W3150A Datasheet

(Ver. 1.0.1)

WIZnet

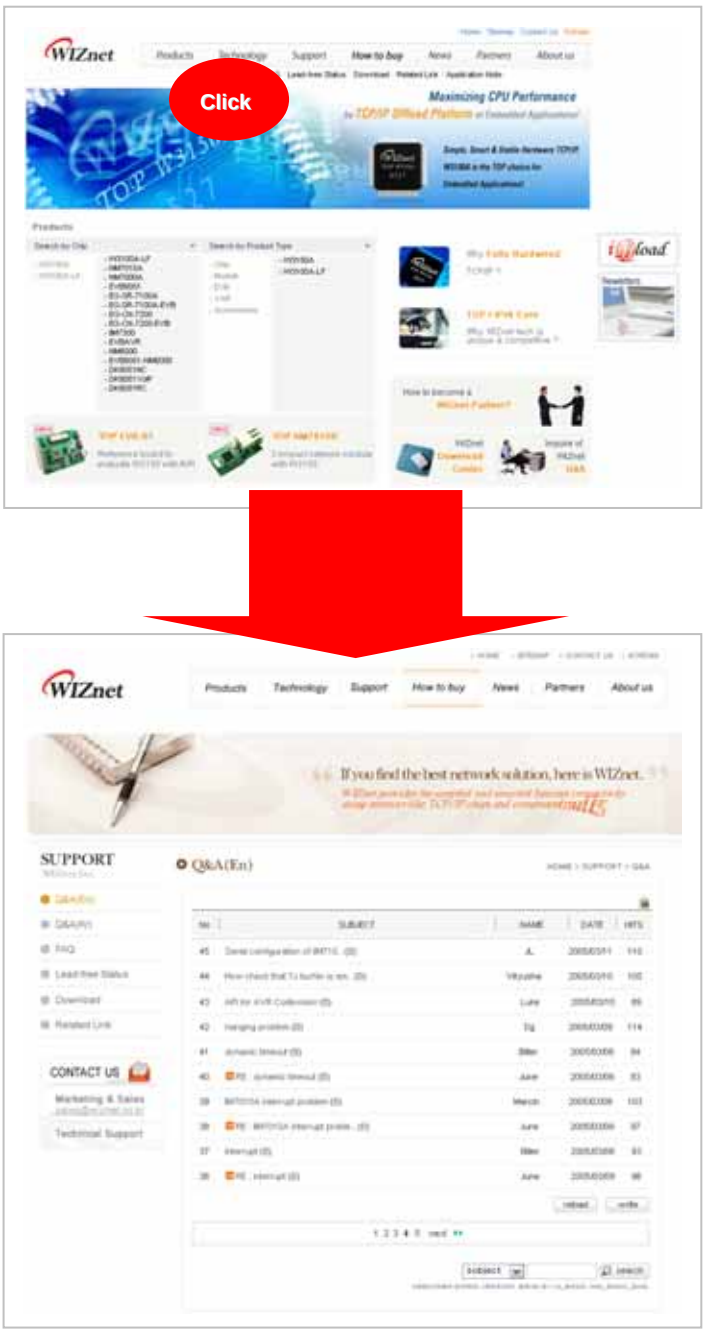
For more information, visit our website at http://www.wiznet.co.kr

Document History Information

Revision	Data	Description
Ver. 1.0.0	OCT 27, 2005	Release with W3150A Launching
Ver. 1.0.1	NOV 21, 2005	Replace, *1.8V* operation → **3.3V** operation (p.3) Change block diagram (p.4) Change figure (p.32) Replace, *g_Sn_TX_BASE* → **g_Sn_RX_BASE** (p.33) Replace, memcpy(, ,*left_size*) → in memcpy(, ,**upper_size**) (p.40, p.41, p.47, p.48, p.49) Replace, get_offset = *Sn_TX_RR* & → get_offset = **Sn_TX_WR** & (p.41, p.49) Replace, *SOCK_UDP* → **SOCK_IPRAW** (p.51)

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.



W3150A Datasheet

Description

The W3150A 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 W3150A offers system designers a quick, easy way to add Ethernet networking functionality to any product. 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 W3150A contains TCP/IP Protocol Stacks such as TCP, UDP, ICMP, IPv4, ARP and PPPoE protocols, as well as Ethernet protocols such as Data Link Control and MAC protocol. The total internal memory size is 16Kbytes, which is the buffer for transmit and receive operations.

The W3150A provides a local bus interface to various MCUs and standard MII(Media Independent Interface) specification consisting of nibble data bus for Ethernet PHY devices.

The W3150A is a best-fitted device for embedded application with Internet-Connectivity including IP-Setop Box, Internet-DVR, Internet phones, VoIP SOC chips, Internet MP3 players, handheld medical devices, various industrial system monitoring and metering, and many other non-portable electronic devices such as large consumer electronic products.

Features

- Support Hardwired TCP/IP Protocols : TCP, UDP, ICMP, IGMP, IPv4, ARP, PPPoE, Ethernet
- Support ADSL connection (with support PPPoE Protocol with PAP/CHAP Authentication mode)
- Supports 4 independent sockets simultaneously
- No support IP Fragmentation
- Standard MII Interface for Ethernet-PHY chip
- Supports 10BaseT/100BaseTX
- Supports full-duplex mode
- Internal 16Kbytes Memory for Tx/Rx Buffers
- 0.18 µm CMOS technology
- 3.3V operation with 5V I/O signal tolerance
- Small 64 Pin LQFP Package
- Lead-Free Package

Block Diagram

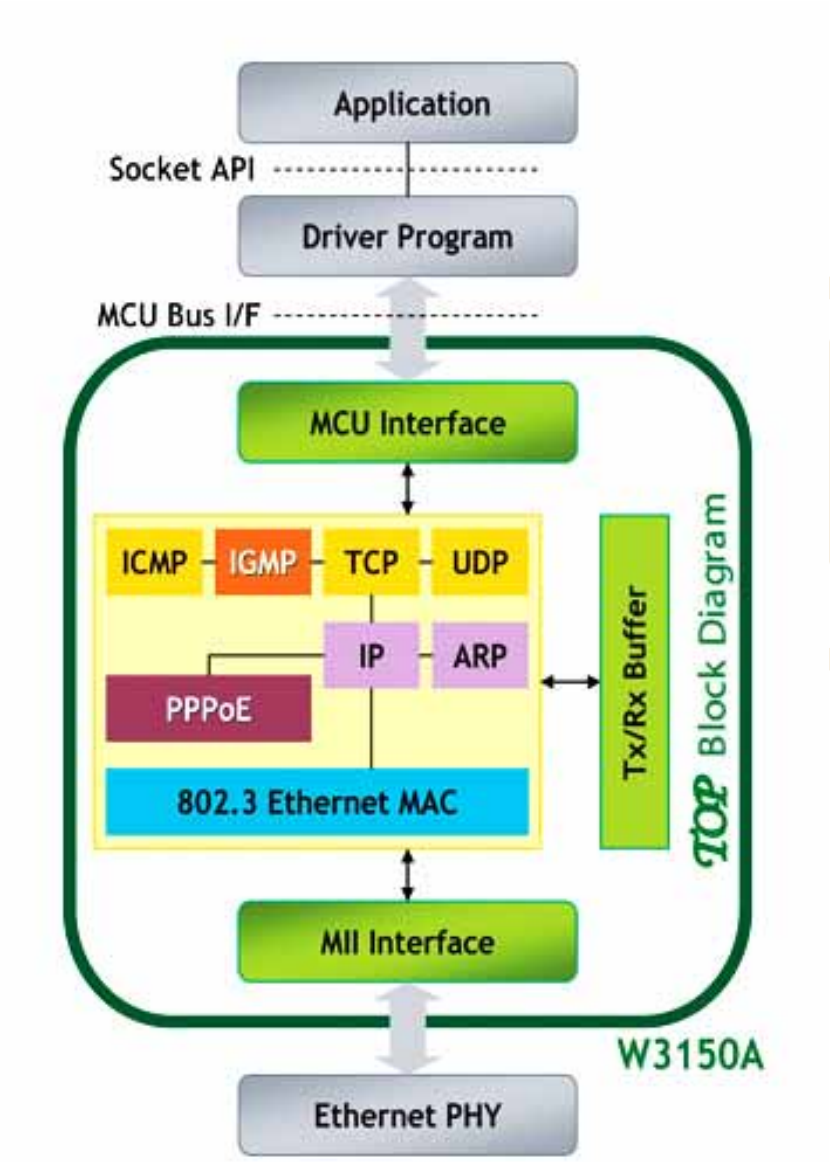
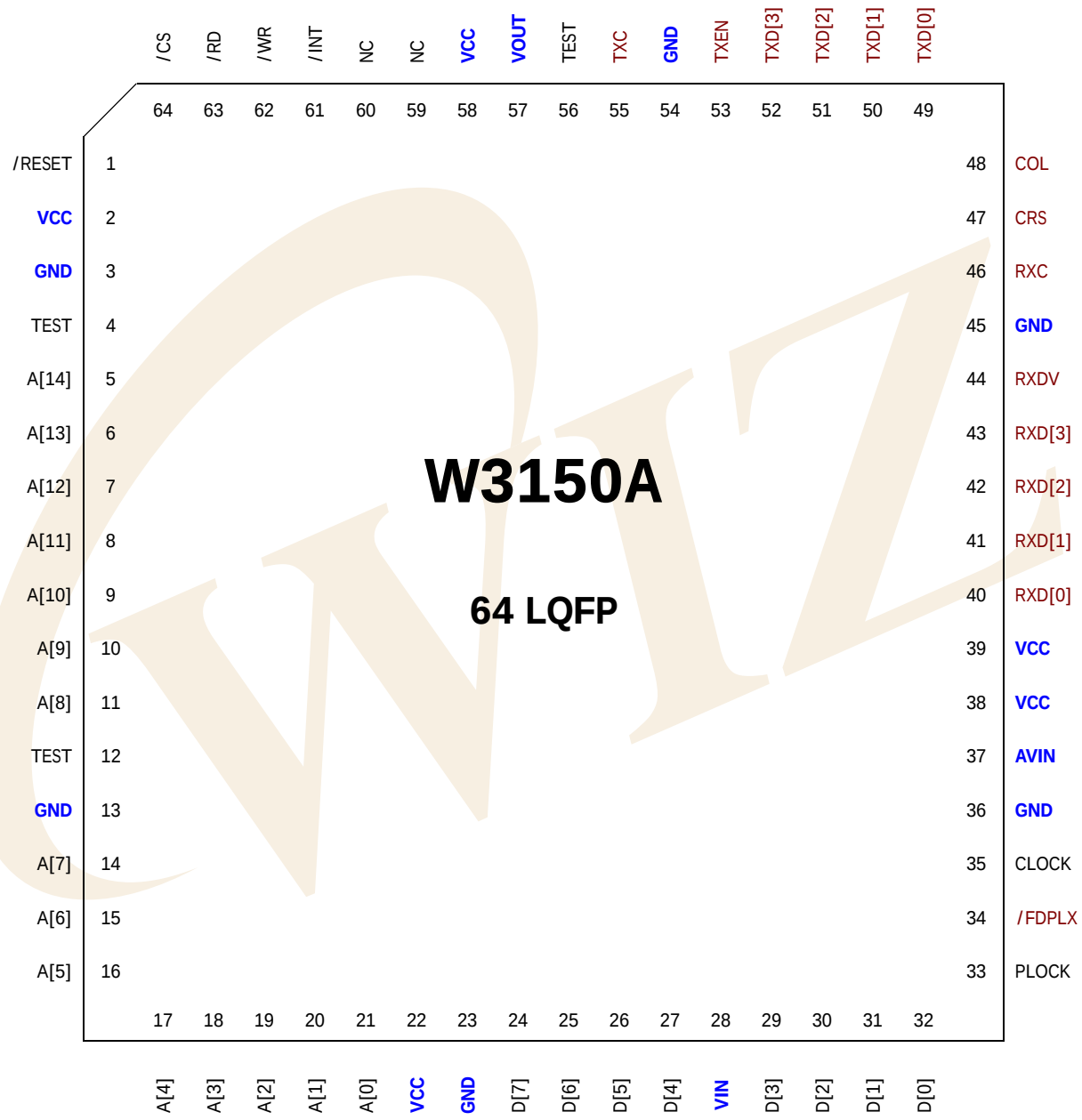


Figure 1. Block Diagram

Table of Contents

1. Pin Assignment



1.1. MII Signal Description

Pin#	Signal	I/O	Description
55	TXC	I	**Transmit Clock** This pin provides a continuous clock as a timing reference for TXD[3:0] and TXEN. TXC is sourced by the PHY. TXC is 2.5 MHz in 10BASET Nibble mode, and 25 MHz in 100BASETX Nibble mode.
53	TXEN	O	**Transmit Enable** This output signal indicates the presence of a valid nibble data on TXD[3:0]. It becomes active when the first nibble data of the packet is valid on TXD[3:0] and goes low after the last nibble data of the packet is clocked out of TXD[3:0]. This signal connects directly to the PHY device. This signal is active high.
52 51 50 49	TXD[3] TXD[2] TXD[1] TXD[0]	O	**Transmit Data** These pins provide Nibble NRZ data to the PHY synchronously with TXC when TXEN is asserted.
46	RXC	I	**Receive Clock** This pin provides a continuous clock reference for RXDV and RXD[3:0] signals. RXC is sourced by the PHY. RXC is 2.5 MHz in 10BASET Nibble mode, and 25 MHz in 100BASETX Nibble mode.
48	COL	I	**Collision Detect** This pin becomes active when a collision has been detected in Half Duplex modes. This signal is asynchronous, active high and ignored during full-duplex operation. This signal is active high.
47	CRS	I	**Carrier Sense** This pin indicates that carrier is present. This signal is active high.
44	RXDV	I	**Receive Data Valid** This signal is asserted high when received data is present on the RXD[3:0] pins; the signal is deasserted at the end of the packet. The signal is valid on the rising of the RXC.
43 42 41 40	RXD[3] RXD[2] RXD[1] RXD[0]	I	**Receive Data** These pins receive Nibble NRZ data from the PHY device synchronously with RXC when RXDV is asserted.

1.2. MCU Interface Signal Description

Pin#	Signal	I/O	Description
1	/RESET	I	**RESET** Active Low input that initializes or re-initializes W3150A. Asserting this pin low for at least 125us will force a reset process to occur which will result in all internal registers re-initializing to their default states.
35	CLOCK	I	**CLOCK** This pin is the Primary clock required for internal operation of W3150A. 25MHz is required. In general, PHY driving clock can be shared for saving cost. *Note) Sharing crystal source clock with multiple devices may cause some troubles. In our reference design, we used one crystal for both PHY and W3150A with verification.* *But for other kind of PHY, please confirm safety prior to decision.*
5:11 14:21	A[14:8] A[7:0]	I	**ADDRESS PINS** These pins are used to select a register or memory.
24:27, 29:32	D[7:4] D[3:0]	I/O	**DATA PINS** These pins are used to read and write register or memory data.
61	/INT	O	**INTERRUPT** This pin Indicates that W3150A requires MCU attention after socket connecting, disconnecting, receiving data or timeout. The interrupt is cleared by reading IR(Interrupt Register) or S*n*_IR (Socket *n*th Interrupt Register). All interrupts are maskable. This signal is active low.
64	/CS	I	**CHIP SELECT** Chip Select places for MCU access to internal registers/memory. /WR and /RD select direction of data transfer. This signal is active low.
62	/WR	I	**WRITE ENABLE** Strobe from MCU to write an internal register/memory selected by A[14:0]. Data is latched into the W3150A on the rising edge of this input. This signal is active low.
63	/RD	I	**READ ENABLE** Strobe from MCU to read an internal register/memory selected by A[14:0]. This signal is active low.

