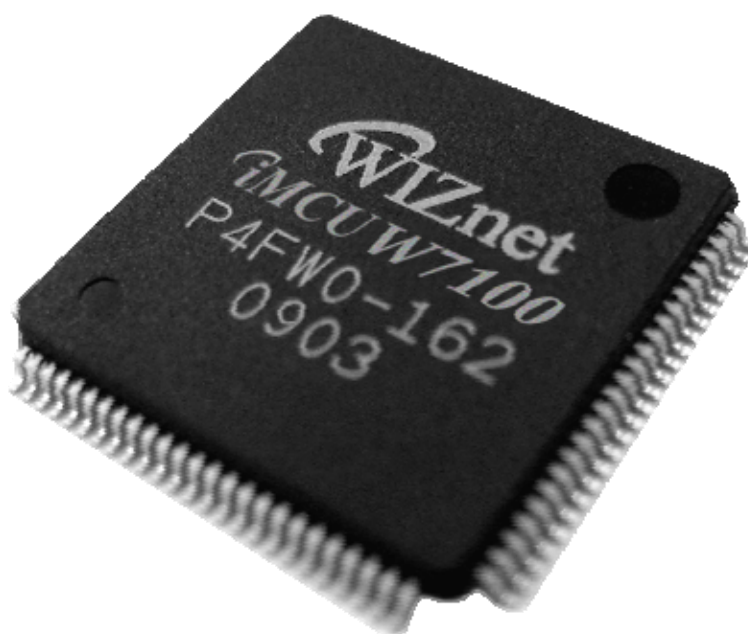


Internet Embedded MCU W7100 Datasheet

Version 0.9.3



© 2009 WIZnet Co., Inc. All Rights Reserved.

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

Document History Information

Version	Date	Descriptions
Ver.0.9 Beta	Sep. 2009	Release with W7100 launching
Ver.0.9.2	Dec. 2009	Delete about "How to program FLASH memory in W7100" document Section 2.4.7, "APP Entry RD/WR Enable" => "APP Entry(0xFFFF7 ~ 0xFFFF) RD/WR Enable" Section 3 I/O Ports, Delete about three-state signals in I/O pins
Ver.0.9.3	Feb. 2010	Modify the XTLP0 and XTLP1 explanation

Copyright Notice

Copyright 2009 WIZnet, Inc. All Rights Reserved.

Technical Support: support@wiznet.co.kr

Sales & Distribution: sales@wiznet.co.kr

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

WIZnet's online Technical Support

If you have any questions regarding WIZnet's Products, please write down your question on our [Q&A Board](#) under the 'Support' menu in the WIZnet website (www.wiznet.co.kr). We will respond to your questions as soon as possible.



The screenshot shows the WIZnet website interface. At the top, there is a navigation bar with links: HOME, ABOUT, JOIN, CONTACT US, CHANGES, PARTNERSHIP, VIDEOS, and Service Mail. Below this, a sidebar on the left contains a menu with links: PRODUCTS, TECHNOLOGY, **Q&A**, LIBRARY, DISTRIBUTOR, PARTNERSHIP, BLOG, and ABOUT US. A red circle with the word "Click" points to the "Q&A" link. The main content area features a banner for "WIZ200WEB" (Embedded Web Server Module With AVR & W5300) and "WIZ830MJ". Below the banner, there are sections for "WHERE TO BUY", "NEW PRODUCT" (W5300), "WIZnet SEARCH", and "WHAT'S UPDATED". A large red arrow points from the "Q&A" link in the sidebar to the "Q&A" section below. The "Q&A" section has a sub-menu with "Q&A", "FAQ", and "E-FORUM". The main content of the "Q&A" section shows a list of questions and answers. The table below is a representation of the data shown in the screenshot.

NO	SUBJECT	NAME	DATE	VIEW
2776	Conditions for WIZ1100R stops working	Fernando Blom	2009-09-01	0
2775	Regarding WIZ1100R	Michael Smith	2009-09-01	15
2774	W5300 non-linked state	Bole Rosenbom	2009-09-01	0
2773	2772 continued	Ben	2009-09-01	11
2772	Reply to 2772	Steve	2009-09-01	2
2771	W5100 conditions for remote data link	Steve	2009-09-01	12
2770	[Re] W5100 conditions for remote data link	WIZnet	2009-09-01	1
2769	Sent to Ethernet Tutorial	Ben	2009-09-01	26
2768	[Re] Sent to Ethernet Tutorial	WIZnet	2009-09-01	11
2767	About answer 2764 interface with C/C++	Bin Tulp	2009-09-01	10
2766	[Re] About answer 2764 interface with C/C++	WIZnet	2009-09-01	2

Table of Contents

1	Overview	11
1.1	Introduction.....	11
1.2	W7100 Features	11
1.3	W7100 Block Diagram & Features	12
1.3.1	ALU (Arithmetic Logic Unit)	12
1.3.2	TCPIPCore	14
1.4	Pin Description	16
1.4.1	Pin Layout	16
1.4.2	Pin Description.....	17
1.4.2.1	Configuration.....	17
1.4.2.2	Timer	18
1.4.2.3	UART.....	18
1.4.2.4	DoCD™ Compatible Debugger	18
1.4.2.5	Interrupt / Clock	18
1.4.2.6	GPIO	19
1.4.2.7	Media Interface	20
1.4.2.8	Network Indicator LED.....	20
1.4.2.9	Power Supply Signal.....	21
2	Memory	23
2.1	Code Memory	23
2.1.1	Code Memory Wait States	26
2.2	Data Memory.....	26
2.2.1	Data Memory Wait States	26
2.3	Internal Data Memory and SFR.....	27
2.4	SFR definition.....	28
2.4.1	Program Code Memory Write Enable Bit	28
2.4.2	Program Code Memory Wait States Register	29
2.4.3	Data Pointer Extended Registers.....	30
2.4.4	Data Pointer Registers	31
2.4.5	Clock Control Register.....	32
2.4.6	Stack Pointer.....	32
2.4.7	New & Extended SFR.....	33
2.4.8	Peripheral Registers.....	33
3	Interrupt.....	35
4	I/O Ports.....	39

5	Timers.....	41
5.1	Timers 0, 1	41
5.1.1	Overview.....	41
5.1.2	Interrupts.....	42
5.1.3	Timer0 - Mode0	43
5.1.4	Timer0 - Mode1	44
5.1.5	Timer0 - Mode2	44
5.1.6	Timer0 - Mode3	44
5.1.7	Timer1 - Mode0	45
5.1.8	Timer1 - Mode1	46
5.1.9	Timer1 - Mode2	46
5.1.10	Timer1 - Mode3	46
5.2	Timer2	47
5.2.1	Overview.....	47
5.2.2	Interrupts.....	48
6	UART	51
6.1	Interrupts.....	52
6.2	Mode0, Synchronous	53
6.3	Mode1, 8-Bit UART, Variable Baud Rate, Timer 1 or 2 Clock Source	54
6.4	Mode2, 9-Bit UART, Fixed Baud Rate.....	54
6.5	Mode3, 9-Bit UART, Variable Baud Rate, Timer1 or 2 Clock Source	55
6.6	Examples of Baud Rate Setting	55
7	Watchdog Timer	56
7.1	Overview	56
7.2	Interrupts.....	56
7.3	Watchdog Timer Reset.....	57
7.4	Simple Timer.....	58
7.5	System Monitor.....	58
7.6	Watchdog Related Registers	58
7.7	Watchdog Control	59
7.7.1	Clock Control.....	60
7.8	Timed Access Registers.....	61
8	TCIPCore	62
8.1	Memory Map	62
8.2	TCIPCore Registers	62
8.2.1	Common Registers.....	62
8.2.2	SOCKET Registers	64

8.3	Register Description	76
8.3.1	Mode Register	76
8.3.2	SOCKET Registers	81
9	Functional Description	99
9.1	Initialization	99
9.2	Data Communication	104
9.2.1	TCP	104
9.2.1.1	TCP SERVER.....	105
9.2.1.2	TCP CLIENT	112
9.2.2	UDP	113
9.2.2.1	Unicast & Broadcast.....	113
9.2.2.2	Multicast	119
9.2.3	IPRAW	122
9.2.4	MACRAW.....	124
10	Electrical Specification	130
11	IR Reflow Temperature Profile (Lead-Free)	134
12	Package Descriptions	135
13	Appendix: Performance Improvement about W7100	137
13.1	Summary.....	137
13.2	8-Bit Arithmetic Functions.....	137
13.2.1	Addition.....	137
13.2.2	Subtraction	138
13.2.3	Multiplication	140
13.2.4	Division.....	140
13.3	16-Bit Arithmetic Functions	140
13.3.1	Addition.....	140
13.3.2	Subtraction	141
13.3.3	Multiplication	141
13.4	32-bit Arithmetic Functions	142
13.4.1	Addition.....	142
13.4.2	Subtraction	143
13.4.3	Multiplication	143

List of Figures

Figure 1.1 W7100 Block Diagram	12
Figure 1.2 Accumulator A Register	13
Figure 1.3 B Register	13
Figure 1.4 Program Status Word Register	13
Figure 1.5 PSW Register	13
Figure 1.6 TCPIPCore Block Diagram	14
Figure 1.7 W7100 Pin Layout.....	16
Figure 1.8 Power Design	22
Figure 2.1 Code / Data Memory Connections	23
Figure 2.2. Boot Sequence Flowchart	24
Figure 2.3 “Code Memory” Map Switching	24
Figure 2.4 APP Entry Process.....	25
Figure 2.5 Changing the code memory Status at RB = ‘0’	25
Figure 2.6 “Data Memory” Map.....	27
Figure 2.11 Internal Memory Map	27
Figure 2.12 SFR Memory Map	28
Figure 2.13 PWE bit of PCON Register.....	28
Figure 2.14 Code memory Wait States Register.....	29
Figure 2.15 Waveform for code memory Synchronous Read Cycle with Minimal Wait States (WTST = ‘3’)	29
Figure 2.16 Waveform for code memory Synchronous Write Cycle with Minimal Wait States(WTST = ‘3’).....	30
Figure 2.17 Data Pointer Extended Register.....	30
Figure 2.18 Data Pointer Extended Register.....	30
Figure 2.19 MOVX @RI Extended Register.....	30
Figure 2.20 Data Pointer Register DPTR0	31
Figure 2.21 Data Pointer 1 Register DPTR1.....	31
Figure 2.22 Data Pointer Select Register	31
Figure 2.23 Clock Control Register - STRETCH bits.....	32
Figure 2.24 Stack Pointer Register.....	32
Figure 2.26 W7100 Configuration Register.....	33
Figure 3.1 Interrupt Enable Register	36
Figure 3.2 Interrupt Priority Register.....	36
Figure 3.3 Timer0, 1 Configuration Register	36
Figure 3.4 UART Configuration Register.....	37

Figure 3.5 Extended Interrupt Enable Register	37
Figure 3.6 Extended Interrupt Priority Register	37
Figure 3.7 Extended Interrupt Flag Register	38
Figure 3.8 Watchdog Control Register.....	38
Figure 4.1 Port0 Register	39
Figure 4.2 Port1 Register	39
Figure 4.3 Port2 Register	39
Figure 4.4 Port3 Register	39
Figure 4.5 Usage of the GPIO pins in the W7100	40
Figure 5.1 Timer0, 1 Control Mode Register.....	41
Figure 5.2 Timer0, 1 Configuration Register	42
Figure 5.3 Interrupt Enable Register	42
Figure 5.4 Interrupt Priority Register.....	42
Figure 5.5 Timer0, 1 Configuration Register	43
Figure 5.6 Timer Counter0, Mode0: 13-Bit Timer/Counter	43
Figure 5.7 Timer/Counter0, Mode1: 16-Bit Timer/Counter.....	44
Figure 5.8 Timer/Counter0, Mode2: 8-Bit Timer/Counter with Auto-Reload	44
Figure 5.9 Timer/Counter0, Mode3: Two 8-Bit Timers/Counters	45
Figure 5.10 Timer/Counter1, Mode0: 13-Bit Timer/Counter	45
Figure 5.11 Timer/Counter1, Mode1: 16-Bit Timers/Counters	46
Figure 5.12 Timer/Counter1, Mode2: 8-Bit Timer/Counter with Auto-Reload	46
Figure 5.13 Timer2 Configuration Register	47
Figure 5.14 Timer/Counter2, 16-Bit Timer/Counter with Auto-Reload	48
Figure 5.15 Interrupt Enable Register – Timer2	48
Figure 5.16 Interrupt Priority Register – Timer2	49
Figure 5.17 Timer2 Configuration Register – TF2	49
Figure 5.18 Timer/Counter2, 16-Bit Timer/Counter with Capture Mode	49
Figure 5.19 Timer2 for Baud Rate Generator Mode	50
Figure 6.1 UART Buffer Register.....	51
Figure 6.2 UART Configuration Register.....	51
Figure 6.3 UART Bits in Power Configuration Register.....	52
Figure 6.4 UART Bits in Interrupt Enable Register.....	53
Figure 6.5 UART Bits in Interrupt Priority Register.....	53
Figure 6.6 UART Configuration Register.....	53
Figure 6.7 Timing Diagram for UART Transmission Mode0 (clk = 88.4736 MHz).....	54
Figure 6.8 Timing Diagram for UART Transmission Mode1	54
Figure 6.9 Timing Diagram for UART Transmission Mode2.....	54

