. and humidity over 4-hour period
4	70 -> 25	70 -> 0	Lowered temp. and humidity over 4-hour period

2 Test Results

2.1 Performed Ping tests through duration of testing period.

2.2 Normal operation maintained.

2.3 Packet loss of 20 ~ 30% occurred during last 4 hours of testing.