1.3. Miscellaneous Signal Description

Pin#	Signal	I/O	Description
34	/FDPLX	I	**FULL/HALF DUPLEX SELECT** This pin selects Half/Full Duplex operation. This pin must be externally pulled low (typically x kΩ) in order to configure the W3150A for Full Duplex operation. Low = Full Duplex High = Half Duplex
33	PLOCK	O	**PLL Lock** Internal PLL Lock detector out. When the lock signal is High, W3150A is ready to start. The lock signal always low when the PLL is in setting time. This signal is active high.
4,12,56, 59,60	TEST	I	**FACTORY TEST INPUT** Used to check the chip's internal functions. This should be tied low during normal operation.

1.4. Power Supply Signal Description

Pin#	Signal	I/O	Description
2, 22, 38, 39, 58	VCC		**POSITIVE 3.3V SUPPLY PINS**
28	VIN		**1.8V power input** 1.8V power supply
37	AVIN		**1.8V Analog power input** 1.8V power supply for analog circuit ; should be well decoupled. Refer Figure 1-1. Reference Schematic for Power input.
57	VOUT		**1.8V power out** Be sure to connect 10uF tantalum capacitor and a 0.1uF capacitor for noise de-coupling. Then connect this pin through a ferrite bead to VIN and AVIN.
3, 13, 23, 36, 45, 54	GND		**NEGATIVE (GROUND) SUPPLY PINS**

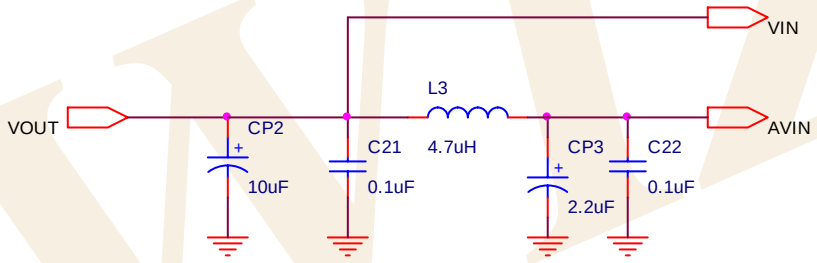
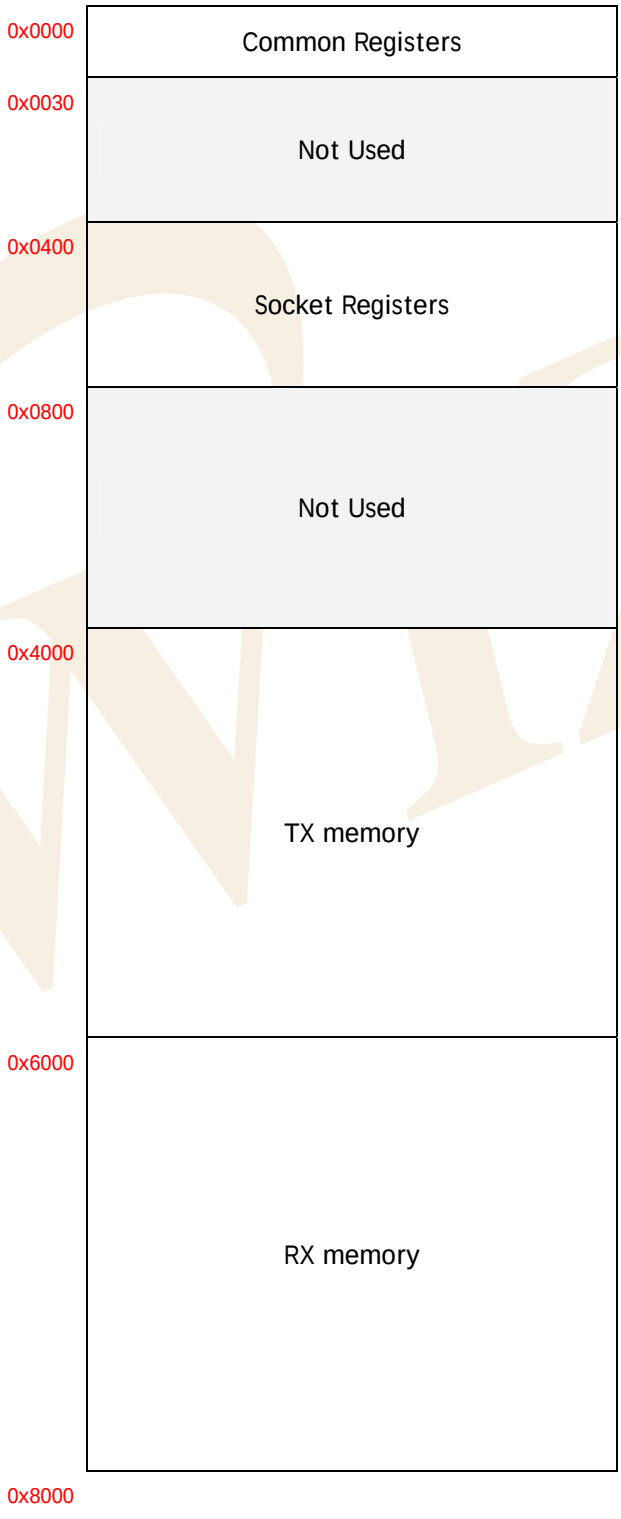


Figure 1-1. Reference Schematic for Power input

2. Memory map

W3150A is composed of Common Register, Socket Register, TX Memory, and RX Memory. Each fields are shown as below.



3. W3150A Registers

3.1. Common Registers

Address	Register
0x0000	Mode (MR)
0x0001	Gateway Address (GAR0)
0x0002	(GAR1)
0x0003	(GAR2)
0x0004	(GAR3)
0x0005	Subnet mask Address (SUBR0)
0x0006	(SUBR1)
0x0007	(SUBR2)
0x0008	(SUBR3)
0x0009	Source Hardware Address (SHAR0)
0x000A	(SHAR1)
0x000B	(SHAR2)
0x000C	(SHAR3)
0x000D	(SHAR4)
0x000E	(SHAR5)
0x000F	Source IP Address (SIPR0)
0x0010	(SIPR1)
0x0011	(SIPR2)
0x0012	(SIPR3)
0x0013 ~ 0x0014	Reserved
0x0015	Interrupt (IR)
0x0016	Interrupt Mask (IMR)
0x0017	Retry Time (RTR0)
0x0018	(RTR1)
0x0019	Retry Count (RCR)

Address	Register
0x001A	RX Memory Size (RMSR)
0x001B	TX Memory Size (TMSR)
0x001C	Authentication Type in PPPoE (PATR0)
0x001D	(PATR1)
0x001E ~ 0x0027	Reserved
0x0028	PPP LCP Request Timer (PTIMER)
0x0029	PPP LCP Magic number (PMAGIC)
0x002A	Unreachable IP Address (UIPR0)
0x002B	(UIPR1)
0x002C	(UIPR2)
0x002D	(UIPR3)
0x002E	Unreachable Port (UPORT0)
0x002F	(UPORT1)
0x0030 ~ 0x03FF	Reserved

3.2. Socket Registers

Address	Register
0x0400	Socket 0 Mode (S0_MR)
0x0401	Socket 0 Command (S0_CR)
0x0402	Socket 0 Interrupt (S0_IR)
0x0403	Socket 0 Socket Status (S0_SSR)
0x0404 0x0405	Socket 0 Source Port (S0_PORT0) (S0_PORT1)
0x0406 0x0407 0x0408 0x0409 0x040A 0x040B	Socket 0 Destination Hardware Address (S0_DHAR0) (S0_DHAR1) (S0_DHAR2) (S0_DHAR3) (S0_DHAR4) (S0_DHAR5)
0x040C 0x040D 0x040E 0x040F	Socket 0 Destination IP Address (S0_DIPR0) (S0_DIPR1) (S0_DIPR2) (S0_DIPR3)
0x0410 0x0411	Socket 0 Destination Port (S0_DPORT0) (S0_DPORT1)
0x0412 0x0413	Socket 0 Maximum Segment Size (S0_MSSR0) (S0_MSSR1)
0x0414	Socket 0 Protocol in IP Raw mode (S0_PROTO)

Address	Register
0x0415	Socket 0 IP TOS (S0_TOS)
0x0416	Socket 0 IP TTL (S0_TTL)
0x0417 ~ 0x041F	Reserved
0x0420 0x0421	Socket 0 TX Free Size (S0_TX_FSR0) (S0_TX_FSR1)
0x0422 0x0423	Socket 0 TX Read Pointer (S0_TX_RD0) (S0_TX_RD1)
0x0424 0x0425	Socket 0 TX Write Pointer (S0_TX_WR0) (S0_TX_WR1)
0x0426 0x0427	Socket 0 RX Received Size (S0_RX_RSR0) (S0_RX_RSR0)
0x0428 0x0429	Socket 0 RX Read Pointer (S0_RX_RR0) (S0_RX_RR1)
0x042A 0x042B	Reserved
0x042C ~ 0x04FF	Reserved

Address	Register
0x0500	Socket 1 Mode (S1_MR)
0x0501	Socket 1 Command (S1_CR)
0x0502	Socket 1 Interrupt (S1_IR)
0x0503	Socket 1 Socket Status (S1_SSR)
0x0504 0x0505	Socket 1 Source Port (S1_PORT0) (S1_PORT1)
0x0506 0x0507 0x0508 0x0509 0x050A 0x050B	Socket 1 Destination Hardware Address (S1_DHAR0) (S1_DHAR1) (S1_DHAR2) (S1_DHAR3) (S1_DHAR4) (S1_DHAR5)
0x050C 0x050D 0x050E 0x050F	Socket 1 Destination IP Address (S1_DIPR0) (S1_DIPR1) (S1_DIPR2) (S1_DIPR3)
0x0510 0x0511	Socket 1 Destination Port (S1_DPORT0) (S1_DPORT1)
0x0512 0x0513	Socket 1 Maximum Segment Size (S1_MSSR0) (S1_MSSR1)
0x0514	Socket 1 Protocol in IP Raw mode (S1_PROTO)

Address	Register
0x0515	Socket 1 IP TOS (S1_TOS)
0x0516	Socket 1 IP TTL (S1_TTL)
0x0517 ~ 0x051F	Reserved
0x0520 0x0521	Socket 1 TX Free Size (S1_TX_FSR0) (S1_TX_FSR1)
0x0522 0x0523	Socket 1 TX Read Pointer (S1_TX_RD0) (S1_TX_RD1)
0x0524 0x0525	Socket 1 TX Write Pointer (S1_TX_WR0) (S1_TX_WR1)
0x0526 0x0527	Socket 1 RX Received Size (S1_RX_RSR0) (S1_RX_RSR0)
0x0528 0x0529	Socket 1 RX Read Pointer (S1_RX_RR0) (S1_RX_RR1)
0x052A 0x052B	Reserved
0x052C ~ 0x05FF	Reserved

Address	Register
0x0600	Socket 2 Mode (S2_MR)
0x0601	Socket 2 Command (S2_CR)
0x0602	Socket 2 Interrupt (S2_IR)
0x0603	Socket 2 Socket Status (S2_SSR)
0x0604 0x0605	Socket 2 Source Port (S2_PORT0) (S2_PORT1)
0x0606 0x0607 0x0608 0x0609 0x060A 0x060B	Socket 2 Destination Hardware Address (S2_DHAR0) (S2_DHAR1) (S2_DHAR2) (S2_DHAR3) (S2_DHAR4) (S2_DHAR5)
0x060C 0x060D 0x060E 0x060F	Socket 2 Destination IP Address (S2_DIPR0) (S2_DIPR1) (S2_DIPR2) (S2_DIPR3)
0x0610 0x0611	Socket 2 Destination Port (S2_DPORT0) (S2_DPORT1)
0x0612 0x0613	Socket 2 Maximum Segment Size (S2_MSSR0) (S2_MSSR1)
0x0614	Socket 2 Protocol in IP Raw mode (S2_PROTO)

Address	Register
0x0615	Socket 2 IP TOS (S2_TOS)
0x0616	Socket 2 IP TTL (S2_TTL)
0x0617 ~ 0x061F	Reserved
0x0620 0x0621	Socket 2 TX Free Size (S2_TX_FSR0) (S2_TX_FSR1)
0x0622 0x0623	Socket 2 TX Read Pointer (S2_TX_RD0) (S2_TX_RD1)
0x0624 0x0625	Socket 2 TX Write Pointer (S2_TX_WR0) (S2_TX_WR1)
0x0626 0x0627	Socket 2 RX Received Size (S2_RX_RSR0) (S2_RX_RSR0)
0x0628 0x0629	Socket 2 RX Read Pointer (S2_RX_RR0) (S2_RX_RR1)
0x062A 0x062B	Reserved
0x062C ~ 0x06FF	Reserved

Address	Register
0x0700	Socket 3 Mode (S3_MR)
0x0701	Socket 3 Command (S3_CR)
0x0702	Socket 3 Interrupt (S3_IR)
0x0703	Socket 3 Socket Status (S3_SSR)
0x0704 0x0705	Socket 3 Source Port (S3_PORT0) (S3_PORT1)
0x0706 0x0707 0x0708 0x0709 0x070A 0x070B	Socket 3 Destination Hardware Address (S3_DHAR0) (S3_DHAR1) (S3_DHAR2) (S3_DHAR3) (S3_DHAR4) (S3_DHAR5)
0x070C 0x070D 0x070E 0x070F	Socket 3 Destination IP Address (S3_DIPR0) (S3_DIPR1) (S3_DIPR2) (S3_DIPR3)
0x0710 0x0711	Socket 3 Destination Port (S3_DPORT0) (S3_DPORT1)
0x0712 0x0713	Socket 3 Maximum Segment Size (S3_MSSR0) (S3_MSSR1)
0x0714	Socket 3 Protocol in IP Raw mode (S3_PROTO)

Address	Register
0x0715	Socket 3 IP TOS (S3_TOS)
0x0716	Socket 3 IP TTL (S3_TTL)
0x0717 ~ 0x071F	Reserved
0x0720 0x0721	Socket 3 TX Free Size (S3_TX_FSR0) (S3_TX_FSR1)
0x0722 0x0723	Socket 3 TX Read Pointer (S3_TX_RD0) (S3_TX_RD1)
0x0724 0x0725	Socket 3 TX Write Pointer (S3_TX_WR0) (S3_TX_WR1)
0x0726 0x0727	Socket 3 RX Received Size (S3_RX_RSR0) (S3_RX_RSR0)
0x0728 0x0729	Socket 3 RX Read Pointer (S3_RX_RR0) (S3_RX_RR1)
0x072A 0x072B	Reserved
0x072C ~ 0x07FF	Reserved

4. Register Descriptions

4.1. Common Registers

MR (Mode Register) [R/W] [0x0000] [0x00][1]

This Register is used for S/W Reset, memory test mode, ping block mode, PPPoE mode and Indirect bus I/F.