Figure 6.10 Timing Diagram for UART Transmission Mode3	55
Figure 7.1 Watchdog Timer Structure	56
Figure 7.2 Interrupt Enable Register	56
Figure 7.3 Extended Interrupt Enable Register	56
Figure 7.4 Extended interrupt Priority Register	57
Figure 7.5 Watchdog Control Register.....	57
Figure 7.6 Watchdog Control Register.....	59
Figure 7.7 Clock Control register - Watchdog bits.....	60
Figure 8.1 TCPIPCore Memory Map	62
Figure 8.2 SOCKET <i>n</i> Status transition.....	88
Figure 8.3 Calculate Physical Address	95
Figure 9.1 Allocation Internal TX/RX memory of SOCKET <i>n</i>	103
Figure 9.2 TCP SERVER & TCP CLIENT	104
Figure 9.3 “TCP SERVER” Operation Flow	105
Figure 9.5 “TCP CLIENT” Operation Flow.....	112
Figure 9.6 UDP Operation Flow.....	113
Figure 9.7 The received UDP data format	115
Figure 9.8 IPRAW Operation Flow	122
Figure 9.9 The received IPRAW data format	123
Figure 9.10 MACRAW Operation Flow.....	124
Figure 9.11 The received MACRAW data format	125
Figure 10.1 Waveform for data memory Read Cycle with Minimal Wait States(CKCON = ‘2’)	131
Figure 10.2 Waveform for data memory Write Cycle with Minimal Wait States(CKCON = ‘2’)	132

List of Tables

Table 2.1 WTST Register Values	29
Table 2.2 DPTR0, DPTR1 Operations.....	31
Table 2.3 MD[2:0] Bit Values	32
Table 3.1 External Interrupt Pin Description	35
Table 3.2 W7100 Interrupt Summary	35
Table 4.1 I/O Ports Pin Description	39
Table 4.2 Read-Modify-Write Instructions.....	39
Table 5.1 Timers 0, 1 Pin Description	41
Table 5.2 Timers 0, 1 Mode	41
Table 5.3 Timer0, 1 interrupts.....	43
Table 5.4 Timer2 Pin Description.....	47
Table 5.5 Timer2 Modes	47
Table 5.6 Timer2 Interrupt	49
Table 6.1 UART Pin Description.....	51
Table 6.2 UART Modes.....	52
Table 6.3 UART Baud Rates.....	52
Table 6.4 UART Interrupt	53
Table 6.5 Examples of Baud Rate Setting	55
Table 7.1 Watchdog Interrupt	57
Table 7.2 Summary for Watchdog Related Bits.....	58
Table 7.4 Watchdog Intervals	60
Table 7.5 Timed Access Registers	61
Table 12.1 Timer / Counter Mode.....	100
Table 12.2 Baud rate	100
Table 12.3 Mode of UART.....	100

1 Overview

1.1 Introduction

iMCU W7100 is the one-chip solution which integrates an 8051 compatible microcontroller, 64KB SRAM and hardwired TCP/IP Core for high performance and easy development.

The TCP/IP core is a market-proven hardwired TCP/IP stack with an integrated Ethernet MAC & PHY. The Hardwired TCP/IP stack supports the TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE which has been used in various applications for years. The W7100 comes with a 32 Kbytes internal buffer for data transmission and reception.

1.2 W7100 Features

- Fully software compatible with industrial standard 8051
- Pipelined architecture which enables execution of instructions 4-5 times faster than a standard 8051
- 2 Data Pointers (DPTRs) for fast memory blocks processing
 - Advanced INC & DEC modes
 - Auto-switch of current DPTR
- 64KBytes Data Memory (RAM)
- 256Bytes data FLASH
- 64KBytes Code Memory
- 2KBytes Boot Code Memory
- Interrupt controller
 - 2 priority levels
 - 4 external interrupt sources
 - 1 Watchdog interrupt
- Four 8-bit I/O Ports
- Three timers/counters
- Full-duplex UART
- Programmable Watchdog Timer
- DoCD™ compatible debugger
- Hardwired TCP/IP Protocols: TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet
- 10BaseT/100BaseTX Ethernet PHY embedded
- Auto Negotiation (Full-duplex and half duplex)
- Auto MDI/MDIX
- ADSL connection with PPPoE Protocol with PAP/CHAP Authentication mode support
- 8 independent sockets which are running simultaneously
- 32Kbytes Data buffer for the Network

- Network status LED outputs (TX, RX, Full/Half duplex, Collision, Link, and Speed)
- Not supports IP fragmentation

1.3 W7100 Block Diagram & Features

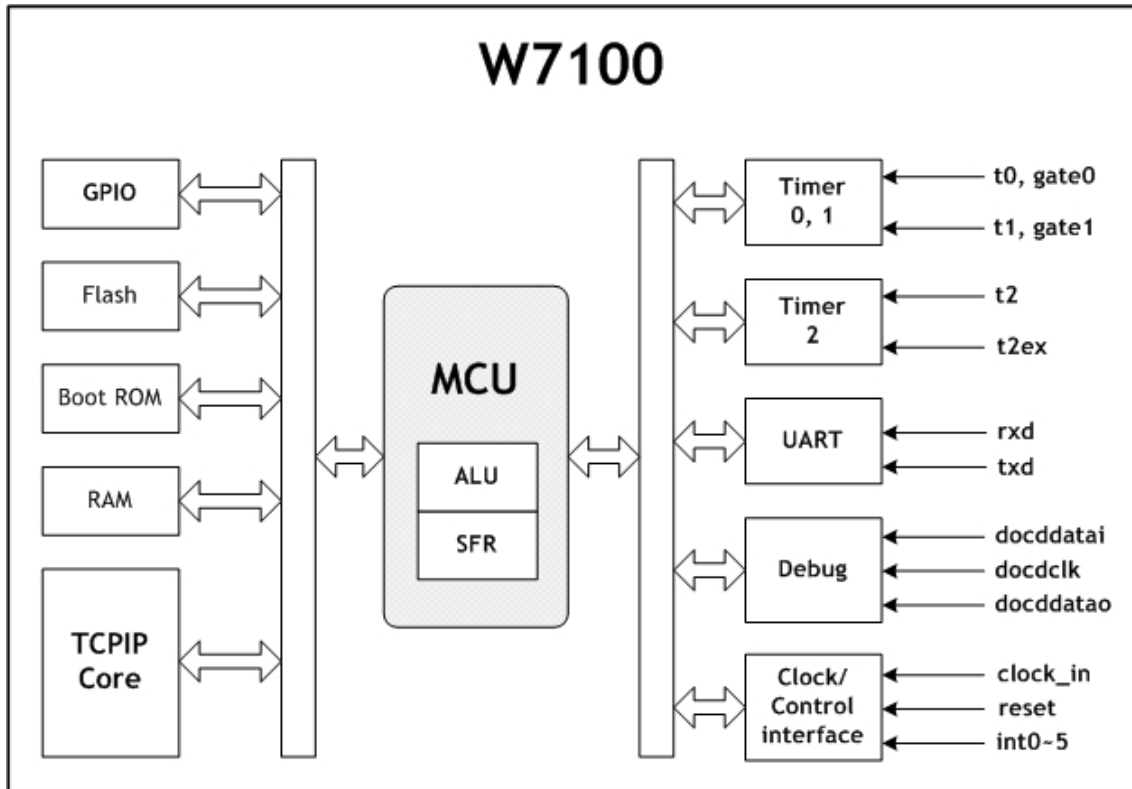


Figure 1.1 W7100 Block Diagram

The W7100 internal block diagram is shown in the Figure 1.1. Details of block functions are described as follows:

ALU - Performs arithmetic and logic operations during execution of an instruction. It contains accumulator (ACC), Program Status Word (PSW), B registers, and related logics such as arithmetic unit, logic unit, multiplier, and divider.

SFR - Controls the access of special registers. It contains standard and user defined registers and related logic. User defined external devices can be quickly accessed (read, write, modified) using all direct addressing mode instructions.

1.3.1 ALU (Arithmetic Logic Unit)

W7100 is fully compatible with the standard 8051 microcontroller, and maintains all instruction mnemonics and binary compatibility. W7100 incorporates many great architectural enhancements which enable the W7100 MCU to execute instructions with high speed.

The ALU of W7100 MCU performs extensive data manipulation and is comprised of the 8-bit arithmetic logic unit (ALU), an ACC (0xE0) register, a B (0xF0) register and a PSW (0xD0)

register

ACC (0xE0)								
7	6	5	4	3	2	1	0	Reset
ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0	0x00

Figure 1.2 Accumulator A Register

The B register is used during multiplication and division operations. In other cases, this register is used as normal SFR.

B (0xF0)								
7	6	5	4	3	2	1	0	Reset
B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	0x00

Figure 1.3 B Register

The ALU performs arithmetic operations such as addition, subtraction, multiplication, and division, and other operations such as increment, decrement, BCD-decimal-add-adjust, and compare. Logic unit uses AND, OR, Exclusive OR, complement, and rotation to perform different operations. The Boolean processor performs bit operations such as set, clear, complement, jump-if-not-set, jump-if-set-and-clear, and move to/from carry.

PSW (0xD0)								
7	6	5	4	3	2	1	0	Reset
CY	AC	F0	RS1	RS0	OV	F1	P	0x00

Figure 1.4 Program Status Word Register

CY	Carry flag	
AC	Auxiliary carry	
F0	General purpose flag 0	
RS[1:0]	Register bank select bits	
	RS[1:0]	Function Description
	00	-Bank 0, data address 0x00 - 0x07
	01	-Bank 1, data address 0x08 - 0x0F
	10	-Bank 2, data address 0x10 - 0x17
	11	-Bank 3, data address 0x18 - 0x1F
OV	Overflow flag	
F1	General purpose flag 1	
P	Parity flag	

Figure 1.5 PSW Register

The PSW register contains several bits that can reflect the current state of MCU.

1.3.2 TCIPCore

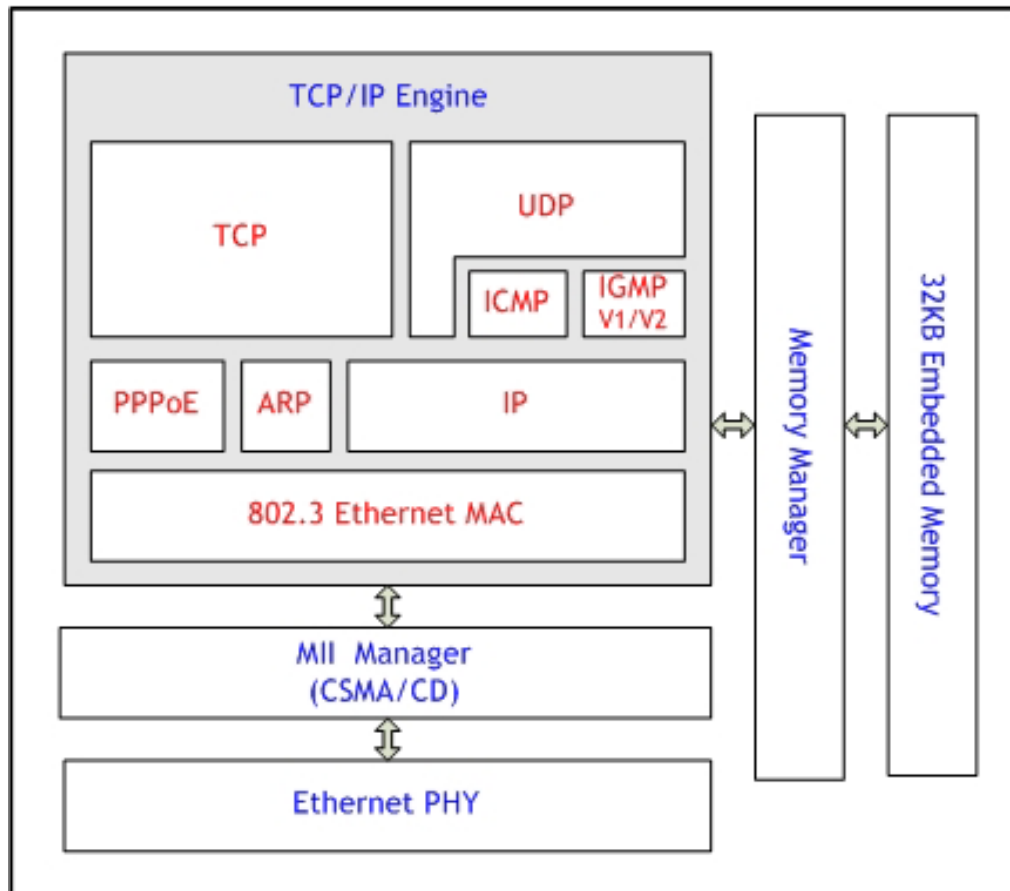


Figure 1.6 TCIPCore Block Diagram

Ethernet PHY

The W7100 includes 10BaseT/100BaseTX Ethernet PHY. It supports half-duplex/full duplex, auto-negotiation and auto MDI/MDIX. It also supports 6 network indicator LED outputs such as Link, TX, RX status, Collision, speed and duplex.

TCIP Engine

TCIP Engine is a hardwired logic based network protocol which contains WIZnet's technology.

- **802.3 Ethernet MAC(Media Access Control)**

This controls Ethernet access of CSMA/CD(Carrier Sense Multiple Access with Collision Detect). The protocol is based on a 48-bit source/destination MAC address.

- **PPPoE(Point-To-Point Protocol over Ethernet)**

This protocol uses PPP service over Ethernet. The payload (PPP frame) is encapsulated inside an Ethernet frame during a transmission. When receiving, it de-capsulates the

PPP frame. PPPoE supports PPP communication with PPPoE server and PAP/CHAP authentications.

- **ARP(Address Resolution Protocol)**

ARP is the MAC address resolution protocol by using IP address. This protocol exchanges ARP-reply and ARP-request from peers to determine the MAC address of each other

- **IP (Internet Protocol)**

This protocol operates in the IP layer and provides data communication. IP fragmentation is not supported. It is not possible to receive the fragmented packets. All protocol number is supported except for TCP or UDP. In case of TCP or UDP, use the hardwired embedded TCPIP stack.

- **ICMP(Internet Control Message Protocol)**

ICMP is a protocol which provides information, unreachable destination. When a Ping-request ICMP packet is received, a Ping-reply ICMP packet is sent.

- **IGMPv1/v2(Internet Group Management Protocol version 1/2)**

This protocol processes IGMP messages such as the IGMP Join/Leave. The IGMP is only enabled in UDP multicast mode. Only version 1 and 2 of IGMP logic is supported. When using a newer version of IGMP, IGMP should be manually implemented in the IP layer.

- **UDP(User Datagram Protocol)**

It is a protocol which supports data communication at the UDP layer. User datagram such as unicast, multicast, and broadcast are supported

- **TCP(Transmission Control Protocol)**

This protocol operates in the TCP layer and provides data communication. Both TCP server and client modes are supported.

1.4 Pin Description

1.4.1 Pin Layout

Package type: LQFP 100

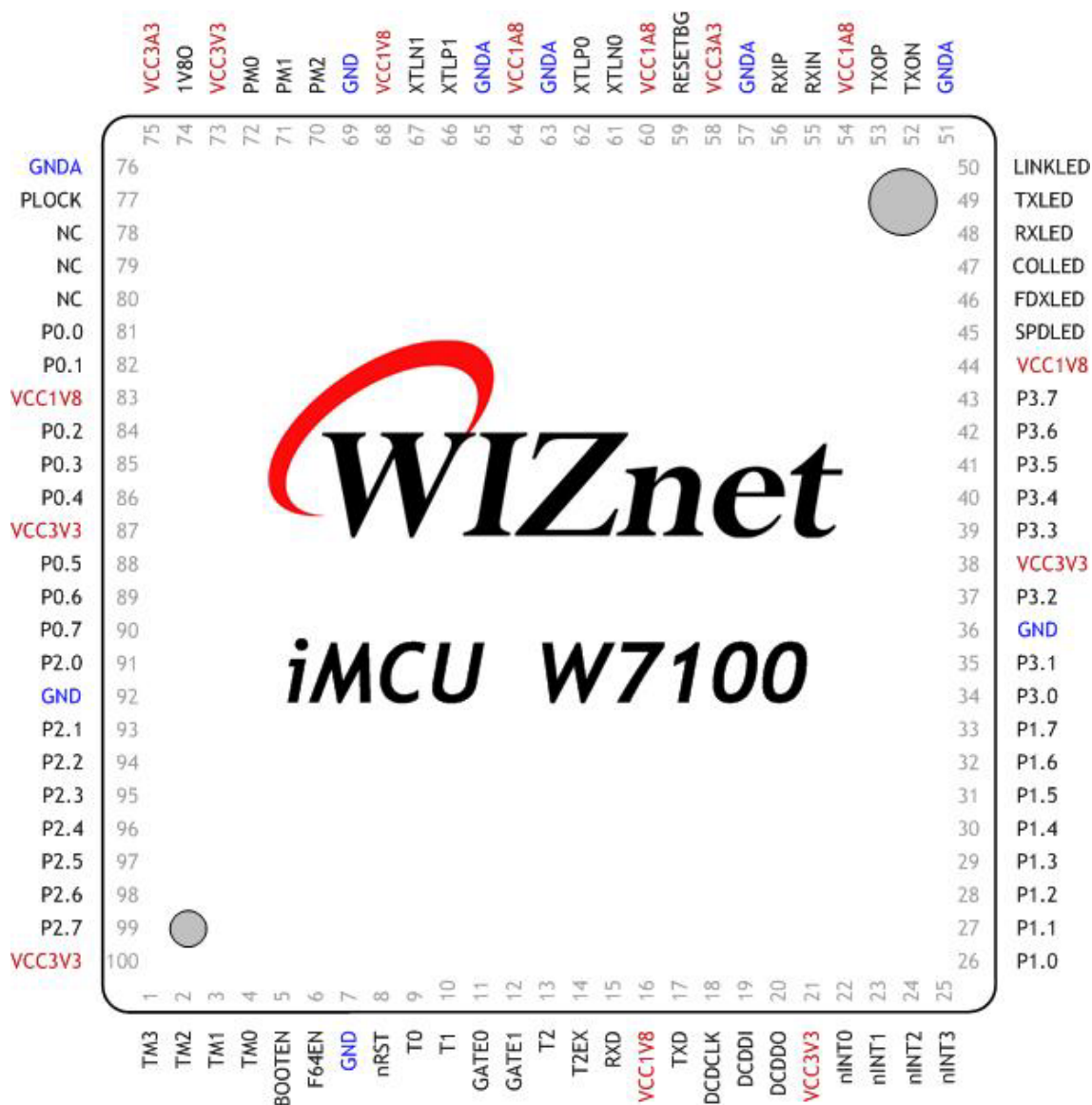


Figure 1.7 W7100 Pin Layout

1.4.2 Pin Description

The pin functionalities are described in the following table. All pins are unidirectional. There are no three-state output pins and internal signals.

Type	Description
I	Input
O	Output with 8mA driving current
IO	Input/Output (Bidirectional)
Pu	Internal pulled-up with 75KΩ resistor
Pd	Internal pulled-down with 75KΩ resistor

1.4.2.1 Configuration

Pin name	Num	I/O Type	Pu/Pd	Description																																							
nRST	8	I	-	Global asynchronous reset, Active low																																							
TM3-0	1,2, 3,4	I	Pd	W7100 Test Mode Normal mode : 0000 Other test modes are internal test mode																																							
PM2 - 0	70, 71, 72	I	Pd	PHY Mode <table><tr><th colspan="3">PM</th><th rowspan="2">Description</th></tr><tr><th>2</th><th>1</th><th>0</th></tr><tr><td>0</td><td>0</td><td>0</td><td>Normal Operation Mode Auto-negotiation enabled with all functionalities</td></tr><tr><td>0</td><td>0</td><td>1</td><td>Auto-negotiation with 100 BASE-TX FDX/HDX ability</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Auto-negotiation with 10 BASE-T FDX/HDX ability</td></tr><tr><td>0</td><td>1</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>0</td><td>Manual selection of 100 BASE-TX FDX</td></tr><tr><td>1</td><td>0</td><td>1</td><td>Manual selection of 100 BASE-TX HDX</td></tr><tr><td>1</td><td>1</td><td>0</td><td>Manual selection of 10 BASE-T FDX</td></tr><tr><td>1</td><td>1</td><td>1</td><td>Manual selection of 10 BASE-T HDX</td></tr></table>	PM			Description	2	1	0	0	0	0	Normal Operation Mode Auto-negotiation enabled with all functionalities	0	0	1	Auto-negotiation with 100 BASE-TX FDX/HDX ability	0	1	0	Auto-negotiation with 10 BASE-T FDX/HDX ability	0	1	1	Reserved	1	0	0	Manual selection of 100 BASE-TX FDX	1	0	1	Manual selection of 100 BASE-TX HDX	1	1	0	Manual selection of 10 BASE-T FDX	1	1	1	Manual selection of 10 BASE-T HDX
PM			Description																																								
2	1	0																																									
0	0	0	Normal Operation Mode Auto-negotiation enabled with all functionalities																																								
0	0	1	Auto-negotiation with 100 BASE-TX FDX/HDX ability																																								
0	1	0	Auto-negotiation with 10 BASE-T FDX/HDX ability																																								
0	1	1	Reserved																																								
1	0	0	Manual selection of 100 BASE-TX FDX																																								
1	0	1	Manual selection of 100 BASE-TX HDX																																								
1	1	0	Manual selection of 10 BASE-T FDX																																								
1	1	1	Manual selection of 10 BASE-T HDX																																								
FDX : Full-Duplex, HDX : Half-Duplex																																											
BOOTEN	5	I	Pd	Boot Enable/Disable 0 - Run User Application Jump to 0x0000, the start address of Code FLASH.																																							

1- Enable Boot

Run boot in Boot ROM

PLOCK	77	O	-	PLL Lock line, It notifies when internal PLL is locked
F64EN	6	I	-	Must be pulled down with 4.7KΩ resister

1.4.2.2 Timer

Pin name	Num	I/O Type	Pu/Pd	Description
Timer0, 1 Interface				
T0	9	I	-	Timer0 external clock input
T1	10	I	-	Timer1 external clock input
GATE0	11	I	-	Timer0 gate control
GATE1	12	I	-	Timer1 gate control
Timer 2 Interface				
T2	13	I	-	Timer2 external clock input
T2EX	14	I	-	Timer2 Capture/Reload trigger

1.4.2.3 UART

Pin name	Num	I/O Type	Pu/Pd	Description
RXD	15	IO	Pu	Serial receiver
TXD	17	O	-	Serial transmitter

1.4.2.4 DoCD™ Compatible Debugger

Pin name	Num	I/O Type	Pu/Pd	Description
DCDCLK	18	O	-	DoCD clock
DCDDI	19	I	-	DoCD data input
DCDDO	20	O	-	DoCD data output

1.4.2.5 Interrupt / Clock

Pin name	Num	I/O Type	Pu/Pd	Description
nINT0	22	I	-	External interrupt0
nINT1	23	I	-	External interrupt1
nINT2	24	I	-	External interrupt2
nINT3	25	I	-	External interrupt3
XTLN0	61	I	-	Crystal input/output for Ethernet PHY, A parallel-resonant 25MHz crystal is used to connect these
XTLP0	62	I	-	

				pins to stabilize the internal oscillator
XTLN1	67	I	-	Crystal input/output for W7100, A parallel-resonant
XTLP1	66	I	-	11.0592MHz crystal is used to connect these pins to stabilize the internal oscillator

1.4.2.6 GPIO

Pin name	Num	I/O Type	Pu/Pd	Description
P0.0	81	IO	Pu	Port0 input/output
P0.1	82	IO	Pu	Port0 input/output
P0.2	84	IO	Pu	Port0 input/output
P0.3	85	IO	Pu	Port0 input/output
P0.4	86	IO	Pu	Port0 input/output
P0.5	88	IO	Pu	Port0 input/output
P0.6	89	IO	Pu	Port0 input/output
P0.7	90	IO	Pu	Port0 input/output
P1.0	26	IO	Pu	Port1 input/output
P1.1	27	IO	Pu	Port1 input/output
P1.2	28	IO	Pu	Port1 input/output
P1.3	29	IO	Pu	Port1 input/output
P1.4	30	IO	Pu	Port1 input/output
P1.5	31	IO	Pu	Port1 input/output
P1.6	32	IO	Pu	Port1 input/output
P1.7	33	IO	Pu	Port1 input/output
P2.0	91	IO	Pu	Port2 input/output
P2.1	93	IO	Pu	Port2 input/output
P2.2	94	IO	Pu	Port2 input/output
P2.3	95	IO	Pu	Port2 input/output
P2.4	96	IO	Pu	Port2 input/output
P2.5	97	IO	Pu	Port2 input/output
P2.6	98	IO	Pu	Port2 input/output
P2.7	99	IO	Pu	Port2 input/output
P3.0	34	IO	Pu	Port3 input/output
P3.1	35	IO	Pu	Port3 input/output
P3.2	37	IO	Pu	Port3 input/output
P3.3	39	IO	Pu	Port3 input/output
P3.4	40	IO	Pu	Port3 input/output

P3.5	41	IO	Pu	Port3 input/output
P3.6	42	IO	Pu	Port3 input/output
P3.7	43	IO	Pu	Port3 input/output

Note: Since the output driving voltage of GPIO is 2.5V, if the user wants 3.3V, it needs additional pull-up resistor. Refer to the section “10. Electrical Specification.”

1.4.2.7 Media Interface

Pin name	Num	I/O Type	Pu/Pd	Description
TXON	52	O	-	TXON/TXOP Signal Pair, The differential data is transmitted to the media on the TXON/TXOP signal pair
TXOP	53	O	-	
RXIN	55	I	-	RXIN/RXIP Signal Pair, The differential data from the media is received on the RXIN/RXIP Signal pair
RXIP	56	I	-	
RESETBG	59	I	-	PHY Off-chip resistor, Connect a resistor of $12.3k\Omega \pm 1\%$ to the ground. Refer to the “Reference schematic”

For the best performance,

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

For more detail refer to “W5100 Layout Guide.pdf”.