7	6	5	4	3	2	1	0
RST		MT	PB	PPPoE	LB	AI	IND

Bit	Symbol	Description
7	RST	**S/W Reset** If this bit is '1', internal register will be initialized. It will be automatically cleared with Reset.
6	Reserved	Reserved
5	MT	**Memory Test Mode** 0 : Disable memory test mode 1 : Enable memory test mode TX memory is used for data transmission. In this memory, only WRITE can be performed. At the RX memory, users can receive data and only READ can be performed. If a user wants to perform both of WRITE and READ in order to check operation status of TX memory and RX memory, the bit should be set as '1'. Memory Test Mode is used only for checking of TX/RX memory operation status. For the general usage, Memory Test Mode should be cleared first.
4	PB	**Ping Block Mode** 0 : Disable Ping block 1 : Enable Ping block If the bit is set as '1', there is no response to the Ping request.
3	PPPoE	**PPPoE Mode** 0 : Disable PPPoE mode 1 : Enable PPPoE mode If you are under the circumstance of ADSL without router or etc. you should set the bit as '1', and connect to ADSL SERVER. For more detail, refer to the application note, "How to connect ADSL".

1. [Read / Write] [Address] [Reset value]

2	LB	**Little-endian / Big-endian ordering in Indirect Bus I/F** 0 : Big-endian ordering 1 : Little-endian ordering At the Indirect Bus I/F mode, Indirect address Register ordering is decided. For more detail, refer to 6.1.2.Indirect Bus IF Mode.
1	AI	**Address Auto-Increment in Indirect Bus I/F** 0 : Disable auto-increment 1 : Enable auto-increment At the Indirect Bus I/F mode, if this bit is set as '1', the address will automatically increase by 1 whenever Read and Write are performed. For more detail, refer to 6.1.2 Indirect Bus IF Mode.
0	IND	**Indirect Bus I/F mode** 0 : Disable Indirect bus I/F mode 1 : Enable Indirect bus I/F mode If this bit is set as '1', Indirect Bis I/F mode is used. For more detail, refer to 6. Application Information, 6.1.2. Indirect Bus IF Mode.

GWR (Gateway IP Address Register) [R/W] [0x0001 – 0x0004] [0x00]

This Register sets up the default gateway address.

Ex) in case of "192.168.0.1"

0x0001	0x0002	0x0003	0x0004
192 (0xC0)	168 (0xA8)	0 (0x00)	1 (0x01)

SUBR (Subnet Mask Register) [R/W] [0x0005 – 0x0008] [0x00]

This Register sets up the subnet mask address.

Ex) in case of "255.255.255.0"

0x0005	0x0006	0x0007	0x0008
255 (0xFF)	255 (0xFF)	255 (0xFF)	0 (0x00)

SHAR (Source Hardware Address Register) [R/W] [0x0009 – 0x000E] [0x00]

This Register sets up the Source Hardware address.

Ex) In case of "00.08.DC.01.02.03"

0x0009	0x000A	0x000B	0x000C	0x000D	0x000E
0x00	0x08	0xDC	0x01	0x02	0x03

SIPR (Source IP Address Register) [R/W] [0x000F – 0x0012] [0x00]

This Register sets up the Source IP address.

Ex) in case of "192.168.0.3"

0x000F	0x0010	0x0011	0x0012
192 (0xC0)	168 (0xA8)	0 (0x00)	3 (0x03)

IR (Interrupt Register) [R] [0x0015] [0x00]

This Register is accessed by the host processor to know the cause of an interrupt.

Any interrupt can be masked in the Interrupt Mask Register (IMR). The /INT signal retain low as long as any unmasked signal is set, and will not go high until all unmasked bits in this Register have been cleared.

7	6	5	4	3	2	1	0
CONFLICT	UNREACH	PPPoE	Reserved	S3_INT	S2_INT	S1_INT	S0_INT

Bit	Symbol	Description
7	CONFLICT	**IP Conflict** It is set as '1', when there is ARP request with same IP address as Source IP address. This bit is cleared to '0' only with Read action.
6	UNREACH	**Destination unreachable** W3150A will receive ICMP(Destination Unreachable) packet if not-existing Destination IP address is transmitted during UDP data transmission. (Refer to 5.2.2. UDP). In this case, the IP address and Port number will be saved in Unreachable IP Address (UIPR) and Unreachable Port Register (UPORT), and the bit will be set as '1'. This bit will be cleared to '0' only with Read action.
5	PPPoE	**PPPoE Close** In the PPPoE Mode, if the PPPoE connection is closed, '1' is set. This bit will be cleared to '0' only with Read action.
4	Reserved	Reserved
3	S3_INT	**Occurrence of Socket 3 Socket Interrupt** It is set in case that interrupt occurs at the socket 3. For more detailed information of socket interrupt, refer to "Socket 3 Interrupt Register(S3_IR). This bit will be automatically cleared when S3_IR is cleared to 0x00.
2	S2_INT	**Occurrence of Socket 2 Socket Interrupt** It is set in case that interrupt occurs at the socket 2. For more detailed information of socket interrupt, refer to "Socket 2 Interrupt Register(S2_IR). This bit will be automatically cleared when S2_IR is cleared to 0x00.

1	S1_INT	**Occurrence of Socket 1 Socket Interrupt** It is set in case that interrupt occurs at the socket 1. For more detailed information of socket interrupt, refer to "Socket 1 Interrupt Register(S1_IR). This bit will be automatically cleared when S1_IR is cleared to 0x00.
0	S0_INT	**Occurrence of Socket 0 Socket Interrupt** It is set in case that interrupt occurs at the socket 0. For more detailed information of socket interrupt, refer to "Socket 0 Interrupt Register(S0_IR). This bit will be automatically cleared when S0_IR is cleared to 0x00.

IMR (Interrupt Mask Register) [R/W] [0x0016] [0x00]

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Register (IR). If an interrupt mask bit is set, an interrupt will be issued whenever the corresponding bit in the IR is set. If any bit in the IMR is set as '0', an interrupt will not occur though the bit in the IR is set.

7	6	5	4	3	2	1	0
IM_IR7	IM_IR6	IM_IR5	Reserved	IM_IR3	IM_IR2	IM_IR1	IM_IR0

Bit	Symbol	Description
7	IM_IR7	**IP Conflict Enable**
6	IM_IR6	**Destination unreachable Enable**
5	IM_IR5	**PPPoE Close Enable**
4	Reserved	**It should be set as '0'**
3	IM_IR3	**Occurrence of Socket 3 Socket Interrupt Enable**
2	IM_IR2	**Occurrence of Socket 2 Socket Interrupt Enable**
1	IM_IR1	**Occurrence of Socket 1 Socket Interrupt Enable**
0	IM_IR0	**Occurrence of Socket 0 Socket Interrupt Enable**

RTR (Retry Time-value Register) [R/W] [0x0017 – 0x0018] [0x07D0]

This register sets the period of timeout. Vaule 1 means 100us. The initial value is 2000(0x07D0). That will be set as 200ms.

Ex) For 400ms configuration, set as 4000(0x0FA0)

0x0017	0x0018
0x0F	0xA0

Re-transmission will occur when there is no response from the remote peer to the commands of CONNECT, DISCON, CLOSE, SEND, SEND_MAC and SEND_KEEP, or the response is delayed.

RCR (Retry Count Register) [R/W] [0x0019] [0x08]

This register sets the number of re-transmission. If retransmission occurs more than recorded number, Timeout Interrupt (TIMEOUT bit of Socket *n* Interrupt Register (Sn_IR) is set as '1') will occur.

RMSR(RX Memory Size Register) [R/W] [0x001A] [0x55]

This register assigns total 8K RX Memory to the socket.

7	6	5	4	3	2	1	0
Socket 3		Socket 2		Socket 1		Socket 0	
S1	S0	S1	S0	S1	S0	S1	S0

The memory size according to the configuration of S1, S0, is as below.

S1	S0	Memory size
0	0	1KB
0	1	2KB
1	0	4KB
1	1	8KB

Accoridng to the value of S1 and S0, the memory is assigned to the sockets from socket 0 within the range of 8KB. If there is not enough memory to be assigned, the socket should not be used. The initial value is 0x55 and the memory is assined to 4 sockets respectively with 2K.

Ex) When setting as 0xAA, the memory should be assigned to each socket with 4KB.

However, the total memory size is 8KB. The memory is normally assigned to the socket 0 and 1, but not to the socket 2 and 3. Therefore, user should not use socket 2 and 3 absolutely.

Socket 3	Socket 2	Socket 1	Socket 0
0KB	0KB	4KB	4KB

TMSR(TX Memory Size Register) [R/W] [0x001B] [0x55]

This register is used in assigning total 8K TX Memory to sockets. Configuration can be done in the same way of RX Memory Size Register (RMSR). The initial value is 0x55 and assined to 4 socketes respectively with 2K.

PATR (Authentication Type in PPPoE mode) [R] [0x001C-0x001D] [0x0000]

This register notifies authentication method that has been agreed at the connection with PPPoE Server. W3150A supports only two ways of PAP and CHAP.

Value	Authentication Type
0xC023	PAP
0xC223	CHAP

PTIMER (PPP LCP Request Timer Register) [R/W] [0x0028] [0x28]

This register indicates the duration for sending LCP Echo Request. Vaule 1 is about 25ms.

Ex) in case that PTIMER is 200,

200 * 25(ms) = 5000(ms) = 5 seconds

PMAGIC (PPP LCP Magic number Register) [R/W] [0x0029] [0x00]

This register is used in Magic number option during LCP negotiation. Refer to the application note, "How to connect ADSL".

UIPR (Unreachable IP Address Register) [R] [0x002A - 0x002D] [0x00]

In case of data transmission by using UDP (refer to 5.2.2. UDP), if transmitting to not-existing IP address, ICMP (Destination Unreachable) packet will be received. In this case, that IP address and Port number will be respectively saved in the Unreachable IP Address Register(UIPR) and Unreachable Port Register(UPORT).

Ex) in case of "192.168.0.11",

0x002A	0x002B	0x002C	0x002D
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

UPORT (Unreachable Port Register) [R] [0x002E - 0x002F] [0x0000]

Refer to Unreachable IP Address Register (UIPR)

Ex) In case of 5000(0x1388),

0x002E	0x002F
0x13	0x88

4.2. Socket Registers

Sn[2]_MR (Socket *n* Mode Register) [R/W] [0x0400, 0x0500, 0x0600, 0x0700] [0x00][3]

This Register sets up socket option or protocol of the corresponding Socket.

7	6	5	4	3	2	1	0
MULTI		MC	ZC	P3	P2	P1	P0

Bit	Symbol	Description
7	MULTI	**Multicasting** 0 : disable Multicasting 1 : enable Multicasting It is applied only in case of UDP. For using multicasting, write multicast group address to Socket *n* Destination IP and multicast group port number to Socket *n* Destination Port Register, before OPEN command.
6	Reserved	Reserved
5	MC	**Multicast** 0 : using IGMP version 2 1 : using IGMP version 1 It is applied only in case of MULTI is '1'
4	ZC	**Zero Chksum Enable** 0 : disable Zero Chksum 1 : enable Zero Chksum It is applied only in case of UDP. If this bit is set as '1', checksum field of transmitting packet is 0x00.
3 2 1	P3 P2 P1	**Protocol** Sets up corresponding Socket as TCP, UDP, IP RAW mode

P3	P2	P1	P0	Meaning
0	0	0	0	Closed
0	0	0	1	TCP
0	0	1	0	UDP
0	0	1	1	IPRAW

[2] *n* is socket number (0, 1, 2, 3).
[3] [Read / Write] [address of socket 0, address of socket 1, address of socket 2, address of socket 3] [Reset value]

0	P0	*** In case of socket 0, MACRAW and PPPoE mode exist.** P3 / P2 / P1 / P0 / Meaning 0 / 1 / 0 / 0 / MACRAW 0 / 1 / 0 / 1 / PPPoE

S*n*_CR (Socket *n* Command Register) [R/W] [0x0401, 0x0501, 0x0601, 0x0701] [0x00]

This register utilized for socket initialization, close, connection establishiment, connection termination, data transmitting and receiving commands. After performing the commands, the register value will be automatically cleared to 0x00.

Value	Symbol	Description
0x01	OPEN	It is used to initialize the socket. According to the vaule of Socket *n* Mode Register(S*n*_MR), Socket *n* Status Register(S*n*_SSR) value is changed to SOCK_INIT, SOCK_UDP, SOCK_IPRAW. For more detail, refer to 5. Functional Description.
0x02	LISTEN	It is only used in TCP mode. It changes the value of Socket *n* Status Register(Sn_SSR) to SOCK_LISTEN in order to wait for a connection request from any remote peer (Client). For more detail, refer to 5.2.1.1. SERVER.
0x04	CONNECT	It is only used in TCP mode. It sends a connection request to remote peer(SERVER). If the connection is failed, Timeout interrupt will occur. For more detail, refer to 5.2.1.2. CLIENT.
0x08	DISCON	It is only used in TCP mode. It sends connection termination request. If connection termination is failed, Timeout interrupt will occur. For more detail, refert to 5.2.1.1. SERVER. ** In case of using CLOSE command instead of DISCON, only the value of Socket n Status Register(Sn_SSR) is changed to SOCK_CLOSED without the connection termination process.*
0x10	CLOSE	It is used to close the socket. It changes the vaule of Socket *n* Status Register(S*n*_SSR) to SOCK_CLOSED.
0x20	SEND	It transmits the data as much as the increased size of Socket *n* TX Write Pointer. For more detail, refert to Socket *n* TX Free Size Register (S*n*_TX_FSR), Socket *n* TX Write Pointer Register(S*n*_TX_WR), and Socket *n* TX Read Pointer Register(S*n*_TX_RR) or 5.2.1.1. SERVER.
0x21	SEND_MAC	It is used in UDP mode. The basic operation is same as SEND. Normally SEND operation is needed

		Destination Hardware Address that is received in ARP(Address Resolution Protocol) process. But SEND_MAC uses Socket *n* Destination Hardware Address(S*n*_DHAR) that is written by users without ARP process.
0x22	SEND_KEEP	It is only used in TCP mode. It checks the connection by sending 1byte data previously. If the connection is already terminated or Peer has no response, Timeout interrupt will occur.
0x40	RECV	Receiving is processed including the value of Socket *n* RX Read Pointer Register(S*n*_RX_RR). For more detail, refer to Socket *n* RX Received Size Register (S*n*_RX_RSR), Socket *n* RX Write Pointer Register(S*n*_RX_WR), and Socket *n* RX Read Pointer Register(S*n*_RX_RR) or 5.2.1.1. SERVER

S*n*_IR (Socket *n* Interrupt Register) [R] [0x0402, 0x0502, 0x0602, 0x0702] [0x00]