1.4.2.8 Network Indicator LED

Pin name	Num	I/O Type	Pu/Pd	Description
SPDLED	45	O	-	Link speed LED According to auto-negotiation, it is asserted low at 100Mbps and high at 10Mbps or manual configuration of PM[2:0].
FDXLED	46	O	-	Full duplex LED This LED pin outputs low in the full-duplex mode and high in half-duplex mode depending on the auto-negotiation or manual configuration of PM[2:0].
COLLED	47	O	-	Collision LED, Active ‘Low’ This LED pin indicates when collisions occur. (only in

				half-duplex mode)
RXLED	48	O	-	Receive activity LED, Active 'Low' This LED pin indicates when the input is receiving data from RXIP/RXIN
TXLED	49	O	-	Transmit activity LED, Active 'Low' This LED pin indicates when the output is transmitting data through the TXOP/TXON
LINKLED	50	O	-	Link LED, Active 'Low' This LED pin indicates the link status of the media (10/100M)

1.4.2.9 Power Supply Signal

Pin name	Num	I/O Type	Pu/Pd	Description
VCC3A3	58, 75	Power	-	Analog 3.3V power supply Be sure to connect a 10uF tantalum capacitor between VCC3A3 and GNDA in order to prevent power compensation
VCC3V3	21, 38, 73, 87, 100	Power	-	Digital 3.3V power supply A 0.1uF decoupling capacitor should be connected between each pair of VCC and GND. A 1uH ferrite bead should be used to separate the VCC3V3 and VCC3A3
VCC1A8	54, 60, 64	Power	-	Analog 1.8V power supply A 10uF tantalum capacitor and a 0.1uF capacitor should be connected between VCC1A8 and GNDA to filter out core power noise
VCC1V8	16, 44, 68, 83	Power	-	Digital 1.8V power supply Between each pair of VCC and GND, a 0.1uF decoupling capacitor should be connected
GNDA	51, 57, 63, 65, 76	Power	-	Analog ground Design the analog ground plane as wide as possible during PCB layout
GND	6, 7, 36, 69, 92	Power	-	Digital ground Design the digital ground plane as wide as possible during PCB layout
1V80	74	Power	-	1.8V regulated output voltage

1.8V/150mA power generated by internal power regulator which is used for core operation power (VCC1A8, VCC1V8).

Between the 1V80 and GND, Be sure to connect a 3.3uF tantalum capacitor for output frequency compensation and a 0.1uF capacitor for high frequency noise decoupling.

1V80 is connected to VCC1V8, separated to 1uH ferrite bead and connected to VCC1A8.

<Notice> 1V80 is the power supply for W7100 use only. This supply should not be connected with any other devices.

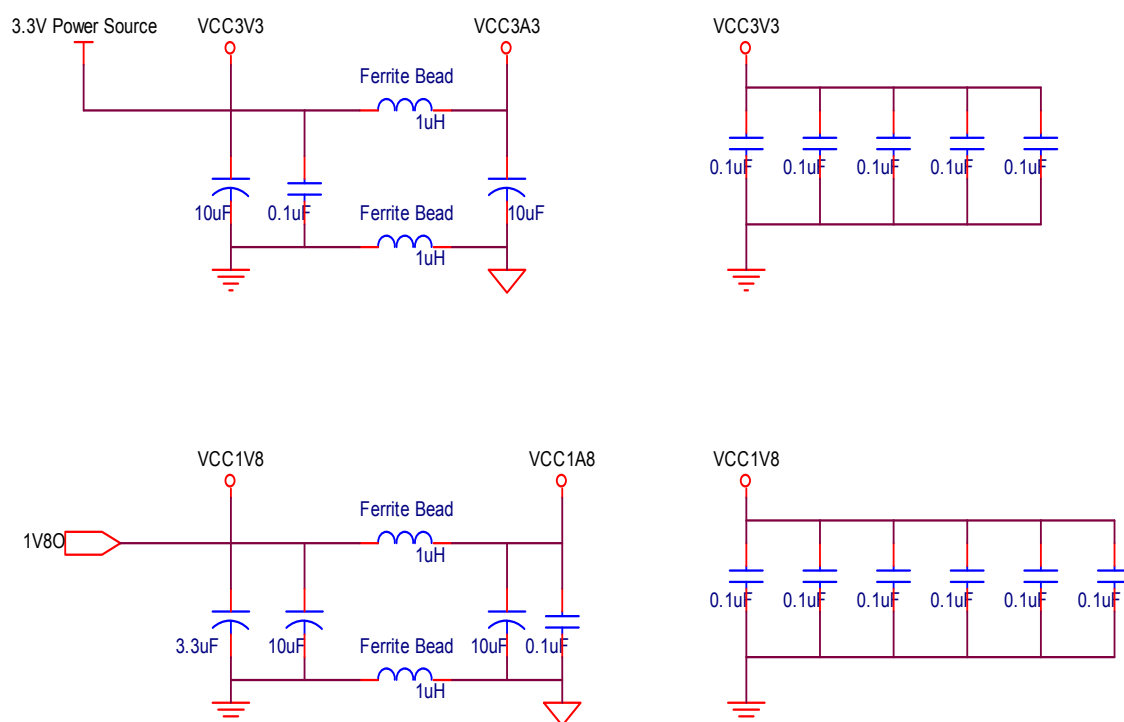


Figure 1.8 Power Design

2 Memory

The W7100's memory is divided into two types of memories: "Code Memory" and "Data Memory". The memory structure of W7100 is roughly shown figure 2.1.

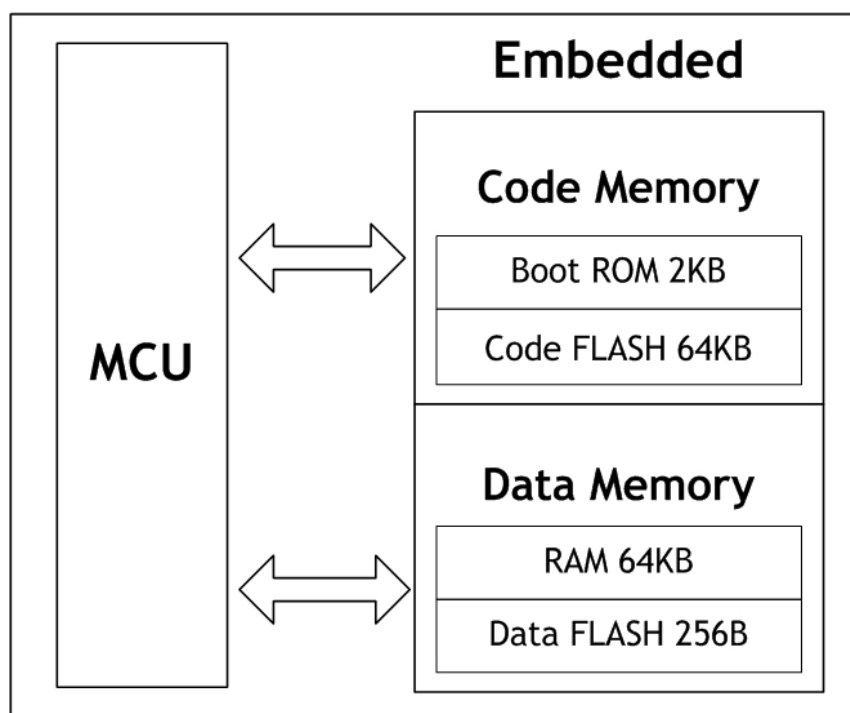


Figure 2.1 Code / Data Memory Connections

2.1 Code Memory

"Code Memory" consists of the Boot ROM from 0x0000 to 0x07FF and Code FLASH from 0x0000 to 0xFFFF. After the system is reset, the W7100 always executes the code of Boot ROM at "Code Memory". According to the BOOTEN pin, the code of Boot ROM executes differently. The Figure 2.2 shows the flow of Boot ROM code. After booting, the system proceeds to either the ISP process or the APP Entry according to the BOOTEN pin. When ISP process is selected (BOOTEN = '1'), the ISP code of the Boot ROM is running. Otherwise(BOOTEN = '0'), the system jumps to the APP Entry without running the ISP code of Boot ROM.

The APP Entry contains the 'memory map switching code' and the jumping code which jumps to the start address 0x0000 of the user application in Code FLASH memory. The memory map switching is as below.

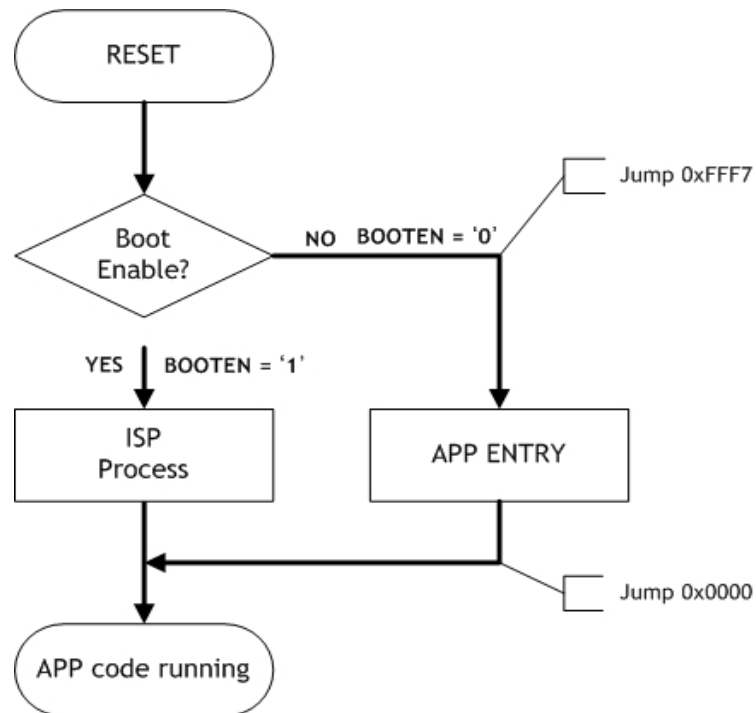


Figure 2.2. Boot Sequence Flowchart

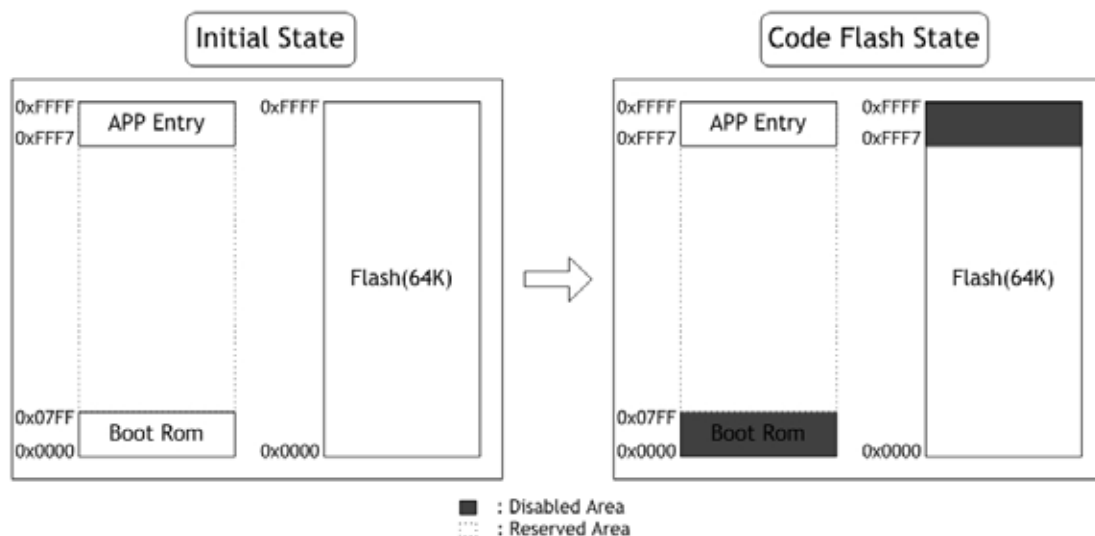


Figure 2.3 "Code Memory" Map Switching

The initial state of W7100 has both 'Boot ROM / APP Entry' and FLASH as in the Figure 2.3. But, since the 'Boot ROM / APP Entry' and FLASH are overlapped, they use same address at 0x0000 ~ 0x07FF / 0xFFFF7 ~ 0xFFFF. So the iMCU W7100 maps the 'Boot ROM / APP Entry' and the FLASH(64K) to the code and data memory respectively.

The FLASH mapped to data memory can be written the user application code. But in this state, the FLASH cannot be used as a code memory because this state is for writing user application code. To use the FLASH as a code memory, it needs switching the memory map.

For this reason, user should select APP Mode by setting the BOOTEN pin to '0', and then the Boot ROM code jumps to APP Entry immediately. Next, the APP Entry un-maps the Boot ROM and maps the Code FLASH to code memory as the Figure 2.3. After switching the code memory map, the APP Entry jumps to start address of Code FLASH (0x0000). This flow is shown in the Figure 2.4.