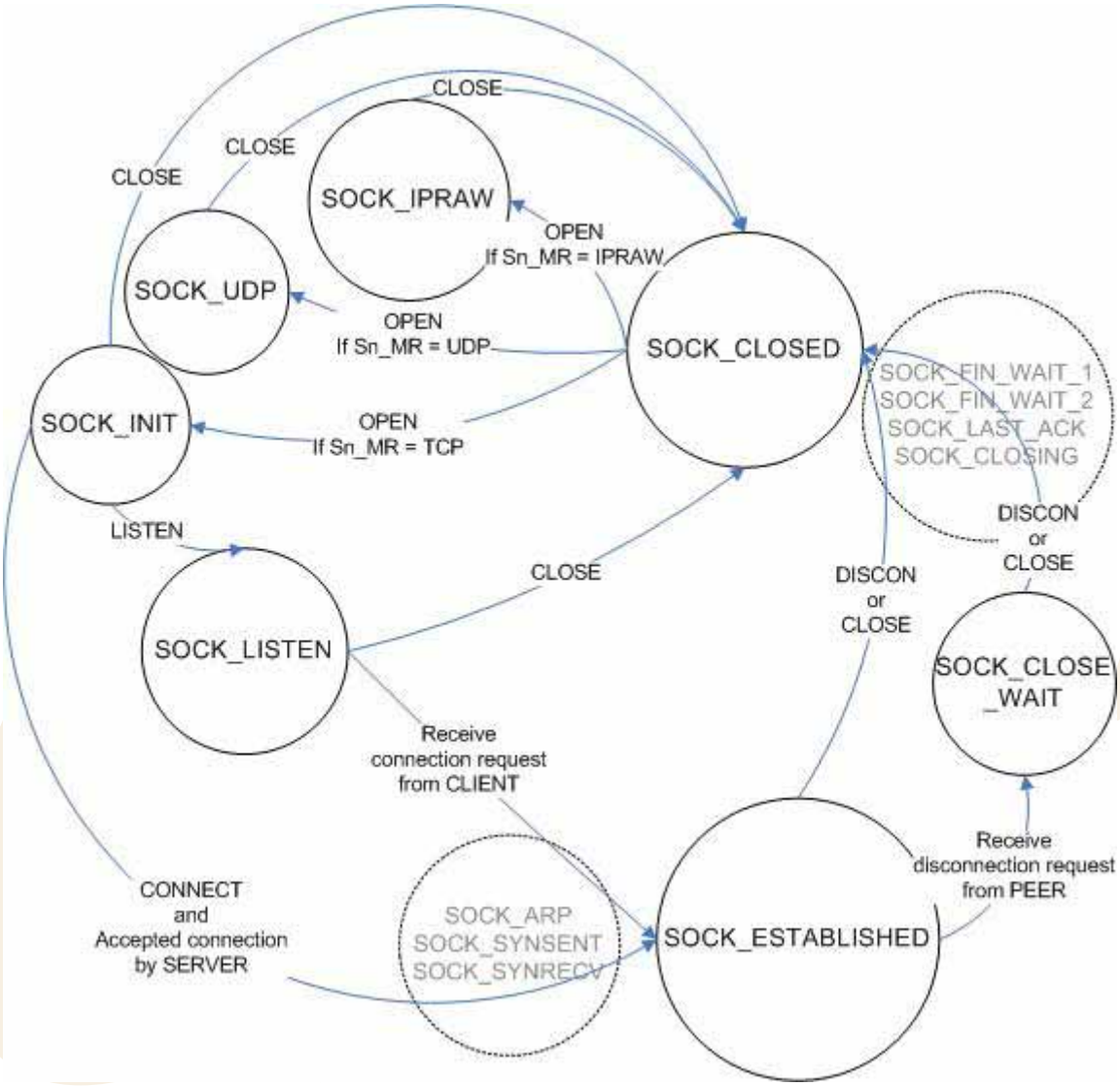
This register is used for notifying connection establishment and termination, receiving data and Timeout. The values are cleared by reading this register. Howerver, it takes 2ms to clear register after reading internally, and you may wait about 2ms after reading this register.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	TIMEOUT	RECV	DISCON	CON

Bit	Symbol	Description
7	Reserved	Reserved
6	Reserved	Reserved
5	Reserved	Reserved
4	Reserved	Reserved
3	TIMEOUT	It is set as ‘1’ if Timeout occurs during connection establishment or termination and data transmission.
2	RECV	It is set as ‘1’ if data is received.
1	DISCON	It is set as ‘1’ if receive connection termination request or if connection termination is finished.
0	CON	It is set as ‘1’ if connection is established.

S*n*_SR (Socket *n* Status Register) [R] [0x0403, 0x0503, 0x0603, 0x0703] [0x00]

This register has the status vaule of socket. The main status is shown in the below diagram.



Value	Symbol	Description
0x00	SOCK_CLOSED	In case that OPEN commands are given to S*n*_CR, Timeout interrupt is asserted or connection is terminated.
0x13	SOCK_INIT	In case that S*n*_MR is set as TCP and OPEN commands are given to S*n*_CR.
0x14	SOCK_LISTEN	In case that under the SOCK_INIT status, LISTEN commands are given to S*n*_CR.
0x17	SOCK_ESTABLISHED	In case that connection is established.
0x1C	SOCK_CLOSE_WAIT	In case that connection termination request is received.

0x22	SOCK_UDP	In case that OPEN commands are given to S*n*_CR when S*n*_MR is set as UDP.
0x32	SOCK_IPRAW	In case that OPEN commands are given to S*n*_CR when S*n*_MR is set as IPRAW.
0x42	SOCK_MACRAW	In case that OPEN commands are given to S0_CR when S0_MR is set as MACRAW.
0x5F	SOCK_PPPOE	In case that OPEN commands are given to S0_CR when S0_MR is set as PPPoE.

Below is shown in the status change, and does not need much attention.

Value	Symbol	Description
0x15	SOCK_SYNSENT	It is shown in case that CONNECT commands are given to Socket *n* Command Register(S*n*_CR) at the SOCK_INIT status. It is automatically changed to SOCK_ESTABLISH when the connection is established.
0x16	SOCK_SYNRECV	It is shown in case that connection request is received from remote peer(CLIENT). It normally responds to the requests and changes to SOCK_ESTABLISH.
0x18	SOCK_FIN_WAIT	It is shown in the process of connection termination. If the termination is normally processed or Timeout interrupt is asserted then automatically changed to SOCK_CLOSED.
0x1A	SOCK_CLOSING	
0X1B	SOCK_TIME_WAIT	
0X1D	SOCK_LAST_ACK	
0x11 0x21 0x31	SOCK_ARP	It is shown when ARP Request is sent in order to acquire Hardware Address of remote peer when it sends connection request in TCP mode or sends data in UDP mode. If ARP Reply is received, it changes to the status, SOCK_SYNSENT, SOCK_UDP or SOCK_ICMP, for next operation.

S*n*_PORT (Socket *n* Source Port Register) [R/W] [0x0404-0x0405, 0x0504-0x0505, 0x0604-0x0605, 0x0704-0x0705] [0x00]

This Register sets the Source Port number for each Socket when using TCP or UDP mode, and the set-up needs to be made before executing the OPEN Command.

Ex) In case of Socket 0 Port = 5000(0x1388), configure as below,

0x0404	0x0405
0x13	0x88

S*n*_DHAR (Socket *n* Destination Hardware Address Register) [R/W] [0x0406-0x040B, 0x0506-0x050B, 0x0606-0x060B, 0x0706-0x070B] [0x00]

This Register sets the Destination Hardware address of each Socket.

Ex) In case of Socket 0 Hardware address = 08.DC.00.01.02.10, configuration is as below,

0x0406	0x0407	0x0408	0x0409	0x040A	0x040B
0x08	0xDC	0x00	0x01	0x02	0x0A

S*n*_DIPR (Socket *n* Destination IP Address Register) [R/W] [0x040C-0x040F, 0x050C-0x050F, 0x060C-0x060F, 0x070C-0x070F] [0x00]

This Register sets the Destination IP Address of each Socket to be used in setting the TCP connection. In active mode, IP address needs to be set before executing the Connect command. In passive mode, W3150A sets up the connection and then updates with peer IP internally.

Ex) In case of Socket 0 Destination IP address = 192.168.0.11, configure as below.

0x040C	0x040D	0x040E	0x040F
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

S*n*_DPORT (Socket *n* Destination Port Register) [R/W] [0x0410-0x0411, 0x0510-0x0511, 0x0610-0x0611, 0x0710-0x0711] [0x00]

This Register sets the Destination Port number of each Socket to be used in setting the TCP connection. In active mode, port number needs to be set before executing the Connect command. In passive mode, W3150A sets up the connection and then updates with peer port number internally.

Ex) In case of Socket 0 Destination Port = 5000(0x1388), configure as below,

0x0410	0x0411
0x13	0x88

S*n*_MSS (Socket *n* Maximum Segment Size Register) [R/W] [0x0412-0x0413, 0x0512-0x0513, 0x0612-0x0613, 0x0712-0x0713] [0x00]

This Register is used for MSS (Maximum Segment Size) of TCP, and the Register displays MSS set by the other party when TCP is activated in Passive Mode.

Ex) In case of Socket 0 MSS = 1460(0x05B4), configure as below,

0x0412	0x0413
0x05	0xB4

Sn_PROTO (Socket *n* IP Protocol Register) [R/W] [0x0414, 0x0514, 0x0614, 0x0714] [0x00]

This IP Protocol Register is used to be set up the Protocol Field of IP Header when executing the IP Layer RAW Mode. There are several protocol numbers defined in advance by registering to IANA. For the overall list of upper level protocol idenentification number that IP is using, refer to online documents of IANA (http://www.iana.org/assignments/protocol-numbers).

Ex) Internet Control Message Protocol (ICMP) = 0x01, Internet Group Management Protocol = 0x02

Sn_TOS (Socket *n* IP Type Of Service Register) [R/W] [0x0415,0x0515,0x0615,0x0715] [0x00]

This Register sets up at the TOS Field of IP Header.

Sn_TTL (Socket *n* IP Time To Live Register) [R/W] [0x0416,0x0516,0x0616,0x0716] [0x80]

This Register sets up at the TTL Field of IP Header.

Sn_TX_FSR (Socket *n* TX Free Size Register) [R] [0x0420-0x0421, 0x0520-0x0521, 0x0620-0x0621, 0x0720-0x0721] [0x0800]

This regiser notifies the information of data size that user can transmit. For data transmission, user should check this value first and control the size of transmitting data. *When checking this register, user should read upper byte(0x0420,0x0520,0x0620,0x0720) first and lower byte(0x0421,0x0521,0x0621,0x0721) later to get the correct value.*

Ex) In case of 2048(0x0800) in S0_TX_FSR,

0x0420	0x0421
0x08	0x00

Total size can be decided according to the value of TX Memory Size Regster. In the process of transmission, it reduces by the size of transmitting data, and automatically increases after transimission finished.

S*n*_TX_RR (Socket *n* TX Read Pointer Register) [R] [0x0422-0x0423, 0x0522-0x0523, 0x0622-0x0623, 0x0722-0x0723] [0x0000]

This register shows the addres that transmission is finished at the TX Memory. With the SEND command of Socket *n* Command Register, it transmits data from current S*n*_TX_RR to S*n*_TX_WR and automatically changes after transmission is finished. Therefore, after transmission is finished, S*n*_TX_RR and S*n*_TX_WR will have same value. When reading this register, user should read upper byte (0x0422, 0x0522, 0x0622, 0x0722) first and lower byte (0x0423, 0x0523, 0x0623, 0x0723) later to get the correct value.

S*n*_TX_WR (Socket *n* TX Write Pointer Register) [R/W] [0x0424-0x0425, 0x0524-0x0525, 0x0624-0x0625, 0x0724-0x0725] [0x0000]

This register offers the location information to write the transmission data. When reading this register, user should read upper byte (0x0424, 0x0524, 0x0624, 0x0724) first and lower byte (0x0425, 0x0525, 0x0625, 0x0725) later to get the correct value.

Ex) In case of 2048(0x0800) in S0_TX_WR,

0x0424	0x0425
0x08	0x00

But this value itself is not the physical address to write. So, the physical address should be calculated as follow.

1. Socket n TX Base Address (hereafter we'll call *gSn_TX_BASE)* and Socket n TX Mask Address (hereafter we'll call *gSn_TX_MASK)* are calculated on TMSR value. *Refer to the psedo code of the 5.1 Initialization if the detail is needed.*
2. The bitwise-AND operation of two values, Sn_TX_WR and *gSn_TX_MASK* give result the offset address(hereafter we'll call *get_offset*) in TX memory range of the socket.
3. Two values *get_offset* and *gSn_TX_BASE* are added together to give result the physical address(hereafter, we'll call *get_start_address*).

Now, write the transmission data to *get_start_address* as large as you want. (* There's a case that it exceeds the TX memory upper-bound of the socket while writing. In this case, write the transmission data to the upper-bound, and change the physical address to the *gSn_TX_BASE*. Next, write the rest of the transmission data.)

After that, be sure to increase the Sn_TX_WR value as large as the data size, that indicates the size of writing data. Finally, give SEND command to Sn_CR(Socket n Command Register).

Refer to the psedo code of the transmission part on 5.2.1.1. TCP Server mode if the detail is needed.

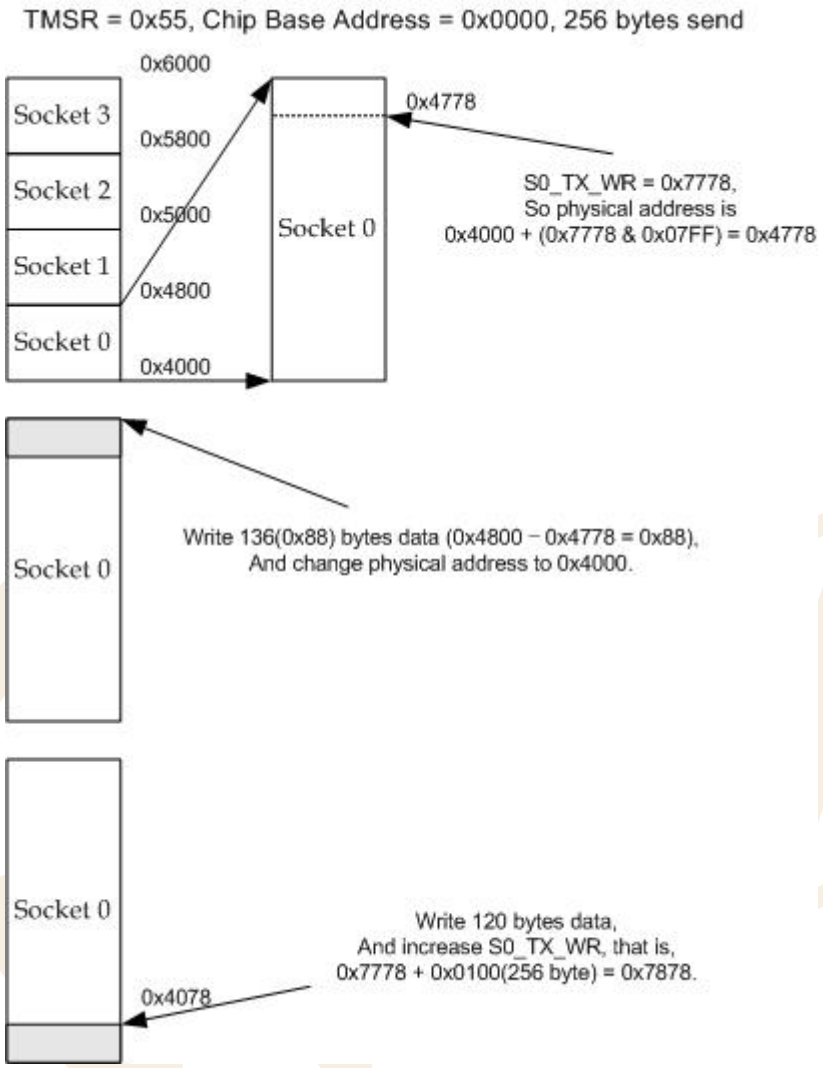


Figure 2. Calculate physical address

S*n*_RX_RSR (RX Received Size Register) [R] [0x0426-0x0427, 0x0526-0x0527, 0x0626-0x0627, 0x0726-0x0727] [0x0000]

This register notifies the data size received in RX Memory. As this value is internally calculated with the values of S*n*_RX_RR and S*n*_RX_WR, it is automatically changed by RECV command of Socket *n* Command Register(S*n*_CR) and receiving data for remote peer. *When reading this register, user should read upper byte(0x0426,0x0526,0x0626,0x0726) first and lower byte(0x0427,0x0527,0x0627,0x0727) later to get the correct value.*

Ex) In case of 2048(0x0800) in S0_RX_RSR,

0x0426	0x0427
0x08	0x00

The total size of this value can be decided according to the value of RX Memory Size Register.