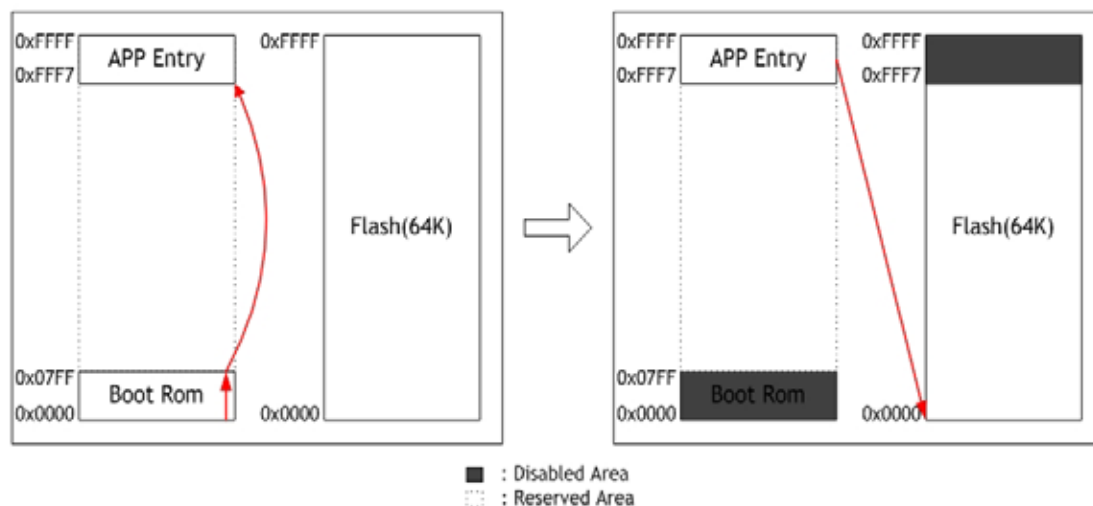


Figure 2.4 APP Entry Process

Otherwise, if the APP Mode is selected, the Code FLASH 64KB can be used as a code memory as the Figure 2.4. But both FLASH and APP Entry are still overlapped at same address. Therefore, to use FLASH 64KB fully, the APP Entry must be un-map on "Code Memory". To un-map APP Entry, User should be set RB bit in WCONF(0xFF) to '0' at user startup code. Then the APP Entry is un-mapped as below.

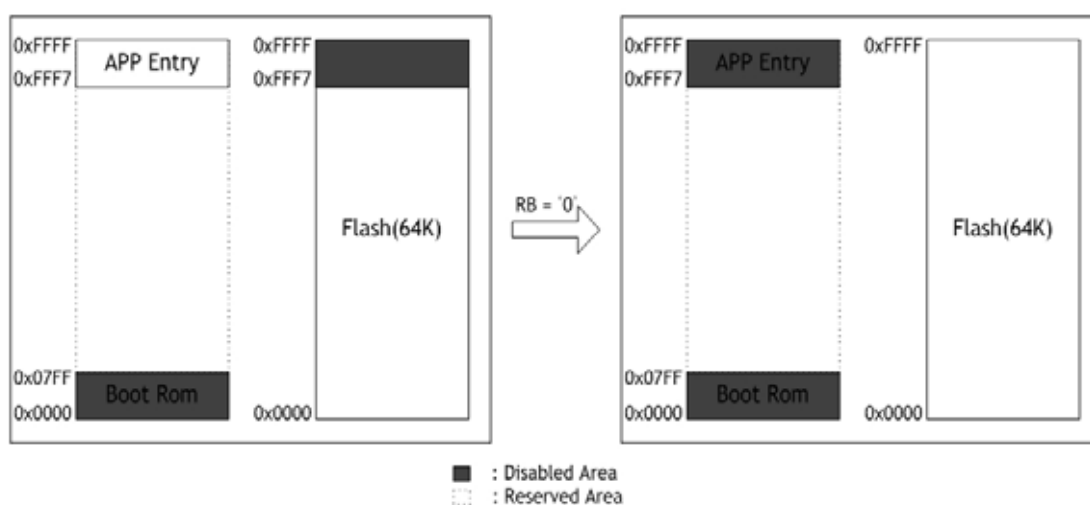


Figure 2.5 Changing the code memory Status at RB = '0'

WCONF (0xFF)								
7	6	5	4	3	2	1	0	Reset
RB	ISPEN	-	-	-	F64EN	FB	BE	0x00

When the Code FLASH takes more than 0xFFFF7, the below code must be inserted into startup code. If using this method, the W7100 immediately disables the APP Entry address after its system reset.

```
ANL    0FFH,    #07FH    ; Clear Reboot flag
```

Set the BOOTEN pin to '0' and clear the **RB** bit of WCONF register at the startup code. Then the embedded Code FLASH 64KB memory of the W7100 can be completely used as a code memory.

2.1.1 Code Memory Wait States

The wait states are managed by internal WTST(0x92) register. The number of wait states is fixed by the value stored in the WTST register. Please refer to the section 2.4.2 for more details.

2.2 Data Memory

The W7100 contains 64KB of embedded RAM, 64KB of TCPIPCore and the 256Byte of the Data FLASH. The Data FLASH is for user's information such as IP address, MAC address, subnet mask and port number and so on. These memories are accessed by MOVX instructions only. The Figure 2.6 shows the "Data Memory" map of W7100.

2.2.1 Data Memory Wait States

The number of wait states is managed by the CKCON (0x8E) register. Please refer to the section 2.4.5 for more details. For the timing information, please refer to the section 10.

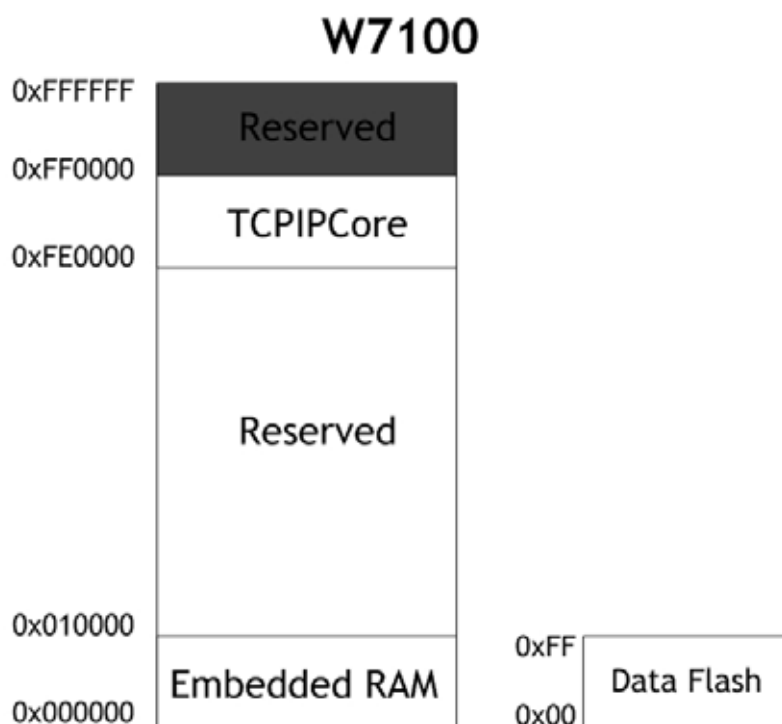


Figure 2.6 “Data Memory” Map

2.3 Internal Data Memory and SFR

The Figure below shows the Internal Memory and Special Function Registers (SFR) map.

0xFF	Upper Internal RAM shared with Stack space <i>(indirect addressing)</i>	SFR Special Function Registers <i>(direct addressing)</i>
0x80	Lower Internal RAM shared with Stack space <i>(direct & indirect addressing)</i>	
0x7F		
0x30	bit addressable area	
0x2F		
0x20	4 banks, R0-R7 each	
0x1F		
0x00		

Figure 2.11 Internal Memory Map

The lower internal RAM consists of four register banks with eight registers each, a bit-addressable segment with 128 bits (16 bytes) that begins at 0x20, and a scratchpad area with 208 bytes is embedded. With **indirect** addressing mode ranging from 0x80 to 0xFF, the highest 128 bytes is accessed as an internal memory. But with **direct** addressing mode ranging from 0x80 to 0xFF, this area is accessed as a SFR memory.

0xF8	EIP	DPSBK	TMPR0	TMPR1	TMPR2	TMPR3	PHYCONF	WCONF	0xFF
0xF0	B	ISPID	ISPADDR		ISPDATA	CKCBK	DPX0BK	DPX1BK	0xF7
0xE8	EIE		MXAX						0xEF
0xE0	ACC								0xE7
0xD8	WDCON								0xDF
0xD0	PSW								0xD7
0xC8	T2CON		RLDL	RLDH	TL2	TH2			0xCF
0xC0						Reserved		TA	0xC7
0xB8	IP								0xBF
0xB0	P3								0xB7
0xA8	IE								0xAF
0xA0	P2								0xA7
0x98	SCON	SBUF							0x9F
0x90	P1	EIF	WTST	DPX0		DPX1			0x97
0x88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON		0x8F
0x80	P0	SP	DPL0	DPH0	DPL1	DPH1	DPS	PCON	0x87

Figure 2.12 SFR Memory Map

New SFR - New additional SFR, described in this section

Extended SFR - Extended from standard 8051, described in this section

Standard - standard 8051 SFR, described in this section

All of the SFR in the left hand side column ending with 0 or 8 are bit addressable.

2.4 SFR definition

The following section describes SFR of **W7100** and their functionalities. For detail information about standard and peripheral SFR, please refer to the peripheral section.

2.4.1 Program Code Memory Write Enable Bit

Inside the PCON register, the Program Write Enable (PWE) bit is used to enable/disable Program Write signal activity during MOVX instructions.

When the PWE bit is set to logic '1', the "MOVX @DPTR, A" instruction writes the data from the accumulator register into the code memory addressed by using the DPTR register (active DPH:DPL)

The "MOVX @Rx, A" instruction writes the data from the accumulator register into code memory addressed by using the P2 register (bits 15:8) and Rx register (bits 7:0).

PCON (0x87)

7	6	5	4	3	2	1	0	Reset
SMOD0	-	-	PWE	-	0	0	0	0x00

Figure 2.13 PWE bit of PCON Register

Note: 1. PCON.2 ~ PCON.0 bits are reserved. They cannot be set to 1.

2.4.2 Program Code Memory Wait States Register

Wait states register provides the information for code memory access time.

WTST (0x92)								
7	6	5	4	3	2	1	0	Reset
-	-	-	-	-	WTST.2	WTST.1	WTST.0	0x07

Figure 2.14 Code memory Wait States Register

Note: 1. These bits are considered during program fetches and MOVC instructions only.

Since code memory write are performed by MOVX instruction, CKCON register regulates the CODE-WR pulse width.

2. Read cycle takes minimal 4 clock period and maximal 8 clock periods.

Table 2.1 WTST Register Values

WTST[2:0]	Access Time [clk]
7	8
6	7
5	6
4	5
3	4
2	Not Used
1	Not Used
0	Not Used

During Instruction fetching, code memory can be accessed by MOVC instruction only. The code memory can be read with minimal 3 wait states. The timing diagrams are shown in the Figures below.

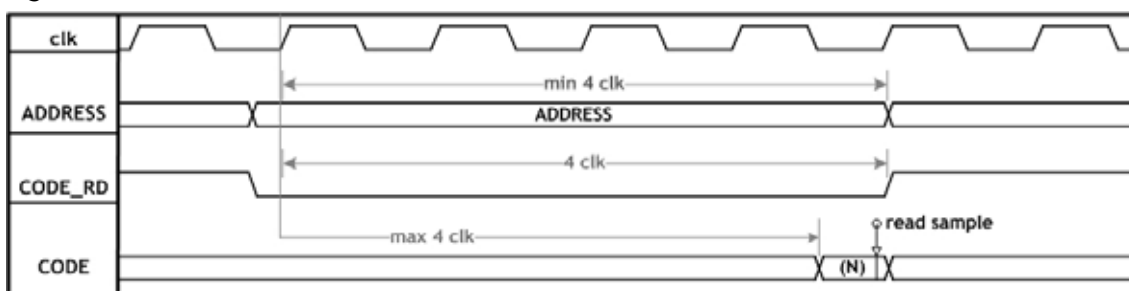


Figure 2.15 Waveform for code memory Synchronous Read Cycle with Minimal Wait States (WTST = '3')

- Note:**
1. clk - System clock frequency (88.4736 MHz)
 2. ADDRESS - Address of the actual modified program byte
 3. CODE_RD - Read signal of the code memory
 4. CODE - Data write to the actual modified program byte

The code memory can be written by MOVX instruction with minimal 3 wait states. It allows W7100 core to operate with fast and slow code memory devices. The timing diagrams are shown in the Figure below.

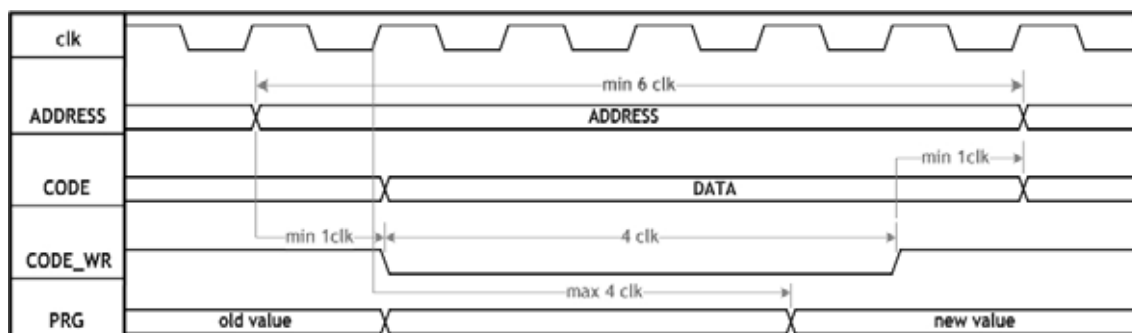


Figure 2.16 Waveform for code memory Synchronous Write Cycle with Minimal Wait States(WTST = '3')

- Note:**
1. clk - System clock frequency (88.4736 MHz)
 2. ADDRESS - Address of the actual modified program byte
 3. CODE - Data write to the actual modified program byte
 4. CODE_WR - Write signal of the code memory
 5. PRG - State of the code memory

2.4.3 Data Pointer Extended Registers

Data Pointer Extended registers, DPX0, DPX1 and MXAX, hold the most significant part of memory addresses when accessing to data located above 64KB. After reset, DPX0, DPX1, and MXAX restores to the default value 0x00.