S*n*_RX_RR (Socket *n* RX Read Pointer Register) [R/W] [0x0428-0x0429, 0x0528-0x0529, 0x0628-0x0629, 0x0728-0x0729] [0x0000]

This register offers the location information to read the receiving data. When reading this register, user should read upper byte (0x0428, 0x0528, 0x0628, 0x0728) first and lower byte (0x0429, 0x0529, 0x0629, 0x0729) later to get the correct value.

Ex) In case of 2048(0x0800) in S0_RX_RR,

0x0428	0x0429
0x08	0x00

But this value itself is not the physical address to read. So, the physical address should be calculated as follow.

1. Socket n RX Base Address (hereafter we'll call *gSn_RX_BASE)* and Socket n RX Mask Address (hereafter we'll call *gSn_RX_MASK)* are calculated on RMSR value. *Refer to the psedo code of the 5.1 Initialization if the detail is needed.*
2. The bitwise-AND operation of two values, Sn_RX_RR and *gSn_RX_MASK* give result the offset address(hereafter we'll call *get_offset)*, in RX memory range of the socket.
3. Two values *get_offset* and *gSn_RX_BASE* are added together to give result the physical address(hereafter, we'll call *get_start_address).*

Now, read the receiving data from *get_start_address* as large as you want. (* There's a case that it exceeds the RX memory upper-bound of the socket while reading. In this case, read the receiving data to the upper-bound, and change the physical address to the *gSn_RX_BASE*. Next, read the rest of the receiving data.)

After that, be sure to increase the Sn_RX_RR value as large as the data size, that indicates the size of reading data. (* Must not increase more than the size of received data. So must check Sn_RX_RSR before receiving process.) Finally, give RECV command to Sn_CR(Socket n Command Register).

Refer to the psedo code of the receiving part on 5.2.1.1. TCP Server mode if the detail is needed.

5. Functional Description

By setting some register and memory operation, W3150A provides internet connectivity. This chapter describes how it can be operated.

5.1. Initialization

- Setting network information

Below register is for basic network configuration information to be configured according to the network environment.

1. Gateway Address Register (GAR)
2. Source Hardware Address Register (SHAR)
3. Subnet Mask Register (SUBR)
4. Source IP Address Register (SIPR)

The Source Hardware Address Regiter (SHAR) is the H/W address to be used in MAC layer, and can be used with the address that manufacturer has been assigned. The MAC address can be assigned from IEEE. For more detail, refer to IEEE homepage.

- Inform Gateway of local network information & Getting Gateway Hardware address

Inform Gateway of local network information for updating ARP cache table on Gateway and getting Gateway Hardware Address using UDP sending. This procedure can be processed as below.

```
{
START:
    /* set TCP on socket 0 mode register and open socket */
    S0_MR = 0x01;
    S0_CR = OPEN;
    if (S0_SSR != SOCK_INIT) S0_CR = CLOSE; goto START;

    /* request TCP connection to any host in different network */
    S0_DIPR0 = SIPR0 + 1; // for making different network
    S0_CR = CONNECT; // Set CONNECT command
    Wait 10 msec;

    /* getting Gateway Hardware Address */
    If (S0_DHAR0 == 0xFF) goto START; // Can't receive information
    else gGatewayHA = S0_DHAR; // save gateway hardware address (6bytes)
```

```
    /* close socket */
    S0_CR = CLOSE;
}
```

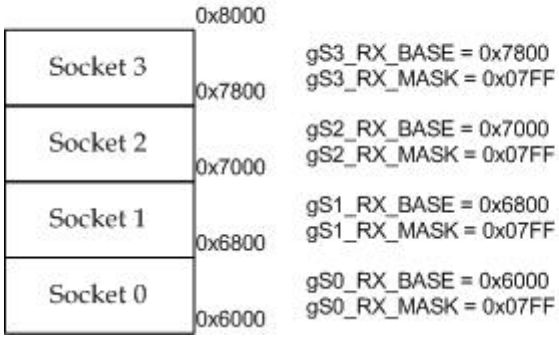
- Set socket memory information

This stage set the socket tx/rx memory information. Each socket base address, mask address are fixed and saved in this stage.

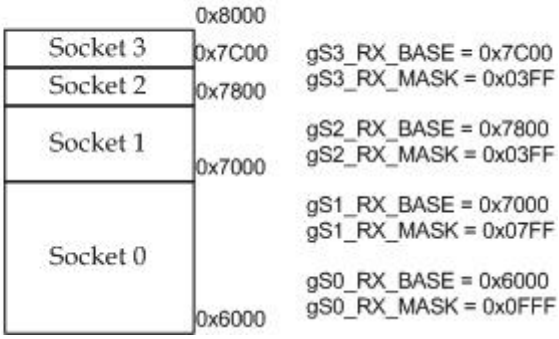
```
In case of, assign 2K rx memory per socket.
{
    RMSR = 0x55; // assign 2K rx memory per socket.
    gS0_RX_BASE = chip_base_address + RX_memory_base_address(0x6000);
    gS0_RX_MASK = 2K – 1 ; // 0x07FF, for getting offset address within assigned socket 0 RX memory.
    gS1_RX_BASE = gS0_BASE + (gS0_MASK + 1);
    gS1_RX_MASK = 2K – 1 ;
    gS2_RX_BASE = gS1_BASE + (gS1_MASK + 1);
    gS2_RX_MASK = 2K – 1 ;
    gS3_RX_BASE = gS2_BASE + (gS2_MASK + 1);
    gS3_RX_MASK = 2K – 1 ;
    TMSR = 0x55; // assign 2K tx memory per socket.
    Same method, set gS0_TX_BASE, gS0_TX_MASK, gS1_TX_BASE, gS1_TX_MASK, gS2_TX_BASE,
    gS2_TX_MASK, gS3_TX_BASE and gS3_TX_MASK.
}
```

```
In case of, assign 4K,2K,1K,1K.
{
    RMSR = 0x06; // assign 4K,2K,1K,1K rx memory per socket.
    gS0_RX_BASE = chip_base_address + RX_memory_base_address(0x6000);
    gS0_RX_MASK = 4K – 1 ; // 0x0FFF, for getting offset address within assigned socket 0 RX memory.
    gS1_RX_BASE = gS0_BASE + (gS0_MASK + 1);
    gS1_RX_MASK = 2K – 1 ; // 0x07FF
    gS2_RX_BASE = gS1_BASE + (gS1_MASK + 1);
    gS2_RX_MASK = 1K – 1 ; // 0x03FF
    gS3_RX_BASE = gS2_BASE + (gS2_MASK + 1);
    gS3_RX_MASK = 1K – 1 ; // 0x03FF
    TMSR = 0x06; // assign 4K,2K,1K,1K rx memory per socket.
    Same method, set gS0_TX_BASE, gS0_TX_MASK, gS1_TX_BASE, gS1_TX_MASK, gS2_TX_BASE,
    gS2_TX_MASK, gS3_TX_BASE and gS3_TX_MASK.
}
```

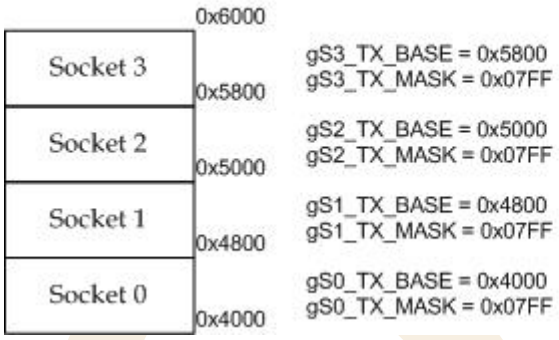
RMSR = 0x55, Chip Base Address = 0x0000



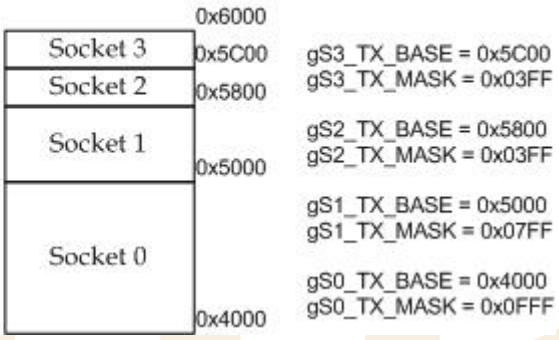
RMSR = 0x06



TMSR = 0x55, Chip Base Address = 0x0000



TMSR = 0x06



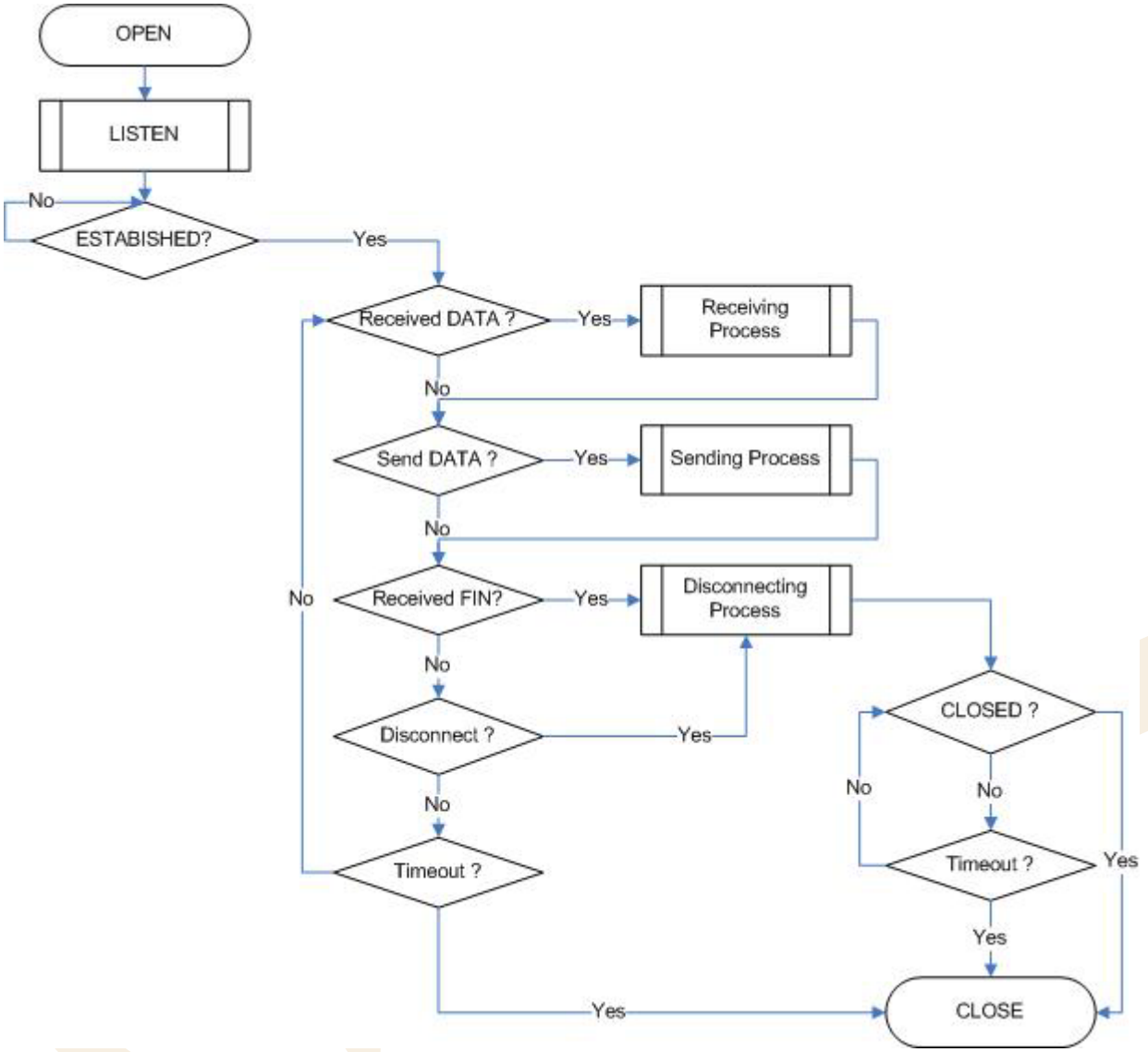
5.2. Data communication

Data communication is available through TCP or UDP. In order to select it, configure protocol field of Socket *n* Mode Register(S*n*_MR) of the commnucation sockets (W3150A supports total 4 sockets) as TCP(0x01) or UDP(0x02).

5.2.1. TCP

TCP is connection oriented communication method that will establish connection in advance and deliver the data through the connection by using IP Address and Port number of the systems. There are two methods to establish the connection. One is SERVER mode(passive open) that is waiting for connection request. The other is CLIENT mode(active open) that sends connection request to SERVER.

5.2.1.1. SERVER mode



■ Socket Initialization

It initializes the socket as TCP,

```
{
START:
    /* sets TCP mode */
    Sn_MR = 0x01;
    /* sets source port number */
    Sn_PORT = source_port;
    /* sets OPEN command */
    Sn_CR = OPEN;
    if (Sn_SSR != SOCK_INIT) Sn_CR = CLOSE; goto START;
}
```

■ LISTEN

In order to wait for a connection request.

```
{
    /* listen socket */
    Sn_CR = LISTEN;
    if (Sn_SSR != SOCK_LISTEN) Sn_CR = CLOSE; goto START; // check socket status
}
```

■ ESTABLISHED ?

If received connection request of remote peer (the stauts of SOCK_SYNRECV), W3150A sends ACK packet and changes to SOCK_ESTABLISHED status. This status can be checked as below.

```
First method :
{
    If (Sn_IR(CON bit) == '1') goto ESTABLISHED stage;
    /* In this case, if the interrupt of Socket n is activated, interrupt occurs. Refer to Interrupt
       Register(IR), Interrupt Mask Register (IMR) and Socket n Interrupt Register (Sn_IR). */
}
```

```
Second method :
{
    If (Sn_SSR == SOCK_ESTABLISHED) goto ESTABLISHED stage;
}
```

As connection is established, data transmission and receipt can be performed.

■ ESTABLISHED : Received Data ?