DPX0 (0x93)

7	6	5	4	3	2	1	0	Reset
DPXP.7	DPX.6	DPX.5	DPX.4	DPX.3	DPX.2	DPX.1	DPX.0	0x00

Figure 2.17 Data Pointer Extended Register

DPX1 (0x95)

7	6	5	4	3	2	1	0	Reset
DPX1.7	DPX1.6	DPX1.5	DPX1.4	DPX1.3	DPX1.2	DPX1.1	DPX1.0	0x00

Figure 2.18 Data Pointer Extended Register

MXAX (0xEA)

7	6	5	4	3	2	1	0	Reset
MXAM.7	MXAX.6	MXAX.5	MXAX.4	MXAX.3	MXAX.2	MXAX.1	MXAX.0	0x00

Figure 2.19 MOVX @RI Extended Register

When MOVX instruction uses DPTR0/DPTR1 register, the most significant part of the address A[23:16] is always equal to the content of DPX0(0x93)/DPX1(0x95). When MOVX instruction uses R0 or R1 register, the most significant part of the address A[23:16] is always equal to the

2.4.4 Data Pointer Registers

DPTR0(0x83:0x82)

Figure 2.20 Data Pointer Register DPTR0

Figure 2.20 Data Pointer Register DPTR0

DPTR1(0x85:0x84)

Figure 2.21 Data Pointer 1 Register DPTR1

Figure 2.21 Data Pointer 1 Register DPTR1

DPS (0x86)

Figure 2.22 Data Pointer Select Register

Figure 2.22 Data Pointer Select Register

INC DPTR

MOV DPTR, #data16

MOVC A, @A + DPTR

MOVX @DPTR, A

MOVX A, @DPTR

When TSL = 0, DPTR related instructions will not affect the state of the SEL bit.

Unimplemented bit - Read as 0 or 1.

Table 2.2 DPTR0, DPTR1 Operations

© Copyright 2009 WIZnet Co., Inc. All rights reserved.

Selected data pointer register is used in the following instructions:

MOVX @DPTR, A

MOVX A, @DPTR

MOVC A, @A + DPTR

JMP @A + DPTR

INC DPTR

MOV DPTR, #data16

2.4.5 Clock Control Register

Clock control register CKCON (0x8E) contains MD [2:0] bits which provide the information for the dedicated data memory read/write signal pulses width.

CKCON (0x8E)								Reset
7	6	5	4	3	2	1	0	
WD1	WD0	-	-	-	MD2	MD1	MD0	0x07

Figure 2.23 Clock Control Register - STRETCH bits

The dedicated data memory read/write signals are activated during MOVX instruction. The purpose of MD[2:0] is to adjust the communication speed with I/O devices such as slow RAM, LCD displays, etc. After reset, MD[2:0] will be restored to the default value of 0x07, which means that slow devices work properly. Users can change the MD[2:0] value to speed up/slow down the software execution. The value of MD[2:0] can be changed any time during program execution (e.g. between MOVX and different speed devices).

Table 2.3 MD[2:0] Bit Values

MD[2:0]	Pulse Width[clock]
7	8
...	...
2	3
1	Not Used
0	Not Used

This read/write pulse width must have a minimum of 3 clock cycle and a maximum of 8 clock cycle. For timing diagram, please refer to the section '10. Electrical specification.'

2.4.6 Stack Pointer

The W7100 has an 8-bit stack pointer called SP(0x81) and is located in the internal RAM space.

SP (0x81)								Reset
7	6	5	4	3	2	1	0	
SP.7	SP.6	SP.5	SP.4	SP.3	SP.2	SP.1	SP.0	0x07

Figure 2.24 Stack Pointer Register

This pointer is incremented before data is stored in PUSH and CALL executions, and decremented after data is popped in POP, RET, and RETI executions. In other words, the Stack pointers always points to the last valid stack byte.

2.4.7 New & Extended SFR

ISPID(0xF1) : ID Register for ISP.

ISPADDR16(0xF2) : 16bit Address Register for ISP

ISPDATA(0xF4) : Data Register for ISP.

CKCBK(0xF5) : CKCON Backup Register.

DPX0BK(0xF6) : DPX0 Backup Register.

DPX1BK(0xF7) : DPX1 Backup Register.

DPSBK(0xF9) : DPX Backup Register.

RAMBA16(0xFA) : RAM Base Address Register.

RAMEA16(0xFC) : RAM End Address Register.

WCONF (0xFF)								
7	6	5	4	3	2	1	0	Reset
RB	ISPEN	-	-	-	F64EN	FB	BE	0x00

Figure 2.26 W7100 Configuration Register

Note: RB : 0 - No Reboot / 1 - Reboot after the ISP done (APP Entry(0xFFFF7 ~ 0xFFFF)
RD/WR Enable)
ISPEN : 0 - Enable ISP in Boot built in W7100 / 1 - Disable
- : Reserved, must be set to '000'
F64EN : Always '0'. Read only.
FB : FLASH Busy Flag for ISP. Read only.
BE : Boot Enable (1 - Boot Running / 0 - Apps Running). Read only.

2.4.8 Peripheral Registers

P0, P1, P2, P3 : Port register. For detail information, please refer to the section 4 for the Functionality of I/O Ports.

TCON(0x88) : Timer0, 1 configuration register. For detail information, please refer to the section 5.1 for the Functionality of Timer0 and Timer 1.

TMOD(0x89) : Timer0, 1 control mode register. For detail information, please refer to the section 5.1 for the Functionality of Timer0 and Timer 1.

TH0(0x8C), TL0(0x8A) : Counter register of timer 0. For detail information, please refer to the section 5.1 for the Functionality of Timer0 and Timer 1.

TH1(0x8D), TL1(0x8B) : Counter register of timer 1. For detail information, please refer to the section 5.1 for the Functionality of Timer0 and Timer 1.

SCON(0x98) : UART Configuration Register. For detail information, please refer to the

section 6 for the Functionality of UART.

SBUF(0x99) : UART Buffer Register. For detail information, please refer to the section 6 for the Functionality of UART.

IE(0xA8) : UART Bits in Interrupt Enable Register. For detail information, please refer to the section 6 for the Functionality of UART.

IP(0xB8) : UART Bits in Interrupt Priority Register. For detail information, please refer to the section 6 for the Functionality of UART.

TA(0xC7) : Timed Access Register. For detail information, please refer to the section 7 for Timed Access Registers of Watchdog Timer.

T2CON(0xC8) : Timer 2 Configuration Register. For detail information, please refer to the section 5.2 for the Functionality of Timer 2.

RLDH(0xCB), RLDL(0xCA) : Capture Registers of Timer 2. For detail information, please refer to the section 5.2 for the Functionality of Timer 2.

TH2(0xCD), TL2(0xCC) : Counter Register of Timer 2. For detail information, please refer to the section 5.2 for the Functionality of Timer 2.

PSW(0xD0) : Program Status Word Register. For detail information, please refer to the section 1.3.1.

WDCON(0xD8) : Watchdog Control Register. For detail information, please refer to the section 7.

3 Interrupt

The interrupt pin functionalities are described in the table below. All pins are unidirectional. There are no three-state output pins and internal signals.

Table 3.1 External Interrupt Pin Description

Pin	Active	Type	Pu/Pd	Description
nINT0/FA6	Low/Falling	I	-	External interrupt 0
nINT1/FA7	Low/Falling	I	-	External interrupt 1
nINT2/FA8	Falling	I	-	External interrupt 2
nINT3/FA9	Falling	I	-	External interrupt 3
nINT4			-	Reserved
TCPIPCore (nINT5)	Falling	I	-	Interrupt Request Signal for TCPIPCore (internally connected)

The W7100 core is implemented with two levels of interrupt priority control. Each external interrupt can be in high or low level priority group by setting or clearing a bit in the IP(0xB8) and EIP(0xF8) registers. External interrupt pins are activated by a falling edge signal. Interrupt requests are sampled at the rising edge of the system's clock.

Table 3.2 W7100 Interrupt Summary

Interrupt Flag	Function	Active Level/Edge	Flag Reset	Vector	Interrupt Number	Natural Priority
IE0	Device pin INT0	Low/Falling	Hardware	0x03	0	1
TF0	Internal, Timer0	-	Hardware	0x0B	1	2
IE1	Device pin INT1	Low/Falling	Hardware	0x13	2	3
TF1	Internal, Timer1	-	Hardware	0x1B	3	4
TI & RI	Internal, UART	-	Software	0x23	4	5
TF2	Internal, Timer2	-	Software	0x2B	5	6
INT2F	Device Pin INT2	Falling	Software	0x43	8	7
INT3F	Device Pin INT3	Falling	Software	0x4B	9	8
INT4F	Reserved					
INT5F TCPIPCore	Interrupt for TCPIPCore	Falling	Software	0x5B	11	10
WDIF	Internal, WATCHDOG	-	Software	0x63	12	11

Each interrupt vector can be individually enabled or disabled by changing the corresponding bit in IE(0xA8) and EIE(0xE8) registers. The IE register contains global interrupt system disable(0)/enable(1) bit called EA.

IE (0xA8)								
7	6	5	4	3	2	1	0	Reset
EA	-	ET2	ES	ET1	EX1	ET0	EX0	0x00

Figure 3.1 Interrupt Enable Register

Note: EA - Enable global interrupt

EX0 - Enable INT0 interrupt

ET0 - Enable Timer0 interrupt

EX1 - Enable INT1 interrupt

ET1 - Enable Timer1 interrupt

ES - Enable UART interrupt

ET2 - Enable Timer2 interrupt

All these bits which generate interrupts can be set or cleared by software, with the same result by hardware. That is, interrupts can be generated or cancelled by software. The only exceptions are the request flags IE0 and IE1. If the external interrupt 0 or 1 are programmed as level-activated, the IE0 and IE1 are controlled by the external source pins nINT0/FA6 and nINT1/FA7 respectively.

IP (0xB8)								
7	6	5	4	3	2	1	0	Reset
-	-	PT2	PS	PT1	PX1	PT0	PX0	0x00

Figure 3.2 Interrupt Priority Register

Note: PX0 - INT0 priority level control (high level at 1)

PT0 - Timer0 priority level control (high level at 1)

PX1 - INT1 priority level control (high level at 1)

PT1 - Timer1 priority level control (high level at 1)

PS - UART priority level control (high level at 1)

PT2 - Timer2 priority level control (high level at 1)

Unimplemented bit - Read as 0 or 1

TCON (0x88)								
7	6	5	4	3	2	1	0	Reset
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0x00

Figure 3.3 Timer0, 1 Configuration Register

Note: IT0 - INT0 level (at 0)/edge (at 1) sensitivity

IT1 - INT1 level (at 0)/edge (at 1) sensitivity

IE0 - INT0 interrupt flag Cleared by hardware when processor branches to

interrupt routine

IE1 - INT1 interrupt flag Cleared by hardware when processor branches to interrupt routine

TF0 - Timer0 interrupt (overflow) flag. Cleared by hardware when processor branches to interrupt routine

TF1 - Timer 1 interrupt (overflow) flag. Cleared by hardware when processor branches to interrupt routine

SCON (0x98)

7	6	5	4	3	2	1	0	Reset
SM0	SM1	SM2	REN	TB8	RB8	TI	RI	0x00

Figure 3.4 UART Configuration Register

Note: RI - UART receiver interrupt flag

TI - UART transmitter interrupt flag

EIE (0xE8)

7	6	5	4	3	2	1	0	Reset
-	-	-	EWDI	EINT5	EINT4	EINT3	EINT2	0x00

Figure 3.5 Extended Interrupt Enable Register

Note: EINT2 - Enable INT2 Interrupt

EINT3 - Enable INT3 Interrupt

EINT4 - Must be '0', if use the EIE register

EINT5 - Enable TCPIPCore Interrupt

EWDI - Enable WATCHDOG Interrupt

EIP (0xF8)

7	6	5	4	3	2	1	0	Reset
-	-	-	PWDI	PINT5	PINT4	PINT3	PINT2	0x00

Figure 3.6 Extended Interrupt Priority Register

Note: PINT2 - INT2 priority level control (high level at 1)

PINT3 - INT3 priority level control (high level at 1)

PINT4 - Must be set to '0', if use the EIP register

PINT5 - TCPIPCore Interrupt priority level control (high level at 1)

PWDI - WATCHDOG priority level control (high level at 1)

EIF (0x91)								
7	6	5	4	3	2	1	0	Reset
-	-	-	-	INT5F	INT4F	INT3F	INT2F	0x00

Figure 3.7 Extended Interrupt Flag Register

Note: INT2F - INT2 interrupt flag. Must be cleared by software
INT3F - INT3 interrupt flag. Must be cleared by software
INT4F - Must be set to '0'. if use the EIF register
INT5F - TCIPCore Interrupt flag. Must be cleared by software

WDCON (0xD8)								
7	6	5	4	3	2	1	0	Reset
-	-	-	-	WDIF	WTRF	EWT	RWT	0x00

Figure 3.8 Watchdog Control Register

Note: WDIF - Watchdog Interrupt Flag. WDIF in conjunction with the Enable Watchdog Interrupt bit (EIE.4) and EWT provides information such as a Watchdog Timer event has been encountered or not and what action should be taken. This bit must be cleared by software before exiting the interrupt service routine, or else another interrupt is generated. By using software to enable the WDIF, a Watchdog interrupt is generated. Enabled software-set WDIF will generate a Watchdog interrupt. Timed Access Register procedure can be used to modify this bit.

4 I/O Ports

The I/O port pin functionalities are described in the following table.

Table 4.1 I/O Ports Pin Description

Pin	Active	Type	Pu/Pd	Description
P0[7:0]	-	IO	Pu	Port0 input / output Recommends additional pull-up resistor
P1[7:0]	-	IO	Pu	Port1 input / output Recommends additional pull-up resistor
P2[7:0]	-	IO	Pu	Port2 input / output Recommends additional pull-up resistor
P3[7:0]	-	IO	Pu	Port3 input / output Recommends additional pull-up resistor

P0 (0x80)

7	6	5	4	3	2	1	0	Reset
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	0xFF

Figure 4.1 Port0 Register

P1 (0x90)

7	6	5	4	3	2	1	0	Reset
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	0xFF

Figure 4.2 Port1 Register

P2 (0xA0)

7	6	5	4	3	2	1	0	Reset
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	0xFF

Figure 4.3 Port2 Register

P3 (0xB0)

7	6	5	4	3	2	1	0	Reset
P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	0xFF

Figure 4.4 Port3 Register

Read and write accesses are performed in the I/O ports via their corresponding SFR: P0 (0x80), P1 (0x90), P2 (0xA0), and P3 (0xB0). Some port-reading instructions read from the data registers while others read from the port pin. The “Read-Modify-Write” instructions are directed to the data registers as shown below.

Table 4.2 Read-Modify-Write Instructions

Instruction	Function Description
ANL	Logic AND
ORL	Logic OR
XRL	Logic exclusive OR
JBC	Jump if bit is set and cleared
CPL	Complement bit
INC, DEC	Increment, decrement byte
DJNZ	Decrement and jump if not zero
MOV Px.y, C	Move carry bit to bit y of port x
CLR Px.y	Clear bit y of port x
SETB Px.y	Set bit y of port x

All other instructions read from a port exclusively through the port pins. All ports pin can be used as GPIO (General Purpose Input Output). In the GPIO mode, the reserved bits of the WCONF register must be set to '000'. The GPIO of W7100 is shown in the Figure below. Since the output driving voltage of GPIO is 2.5V, if the user wants 3.3V, it needs additional pull-up register.

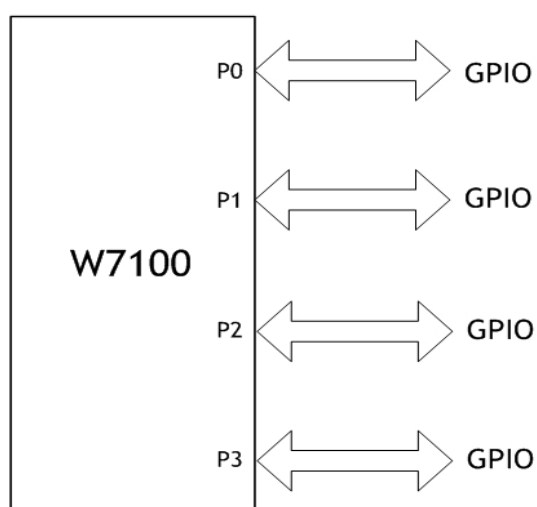


Figure 4.5 Usage of the GPIO pins in the W7100

5 Timers

The W7100 contains two 16-bit timers/counters, Timer0 and Timer 1. In the ‘timer mode’, the timer registers are incremented by every 12 CLK periods. In “counter mode”, the timer registers are incremented during the falling transition on their corresponding input pins: T0 or T1. The input pins are sampled at every CLK period.

5.1 Timers 0, 1

5.1.1 Overview

The Timer0, 1 pin functionalities are described in the following table. All pins are unidirectional. There are no three-state output pins and internal signals.

Table 5.1 Timers 0, 1 Pin Description

Pin	Active	Type	Pu/Pd	Description
T0/FCS	Falling	I	-	Timer0 clock
GATE0/FOE	High	I	-	Timer0 clock gate control
T1/FAE	Falling	I	-	Timer1 clock
GATE1/FA0	High	I	-	Timer1 clock gate control

Timer0 and Timer 1 are fully compatible with the standard 8051 timers. Each timer consists of two 8-bit registers, TH0 (0x8C) and TL0 (0x8A), TH1 (0x8D) and TL1 (0x8B). The timers work in four modes which are described below.

Table 5.2 Timers 0, 1 Mode

M1	M0	Mode	Function Description
0	0	0	THx operates as a 8-bit timer/counter with a divided-by-32 prescaler served by lower 5-bit of TLx
0	1	1	16-bit timer/counter. THx and TLx are cascaded.
1	0	2	TLx operates as a 8-bit timer/counter with 8bit auto-reload by THx.
1	1	3	TL0 is configured as a 8-bit timer/counter controlled by the standard Timer0 bits. TH0 is a 8-bit timer controlled by Timer 1 control bits. Timer 1 holds its count.

TMOD (0x89)

Timer1				Timer0				
7	6	5	4	3	2	1	0	Reset
GATE	CT	M1	M0	GATE	CT	M1	M0	0x00

Figure 5.1 Timer0, 1 Control Mode Register

Note: GATE - Gating control

1: Timer x is enabled while GATEx pin is at high and TRx control bit is set

0: Timer x is enabled while TRx control bit is set

CT - Counter or timer select bit

1: Counter mode, Timer x clock source from Tx pin

0: Timer mode, internally clocked

M1, M0 - Mode select bits

TCON (0x88)

7	6	5	4	3	2	1	0	Reset
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0x00

Figure 5.2 Timer0, 1 Configuration Register

Note: TR0 - Timer0 run control bit

1: Enabled

0: Disabled

TR1 - Timer 1 run control bit

1: Enabled

0: Disabled

External input pins, GATE0 and GATE1, can be programmed to function as a gate to facilitate pulse width measurements.

5.1.2 Interrupts

Timer0, 1 interrupt related bits are shown below. An interrupt can be toggled by the IE register, and priorities can be configured in the IP register.

IE (0xA8)

7	6	5	4	3	2	1	0	Reset
EA	-	ET2	ES	ET1	EX1	ET0	EX0	0x00

Figure 5.3 Interrupt Enable Register

Note: EA - Enable global interrupts

ET0 - Enable Timer0 interrupts

ET1- Enable Timer1 interrupts

IP (0xB8)

7	6	5	4	3	2	1	0	Reset
-	-	PT2	PS	PT1	PX1	PT0	PX0	0x00

Figure 5.4 Interrupt Priority Register

Note: PT0 - Enable global interrupts

PT1 - Enable Timer0 interrupts

Unimplemented bit - Read as 0 or 1

TCON (0x88)

7	6	5	4	3	2	1	0	Reset
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0x00

Figure 5.5 Timer0, 1 Configuration Register

Note: TF0 - Timer0 interrupt (overflow) flag. Cleared by hardware when processor branches to interrupt routine

TF1 - Timer1 interrupt (overflow) flag. Cleared by hardware when processor branches to interrupt routine

All of the bits which generate interrupts can be set or cleared by software, with the same result by hardware. That is, interrupts can be generated or cancelled by software.

Table 5.3 Timer0, 1 interrupts

Interrupt Flag	Function	Active Level/Edge	Flag Resets	Vector	Natural Priority
TF0	Internal, Timer0	-	Hardware	0x0B	2
TF1	Internal, Timer1	-	Hardware	0x1B	4

5.1.3 Timer0 - Mode0

The Timer0 register is configured as a 13-bit register (8bit: Timer, 5bit: prescaler). As the all counts (valid bits) roll over from 1 to 0, Timer0 interrupt flag TF0 is set. The timer starts counting when TCON.4 =1 and either TMOD.3 = 0 or GATE0 = 1. By setting TMOD.3 = 1, the external input GATE0 can control Timer0 to manage the pulse width measurements. The 13-bit register consists of 8 bits TH0 and 5 bits of TL0. The upper 3 bits of TL0 should be ignored. Refer to the following Figure for details.

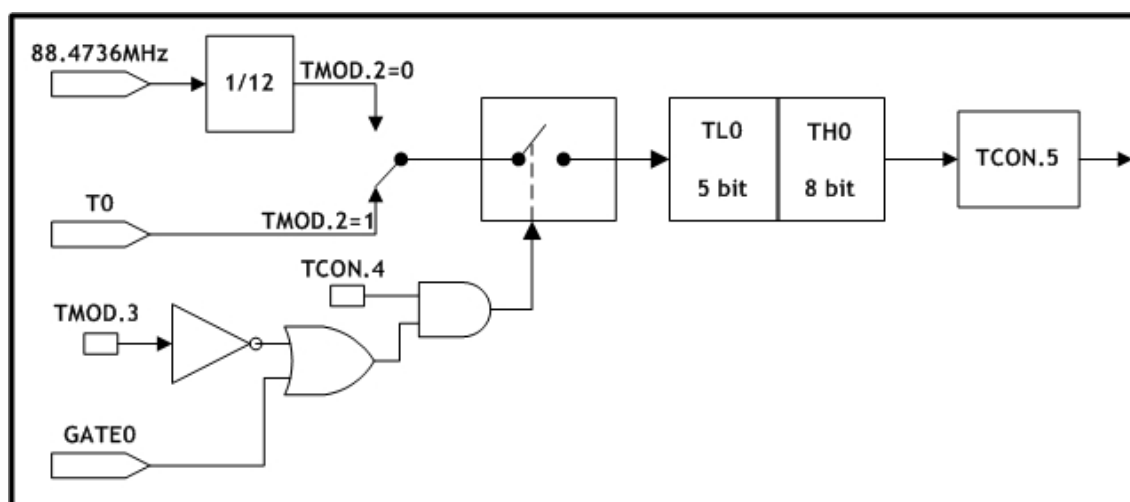


Figure 5.6 Timer Counter0, Mode0: 13-Bit Timer/Counter

5.1.4 Timer0 - Mode1

Mode1 is the same as Mode0, except that the timer register is running with all 16 bits. Mode1 is shown in the Figure below.

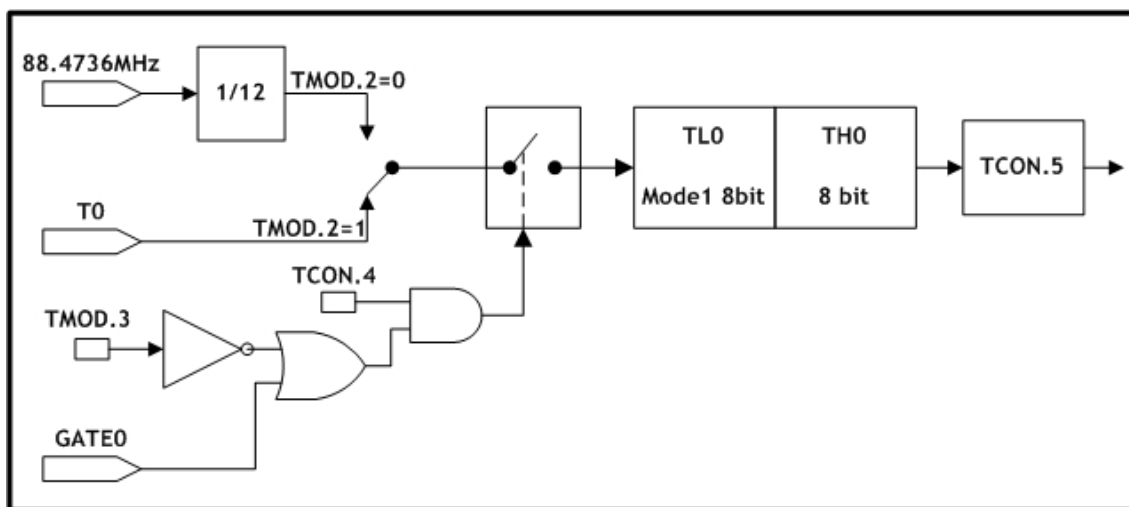


Figure 5.7 Timer/Counter0, Mode1: 16-Bit Timer/Counter

5.1.5 Timer0 - Mode2

Mode2 configures the timer register as a 8-bit counter TL0 with automatic reload as shown in the Figure below. During an overflow from TL0, it sets TF0 and reloads the contents of TH0 into TL0. TH0 remains unchanged after the reload is completed.