Check as below to know if data is received from remote peer or not.

```
First method :
{
    if (Sn_RX_RSR != 0x0000) goto Receving Process stage;
}
```

```
Second Method :
{
    If (Sn_IR(RECV bit) == '1') goto Receving Process stage;
    /* In this case, if the interrupt of Socket n is activated, interrupt occurs. Refer to Interrupt
       Register(IR), Interrupt Mask Register (IMR) and Socket n Interrupt Register (Sn_IR). */
}
```

■ ESTABLISHED : Receving Process

Recevied data can be processed as below.

```
{
    /* first, get the received size */
    get_size = Sn_RX_RSR;
    /* calculate offset address */
    get_offset = Sn_RX_RR & gSn_RX_MASK;
    /* calculate start address(physical address) */
    get_start_address = gSn_RX_BASE + get_offset;

    /* if overflow socket RX memory */
    if ( (get_offset + get_size) > (gSn_RX_MASK + 1) )
    {
        /* copy upper_size bytes of get_start_address to destination_addr */
        upper_size = (gSn_RX_MASK + 1) - get_offset;
        memcpy(get_start_address, destination_addr, upper_size);
        /* update destination_addr*/
        destination_addr += upper_size;
        /* copy left_size bytes of gSn_RX_BASE to destination_addr */
        left_size = get_size - upper_size;
        memcpy(gSn_RX_BASE, destination_addr, left_size);
    }
    else
    {
        /* copy get_size bytes of get_start_address to destination_addr */
        memcpy(get_start_address, destination_addr, get_size);
    }
    /* increase Sn_RX_RR as length of get_size */
    Sn_RX_RR += get_size;
    /* set RECV command */
    Sn_CR = RECV;
}
```

■ ESTABLISHED : Send DATA ? / Sending Process

The sending procedure is as below.

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

    /* calculate offset address */
    get_offset = Sn_TX_WR & gSn_TX_MASK;
    /* calculate start address(physical address) */
    get_start_address = gSn_TX_BASE + get_offset;

    /* if overflow socket TX memory */
    if ( (get_offset + send_size) > (gSn_TX_MASK + 1) )
    {
        /* copy upper_size bytes of source_addr to get_start_address */
        upper_size = (gSn_TX_MASK + 1) - get_offset;
        memcpy(source_addr, get_start_address, upper_size);
        /* update source_addr*/
        source_addr += upper_size;
        /* copy left_size bytes of source_addr to gSn_TX_BASE */
        left_size = send_size - upper_size;
        memcpy(source_addr, gSn_TX_BASE, left_size);
    }
    else
    {
        /* copy send_size bytes of source_addr to get_start_address */
        memcpy(source_addr, get_start_address, send_size);
    }
    /* increase Sn_TX_WR as length of send_size */
    Sn_TX_WR += send_size;
    /* set SEND command */
    Sn_CR = SEND;
}
```

■ ESTABLISHED : Received FIN?

Waiting for a connection termination request from remote peer.

It can be checked as below if it received connection termination request of remote peer.

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

```
Second method :
{
    If (Sn_SSR == SOCK_CLOSE_WAIT) goto CLOSED stage;
}
```

■ ESTABLISHED : Disconnect ? / Disconnecting Process

Check if user requests to terminate this connection.

To terminate the connection, proceed as below,

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

■ ESTABLISHED : CLOSED ?

No connection state at all. It can be checked as below,

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

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

■ ESTABLISHED : Timeout

In case that connection is closed due to the error of remote peer during data receving or connection closing process, data transmission can not be normally processed. At this time Timeout occurs after some time.

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

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

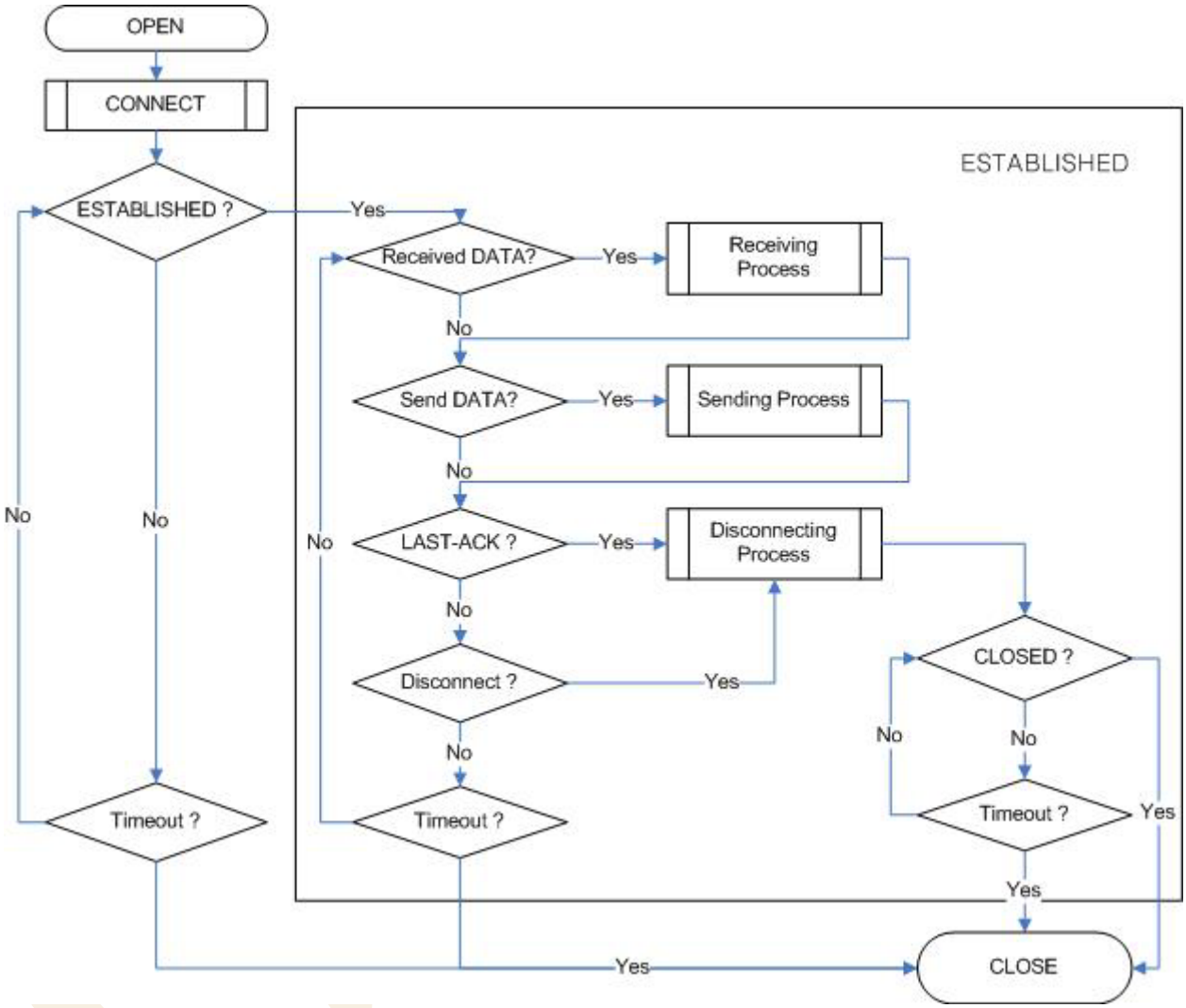
■ Socket Close

This process should be processed in case that connection is closed after data exchage, socket should be closed with Timeout occurrence, or forcible disconnection is necessary due to abonormal operation.

```
{
    /* set CLOSE command */
    Sn_CR = CLOSE;
}
```

5.2.1.2. CLIENT mode

Whole process is shown as below.



■ Socket Initialization

Refer to 5.2.1.1 SERVER (The operation is same as SERVER).

■ CONNECT

Sends connection request to remote HOST(SERVER) is as below.

```
{
    /* Write the value of server_ip, server_port to the Socket n Destination IP Address Register(Sn_DIPR),
       Socket n Destination Port Register(Sn_DPORT). */
    Sn_DIPR = server_ip;
    Sn_DPORT = server_port;
    /* set CONNECT command */
    Sn_CR = CONNECT;
}
```

■ ESTABLISHED ?

The connection is established. It can be checked as below,

```
First method :
{
    If (Sn_IR(CON bit) == '1') goto ESTABLISHED stage;
    /* In this case, if the interrupt of Socket n is activated, interrupt occurs. Refer to Interrupt
       Register(IR), Interrupt Mask Register (IMR) and Socket n Interrupt Register (Sn_IR). */
}
```

```
Second method :
{
    If (Sn_SSR == SOCK_ESTABLISHED) goto ESTABLISHED stage;
}
```

■ Timeout

Socket is closed as Timeout occurs as there is not response from remote peer. It can be checked as below.

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

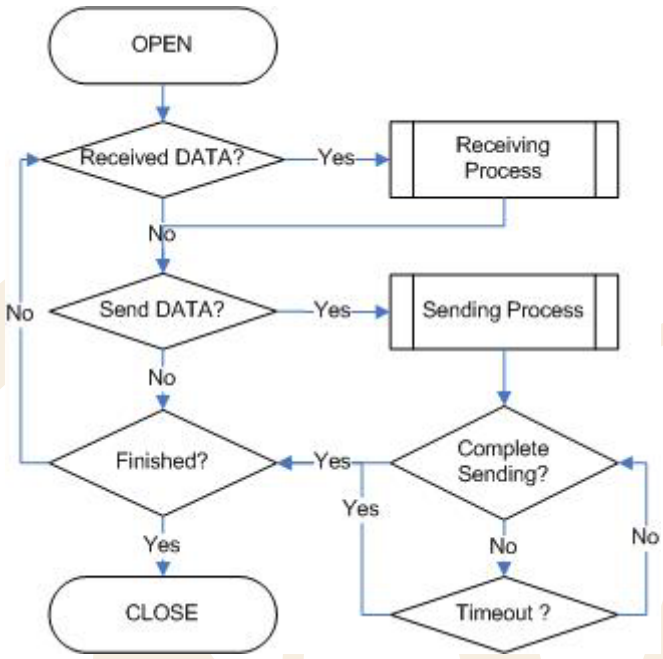
```
Second method :
{
    If (Sn_SSR == SOCK_CLOSED) goto CLOSED stage;
}
```

■ ESTABLISHED

Refer to 5.2.1.1. SERVER (The operation is same as SERVER mode)

5.2.2. UDP

UDP provides unreliable and connectionless datagram transmission structure. It processes data without connection establishment that TCP does. Therefore, UDP message can be lost, overlapped, reversed. As packets can arrive faster, recipient can not process all of them. In this case, user application should guarantee the reliability of data transmission. UDP transmission can be proceeded as below,



■ Socket Initialization

Initialize the socket as UDP.

```
{
START:
    /* sets UDP mode */
    Sn_MR = 0x02;
    /* sets source port number */
     /* ※ The value of Source Port can be appropriately delivered when remote HOST knows it. */
    Sn_PORT = source_port;
    /* sets OPEN command */
    Sn_CR = OPEN;
    /* Check if the value of Socket n Socket Status Register(Sn_SSR) is SOCK_UDP. */
    if (Sn_SSR != SOCK_UDP) Sn_CR = CLOSE; goto START;
}
```

■ Received DATA?

It can be checked as below if data is received from remote peer.

```
First method :
{
    if (Sn_RX_RSR != 0x0000) goto Receving Process stage;
}
```

```
Second Method :
{
    If (Sn_IR(RECV bit) == '1') goto Receving Process stage;
    /* In this case, if the interrupt of Socket n is activated, interrupt occurs. Refer to Interrupt
      Register(IR), Interrupt Mask Register (IMR) and Socket n Interrupt Register (Sn_IR). */
}
```

■ Receiving Process

Received data can be processed as below. In case of UDP, 8byte header is attached to receiving data. The structure of the header is as below.

IP Address (4)	Port (2)	Data size (2) (*data size except for 8byte of header)

```
{
    /* first, get the received size */
    get_size = Sn_RX_RSR;
    /* calculate offset address */
    get_offset = Sn_RX_RR & gSn_RX_MASK;
    /* calculate start address(physical address) */
    get_start_address = gSn_RX_BASE + get_offset;

    /* read head information (8 bytes) */
    header_size = 8;
    /* if overflow socket RX memory */
    if ( (get_offset + header_size) > (gSn_RX_MASK + 1) )
    {
        /* copy upper_size bytes of get_start_address to header_addr */
        upper_size = (gSn_RX_MASK + 1) – get_offset;
        memcpy(get_start_address, header_addr, upper_size);
        /* update header_addr*/
        header_addr += upper_size;
        /* copy left_size bytes of gSn_RX_BASE to header_addr */
```

```
        left_size = header_size – upper_size;
        memcpy(gSn_RX_BASE, header_addr, left_size);
        /* update get_offset */
        get_offset = left_size;
    }
    else
    {
        /* copy header_size bytes of get_start_address to header_addr */
        memcpy(get_start_address, header_addr, header_size);
        /* update get_offset */
        get_offset += header_size;
    }
    /* update get_start_address */
    get_start_address = gSn_RX_BASE + get_offset;

    /* save remote peer information & received data size */
    peer_ip = header[0 to 3];
    peer_port = header[4 to 5];
    get_size = header[6 to 7];

    /* if overflow socket RX memory */
    if ( (get_offset + get_size) > (gSn_RX_MASK + 1) )
    {
        /* copy upper_size bytes of get_start_address to destination_addr */
        upper_size = (gSn_RX_MASK + 1) – get_offset;
        memcpy(get_start_address, destination_addr, upper_size);
        /* update destination_addr*/
        destination_addr += upper_size;
        /* copy left_size bytes of gSn_RX_BASE to destination_addr */
        left_size = get_size – upper_size;
        memcpy(gSn_RX_BASE, destination_addr, left_size);
    }
    else
    {
        /* copy get_size bytes of get_start_address to destination_addr */
        memcpy(get_start_address, destination_addr, get_size);
    }
```

```
    /* increase Sn_RX_RR as length of get_size+header_size */
    Sn_RX_RR = Sn_RX_RR + get_size + header_size;
    /* set RECV command */
    Sn_CR = RECV;
}
```

■ Send Data? / Sending Process

Data transmission process is as below.

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

    /* Write the value of remote_ip, remote_port to the Socket n Destination IP Address Register(Sn_DIPR),
       Socket n Destination Port Register(Sn_DPORT). */
    Sn_DIPR = remote_ip;
    Sn_DPORT = remote_port;

    /* calculate offset address */
    get_offset = Sn_TX_WR & gSn_TX_MASK;
    /* calculate start address(physical address) */
    get_start_address = gSn_TX_BASE + get_offset;

    /* if overflow socket TX memory */
    if ( (get_offset + send_size) > (gSn_TX_MASK + 1) )
    {
        /* copy upper_size bytes of source_addr to get_start_address */
        upper_size = (gSn_TX_MASK + 1) - get_offset;
        memcpy(source_addr, get_start_address, upper_size);
        /* update source_addr*/
        source_addr += upper_size;
        /* copy left_size bytes of source_addr to gSn_TX_BASE */
        left_size = send_size - upper_size;
        memcpy(source_addr, gSn_TX_BASE, left_size);
    }
    else
```

```
    {
        /* copy send_size bytes of source_addr to get_start_address */
        memcpy(source_addr, get_start_address, send_size);
    }
    /* increase Sn_TX_WR as length of send_size */
    Sn_TX_WR += send_size;
    /* set SEND command */
    Sn_CR = SEND;
}
```

■ Complete Sending?

The sending completion should be checked after SEND command.

```
{
    If (Sn_CR == 0x00) transmission is completed.
}
```

■ Timeout

Timeout occurs if there is not remote peer or data transmission is not proceeded due to abnormal network. It can be checked as below.

```
{
    If (Sn_IR(TIMEOUT bit) == '1') goto next stage;
    /* In this case, if the interrupt of Socket n is activated, interrupt occurs. Refer to Interrupt
       Register(IR), Interrupt Mask Register (IMR) and Socket n Interrupt Register (Sn_IR). */
}
```

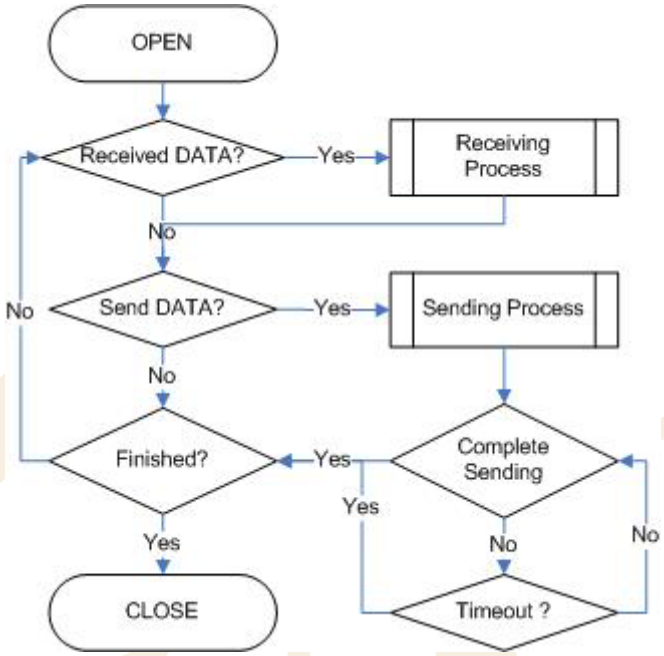
■ Finished? / Socket Close

If all the actions are finished, close the socket.

```
{
    /* set CLOSE command */
    Sn_CR = CLOSE;
}
```

5.2.3. IP raw

IP Raw mode can be utilized if transport layer protocol of some ICMP or IGMP that W3150A does not support, needs to be processed.



■ Socket Initialization

It initializes the socket as IP raw.

```
{
START:
    /* sets IP raw mode */
    Sn_MR = 0x03;
    /* sets Protocol value */
    /* The value of Protocol is the value used in Protocol Field of IP Header.
     For the list of protocol identification number of upper classification, refer to on line documents of
     IANA (http://www.iana.org/assignments/protocol-numbers). */
    Sn_PROTO = protocol_value;
    /* sets OPEN command */
    Sn_CR = OPEN;
    /* Check if the value of Socket n Socket Status Register(Sn_SSR) is SOCK_IPRAW. */
    if (Sn_SSR != SOCK_IPRAW) Sn_CR = CLOSE; goto START;
}
```

■ Received DATA?

This is same as UDP. Refer to 5.2.2 UDP.

■ Receiving Process

This is same as UDP. Refer to 5.2.2 UDP except the header information and header size.

In case of IP raw, 6byte header is attached to the data received. The header structure is as below.

IP Address (4)	Data Size (2) (*Data size except for 6 bytes of header)

■ Send DATA? / Sending Process

This is same as UDP. Refer to 5.2.2 UDP except that remote_port information is don't needed.

■ Complete Sending

■ Timeout

■ Finished? / Socket Closed

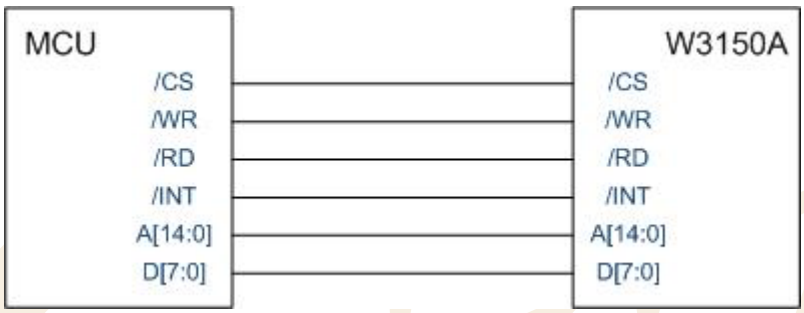
Next actions are same as UDP. Refer to 5.2.2 UDP.

6. Application Information

For the communication with MCU, W3150A provides Direct Bus I/F and Indirect Bus I/F mode. For the communication with Ethernet PHY, MII is used.

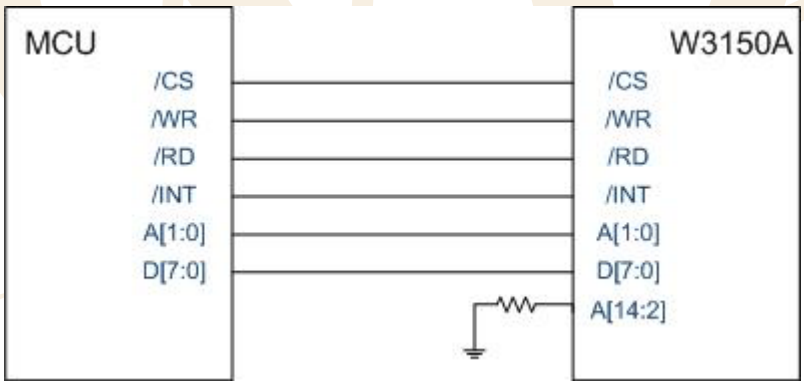
6.1. Direct Bus I/F Mode.

Direct Bus I/F mode uses 15bit address line and 8bit data line, /CS, /RD, /WR, /INT.



6.2. Indirect Bus I/F Mode.

Indirect Bus I/F mode uses 2bit address line and 8bit data line, /CS, /RD, /WR, /INT.
[14:2], other address lines should process Pull-down.



Indirect bus I/F mode related register is as below.

<table>
<tr><th>Value</th><th>Symbol</th><th>Description</th></tr>
<tr><td>0x00</td><td>MR</td><td>It performs the selection of Indirect bus I/F mode, Little/Big endian, and address automatic increase. Refer to 4. Register Description” for more detail.</td></tr>
<tr><td>0x01
0x02</td><td>IDM_AR0
IDM_AR1</td><td>Indirect bus I/F mode address Register
MSB/LSB can be decided by LB bit of MODE Register.
1. In case that LB bit is set to 0
0x01 | 0x02
IDM_AR0 : MSB | IDM_AR1 : LSB
Ex) In case of reading S0_CR(0x0401),
0x01(IDM_AR0) | 0x02(IDM_AR1)
0x04 | 0x01
2. In case that LB bit is set to 1
0x01 | 0x02
IDM_AR0 : LSB | IDM_AR1 : MSB
Ex) In case of reading S0_CR(0x0401),
0x01(IDM_AR0) | 0x02(IDM_AR1)
0x01 | 0x04</td></tr>
<tr><td>0x03</td><td>IDM_DR</td><td>Indirect bus I/F mode data Register</td></tr>
</table>

In order to read or write the internal register or internal TX/RX Memory,

1. Write the address to read or write on IDM_AR0,1.
2. Read or Write IDM_DR.

In order to read or write the data on the sequential address, set AI bit of MR(Mode Register). With this, user performs above 1 only one time. Whenever reading or writing IDM_DR, IDM_AR value increases by 1 automatically. So, the value can be processed on the sequential address just by continuous reading or writing of IDM_DR.

6.3. MII (Media Independent Interface)

The MII handles the data transfer between the W3150A and the Physical Layer Device.

The MII is composed of TX_CLK, TXE, and TXD[0:3] signals for sending data and RX_CLK, CRS, RXDV, RXD[0:3], and COL signals for receiving data.

When sending data from the W3150A, TXE and TXD[0:3] are output in synchronization with the falling edges of TX_CLK input from the Physical Layer Device because Physical Layer Devices generally recognize the rising edges of TX_CLK.

When receiving data, in general, the Physical Layer Devices output CRS, RXDV, RXD[0:3], and COL signals in synchronization with the falling edges of RX_CLK, so the W3150A recognizes the signals at the rising edges of RX_CLK.

7. Electrical Specification

7.1. 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
I_{IN}	DC input current	±5	mA
T_{OP}	Operating temperature	0 to 80	°C
T_{STG}	Storage temperature	-55 to 125	°C

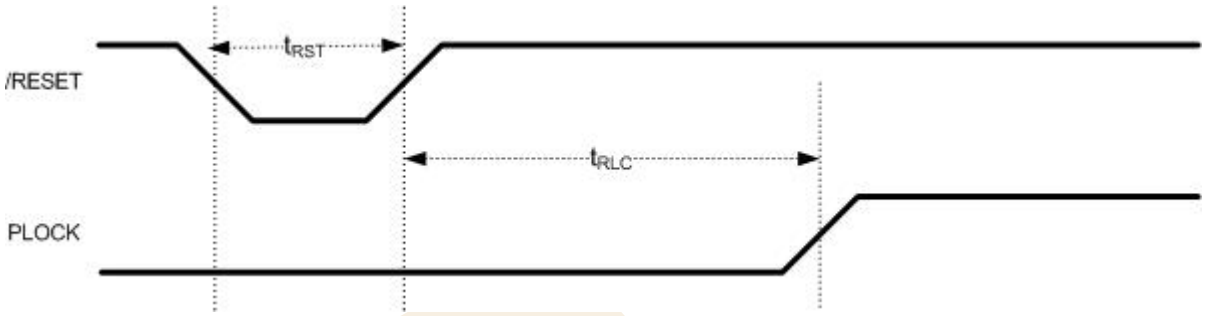
***COMMENT**: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage.

7.2. DC Characteristics

Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
V_{DD}	DC Supply voltage	Junction temperature is from -55C to 125C	3.0		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, 4, 8, 12, 16, 24 mA	2.0		3.6	V
V_{OL}	Low level output voltage	I_{OL} = -2, -4, -8, -12, -16, -24 mA	0.0		0.4	V
I_I	Input Current	$V_{IN} = V_{DD}$			±5	μA

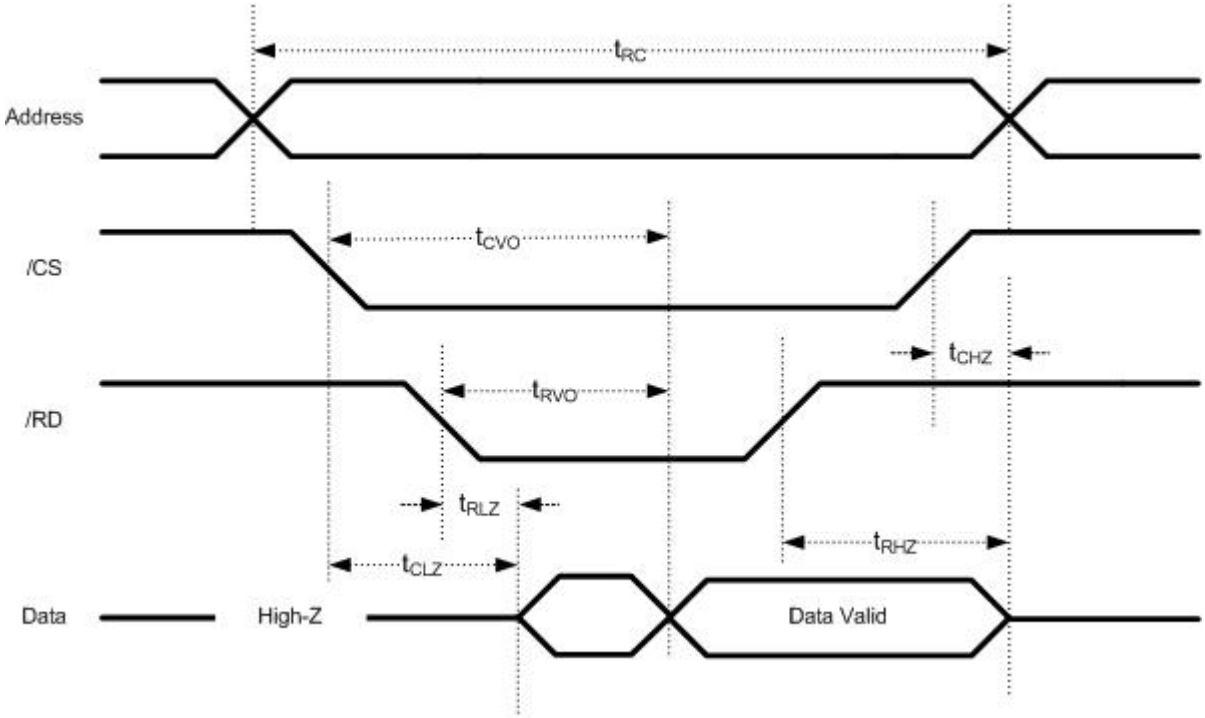
7.3. AC Characteristics

7.3.1. Reset Timing



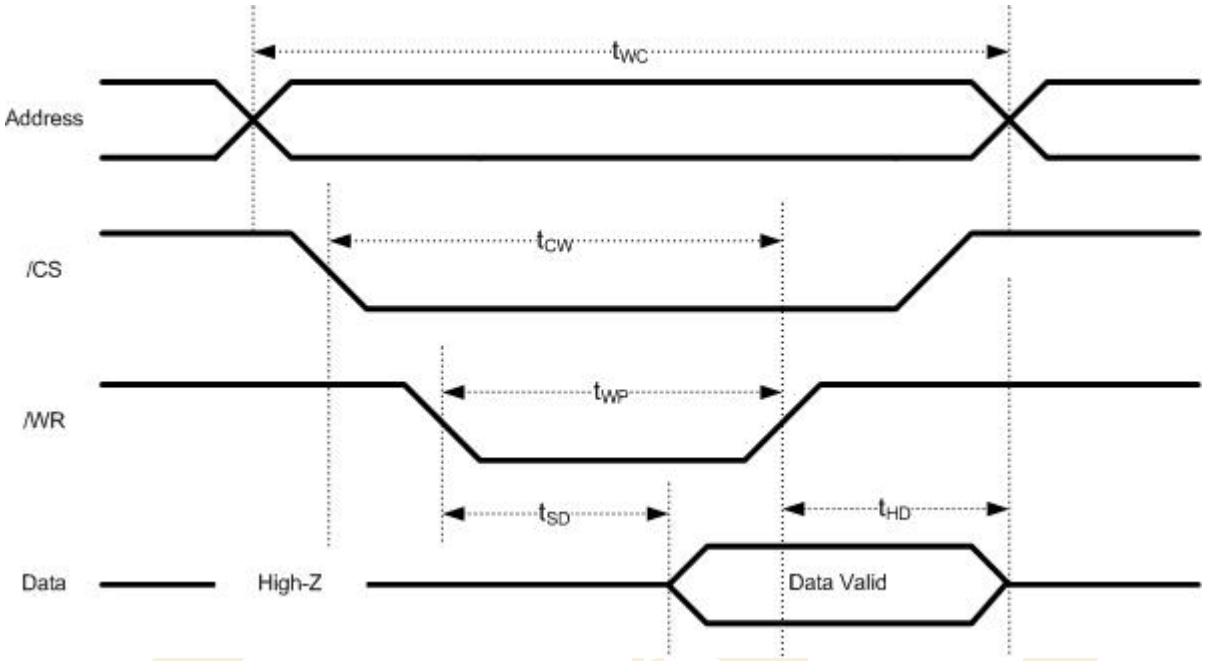
Symbol	Parameter	Min	Max
t_{RST}	Reset Cycle Time	96 ns	-
t_{RLC}	/RESET to PLOCK Output	-	10 ms

7.3.2. Register/Memory READ Timing



Symbol	Parameter	Min	Max
t_{RC}	Read Cycle Time	80 ns	-
t_{CVO}	/CS to Valid Output	-	80 ns
t_{RVO}	/RD to Valid Output	-	80 ns
t_{CLZ}	/CS to Low-Z Output	0 ns	-
t_{RLZ}	/RD to Low-Z Output	0 ns	-
t_{CHZ}	/CS to High-Z Output	-	1 ns
t_{RHZ}	/RD to High-Z Output	-	1 ns

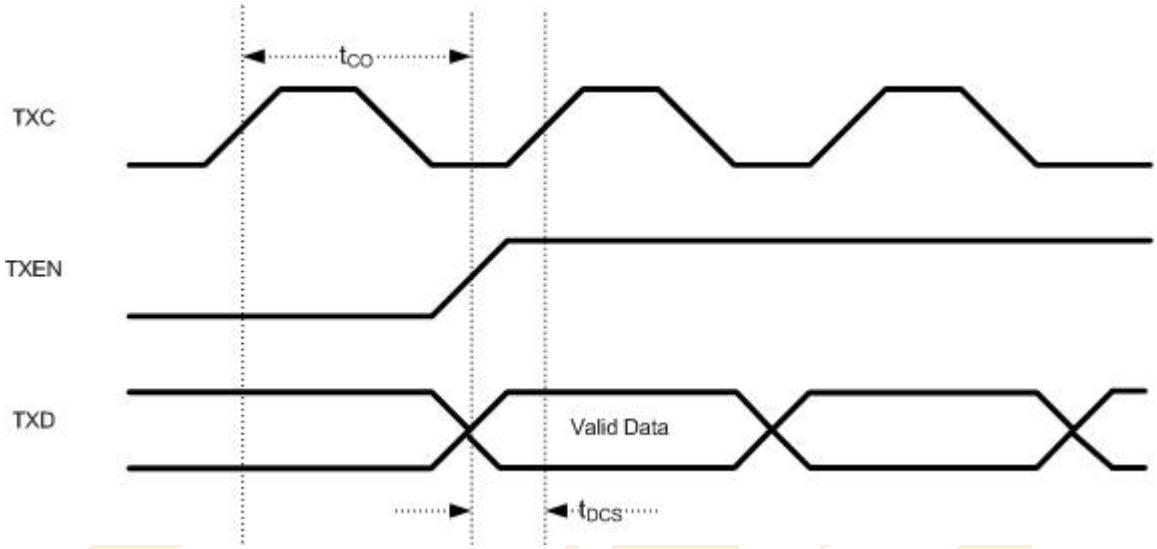
7.3.3. Register/Memory WRITE Timing



Symbol	Parameter	Min	Max
t_{WC}	Write Cycle Time	70 ns	-
t_{CW}	/CS to Write End	70 ns	-
t_{WP}	/WR Pulse width	63 ns	-
t_{SD}	/WR low to SD valid	-	14 ns
t_{HD}	Data Hold from Write End	0 ns	-

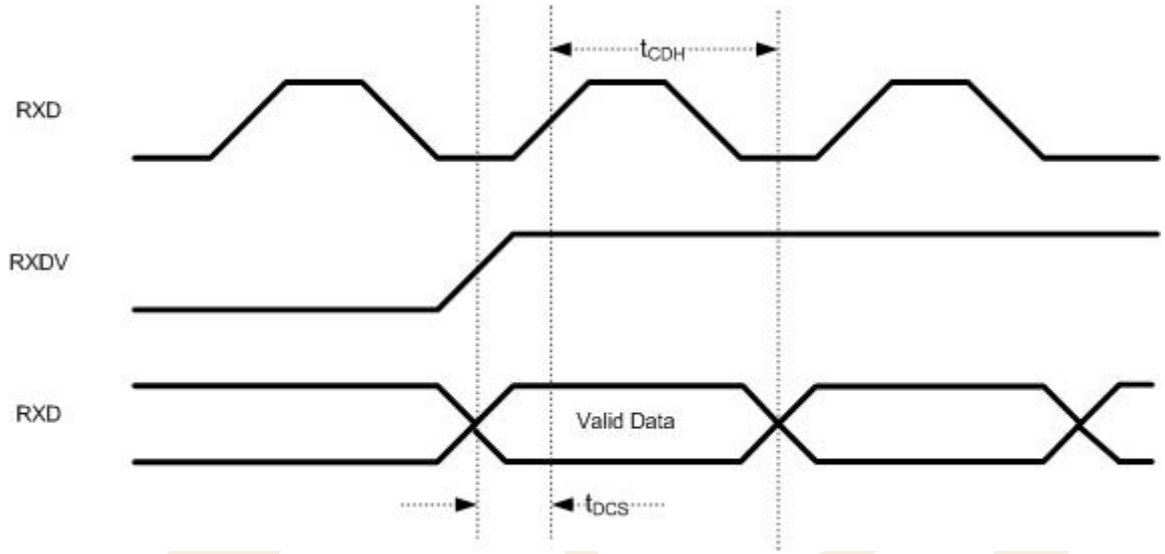
7.3.4. MII(Media Independent Interface) Timing

- MII Tx TIMING



Symbol	Parameter	Notes	Min	Typ	Max
t_{CO}	TX_CLK to TXD, TX_EN	10 Mbps	202 ns	-	205 ns
t_{DCS}	TXD, TX_EN setup time to TX_CLK	10 Mbps	195 ns	-	198 ns
t_{CO}	TX_CLK to TXD, TX_EN	100 Mbps	22 ns	-	25 ns
t_{DCS}	TXD, TX_EN setup time to TX_CLK	100 Mbps	15 ns	-	18 ns

■ MII Rx TIMING

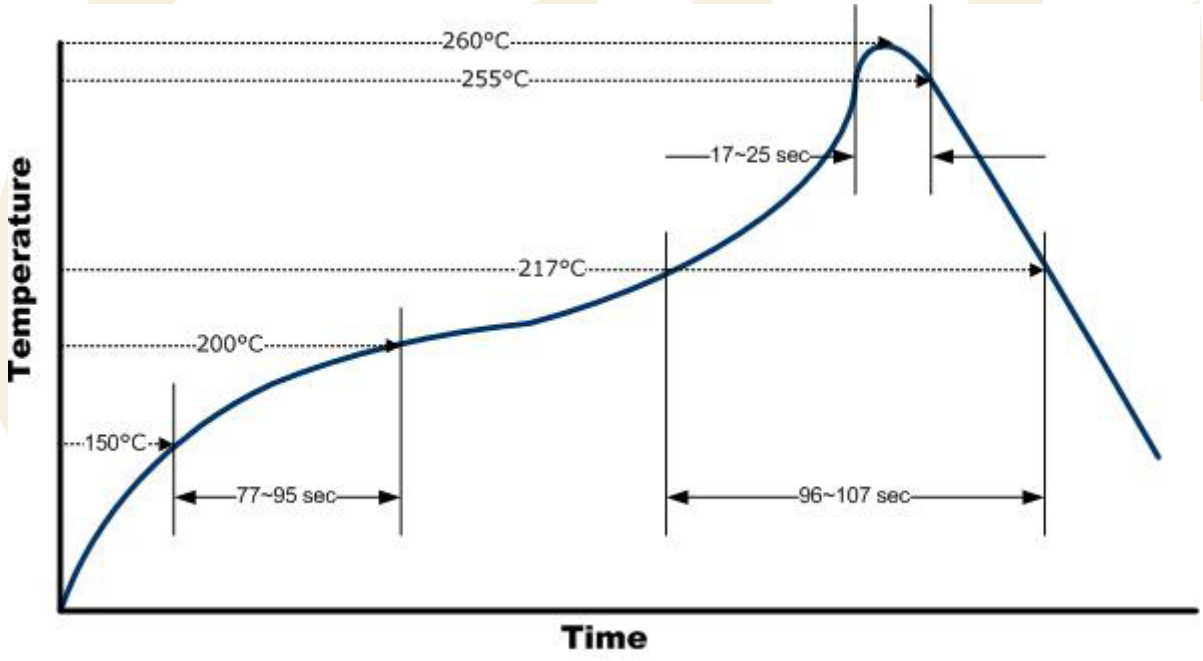


Symbol	Parameter	Notes	Min	Typ	Max
t_{DCS}	Valid Data to RX_CLK (setup)	10 Mbps	5 ns	-	-
t_{CDH}	RX_CLK to Valid Data (hold)	10 Mbps	5 ns	-	-
t_{DCS}	Valid Data to RX_CLK (setup)	100 Mbps	5 ns	-	-
t_{CDH}	RX_CLK to Valid Data (hold)	100 Mbps	5 ns	-	-

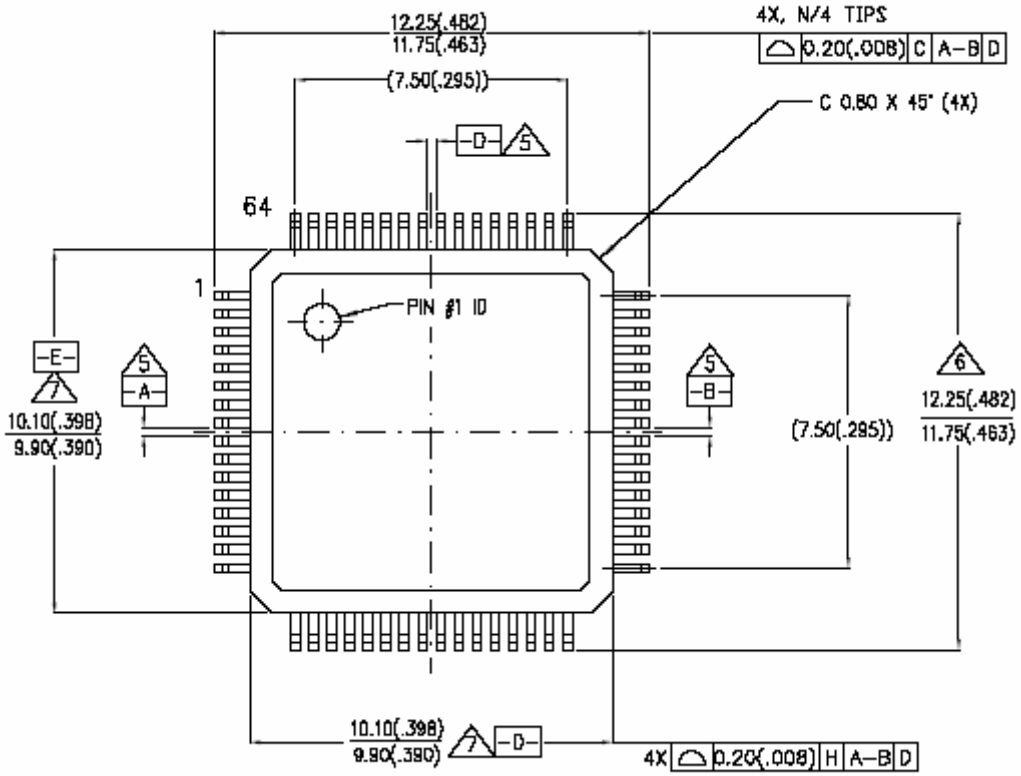
8. IR Reflow Temperature Profile (Lead-Free)

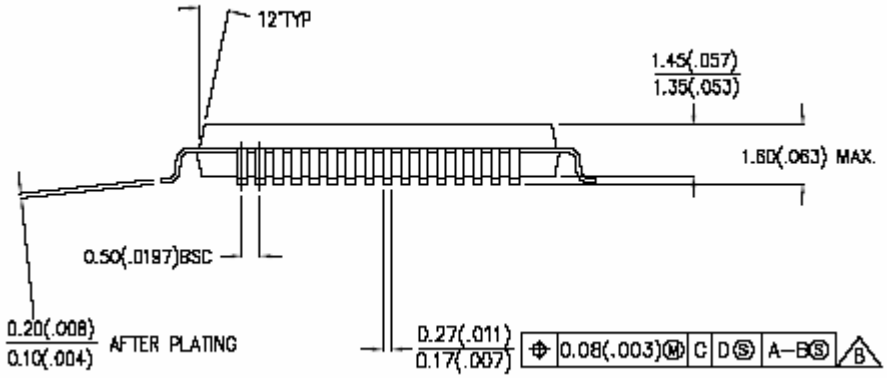
- Moisture Sensitivity Level at 260oC IR Condition: 2.
- Dry Bag Required: Yes
- 1 year out of bag time at max 30oC/60%RH.

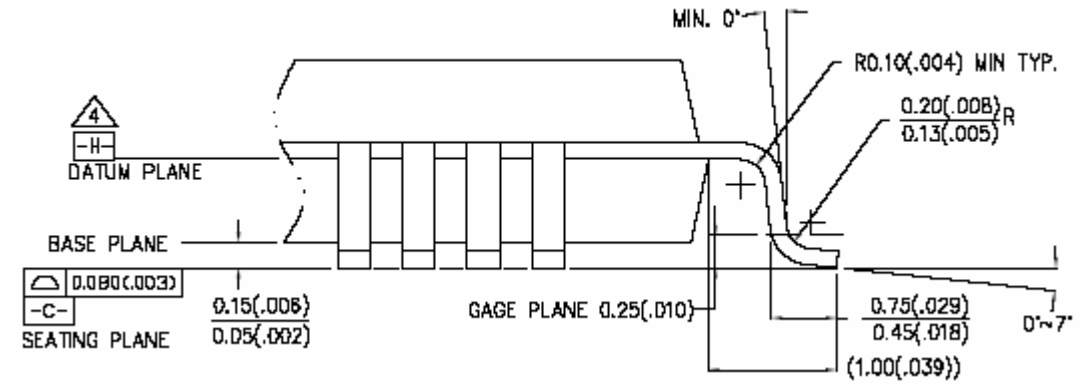
Max. Temperature 260oC	
Ramp up rate	< 3oC/second
Pre-heat temperature at 175°C(±25°C)	77-95 seconds
Temperature above 217°C	96-107 seconds
Time within 5°C of actual peak temperature	17-25 seconds
Peak temperature range	258-260oC
Ramp-down rate	< 6oC/second



9. Package Description







NOTES :

1. PACKAGE DIMENSIONS CONFORM TO JEDEC REGISTRATION MO – 136 – BCD.
2. CONTROLLING DIMENSIONS : MILLIMETERS. INCH ARE SHOWN IN PARENTHESES.
3. DIMENSIONS AND TOLERANCING PER ANSI Y 14.5–1982.
4. DATUM PLANE "H" IS LOCATED AT MOLD PARTING LINE AND IS COINCIDENT WITH THE LEAD EXITS THE PLASTIC BODY AT BOTTOM OF THE PARTING LINE.
5. DATUMS "A–B" AND "D" TO BE DETERMINED AT DATUM PLANE "H".
6. TO BE DETERMINED AT THE SEATING PLANE "C"
7. THESE DIMENSIONS TO BE DETERMINED AT DATUM PLANE "H". DIMENSIONS D AND E DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25MM(.010") PER SIDE.
8. LEAD WIDTH DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 MM/0.003" TOTAL IN EXCESS OF THIS DIMENSIONS AT MAXIMUM MATERIAL CONDITION.