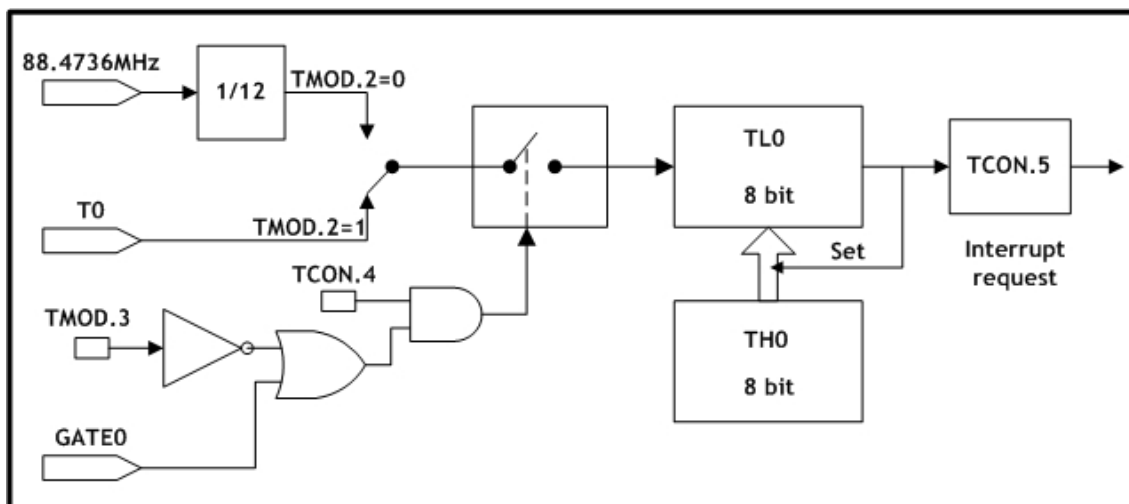


Figure 5.8 Timer/Counter0, Mode2: 8-Bit Timer/Counter with Auto-Reload

5.1.6 Timer0 - Mode3

In this mode, the TL0 and TH0 are divided into two separate counters. The following Figure shows the logic for Timer0 running in Mode3. The TL0 uses the Timer0 control bits: C/T, GATE, TR0, GATE0 and TF0. And the TH0 is locked into a timer function and uses TR1 and TF1 flags from Timer 1 and controls Timer 1 interrupt. Mode3 is used in applications which require an

extra 8-bit timer/counter. When Timer0 is in Mode3, Timer 1 can be turned on/off by switching itself into Mode3, or can still be used by the serial channel as a baud rate generator, or in any application where interrupt from Timer 1 is not required.

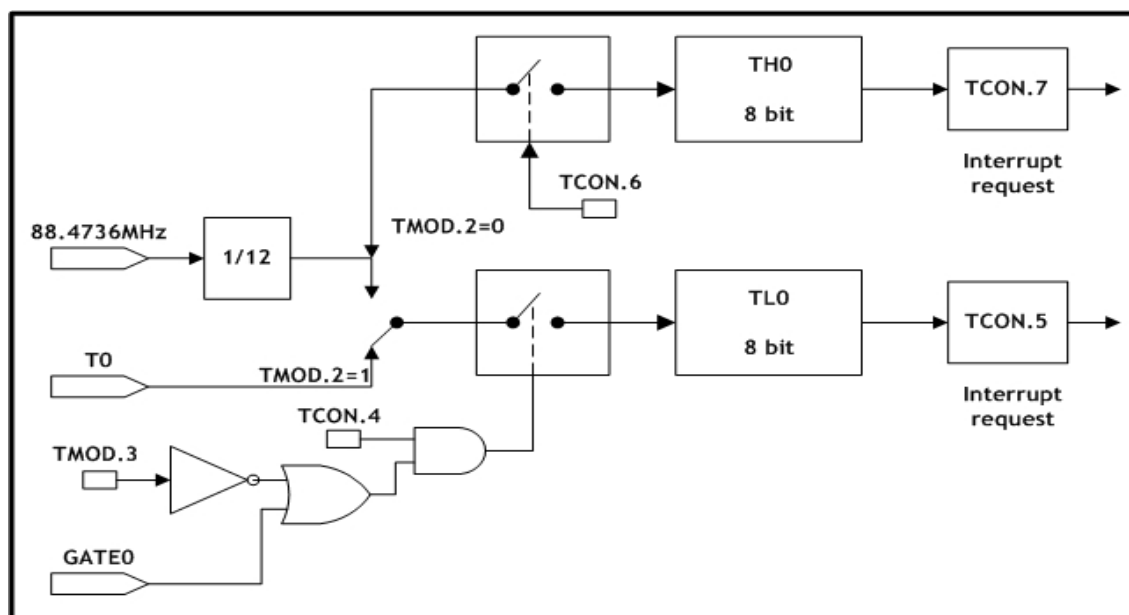


Figure 5.9 Timer/Counter0, Mode3: Two 8-Bit Timers/Counters

5.1.7 Timer1 - Mode0

In this mode, the Timer1 register is configured as a 13-bit register (8bit: Timer, 5bit: prescaler). As the all counts (valid bits) roll over from 1 to 0, Timer 1 interrupt flag TF1 is set. The counted input is enabled to Timer 1 when TCON.6 = 1 and either TMOD.6 = 0 or GATE1 = 1. (Setting TMOD.7 = 1 allows Timer 1 controlled by external input GATE1, to facilitate pulse width measurements). The 13-bit register consists of 8 bits TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Refer to the following Figure for detail.

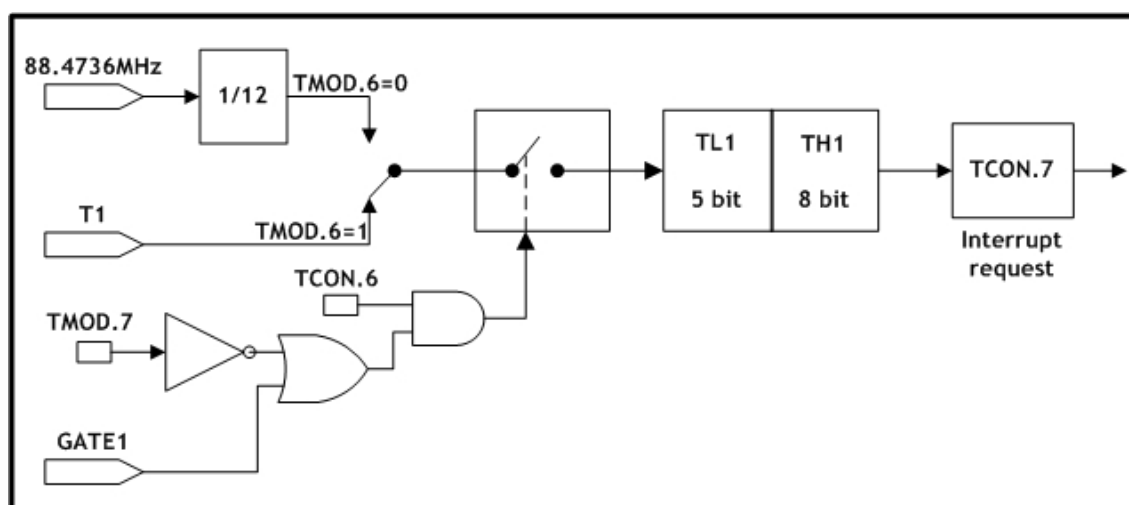


Figure 5.10 Timer/Counter1, Mode0: 13-Bit Timer/Counter

5.1.8 Timer1 - Mode1

Mode1 is the same as Mode0, except that the timer register is running with all 16 bits. Mode1 is shown in the Figure below.

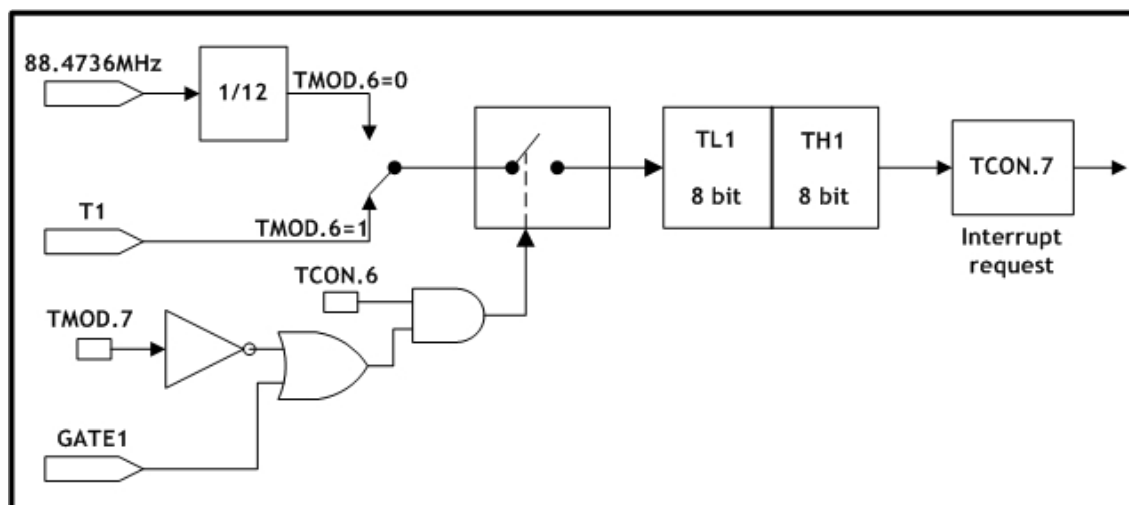


Figure 5.11 Timer/Counter1, Mode1: 16-Bit Timers/Counters

5.1.9 Timer1 - Mode2

Mode2 configures timer register as 8-bit counter TL1, with automatic reload as shown in Figure below. Overflow from TL1 only sets TF1, but also automatically reloads TL1 with the contents of TH1. The reload leaves TH1 unchanged.

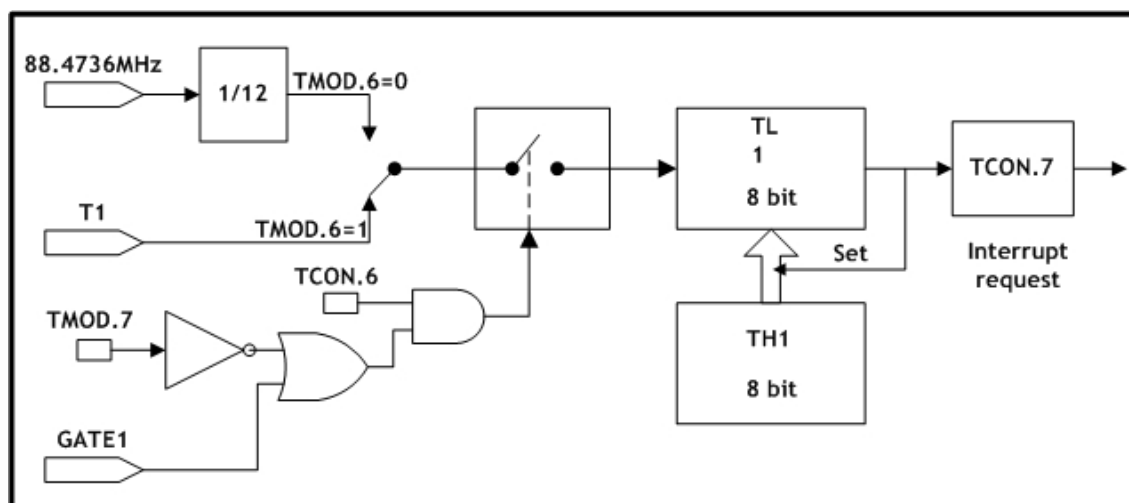


Figure 5.12 Timer/Counter1, Mode2: 8-Bit Timer/Counter with Auto-Reload

5.1.10 Timer1 - Mode3

Timer1 in Mode3 holds counting. The effect is the same as setting TR1 = 0 because it is used for Timer0-Mode3. For more detail, please refer to the section "5.1.6 Timer0-Mode3".

5.2 Timer2

5.2.1 Overview

The Timer2 pin functionalities are described in the following table. All pins are unidirectional. There are no three-state output pins and internal signals.

Table 5.4 Timer2 Pin Description

Pin	Active	Type	Pu/Pd	Description
T2/FA1	Falling	I	-	Timer2 external clock input
T2EX/FA2	Falling	I	-	Timer2 capture/reload trigger

Timer2 of W7100 is fully compatible with the standard 8051 Timer2. A total of five SFR are used to control Timer2 operation, TH2/TL2 (0xCD/0xCC) counter registers, RLDH/RLDL (0xCB/0xCA) capture registers, and T2CON (0xC8) control register. Timer2 works under three modes selected by T2CON bits as shown in the table below.

Table 5.5 Timer2 Modes

RCLK,TCLK	CPRL2	TR2	Function Description
0	0	1	16-bit auto-reload mode. TF2 bit is set when Timer2 overflows. TH2 and TL2 registers are reloaded with 16-bit value from RLDH and RLDL.
0	1	1	16-bit capture mode. TF2 bit is set when Timer2 overflows. When EXEN2=1 and T2EX pin is on the falling edge, TH2 and TL2 register values are stored into RLDH and RLDL.
1	X	1	Baud rate generator for UART interface
X	X	0	Timer2 is off.

T2CON (0xC8)

7	6	5	4	3	2	1	0	Reset
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	CT2	CPRL2	0x00

Figure 5.13 Timer2 Configuration Register

Note: EXF2 - indicates a Falling edge in the T2EX pin when EXEN2=1. Must be cleared by software

RCLK - Receive clock enable

0: UART receiver is clocked by Timer1 overflow pulses

1: UART receiver is clocked by Timer2 overflow pulses

TCLK - Transmit clock enable

0: UART transmitter is clocked by Timer1 overflow pulses

1: UART transmitter is clocked by Timer2 overflow pulses

EXEN2 - Enable T2EX pin functionality

0: Ignore T2EX events

1: Allow capture or reload as a result of T2EX pin falling edge

TR2 - Start/Stop Timer2

0: Stop

1: Start

CT2 - Timer/Counter select

0: Internally clocked timer

1: External event counter. Clock source is T2 pin

CPRL2 - Capture/Reload select

0: Automatic reload occurs when Timer2 overflow or falling edge of the T2EX pin with EXEN2=1. When RCLK or TCLK is set, this bit is ignored and automatic reload when Timer2 overflows.

1: On the falling edge of T2EX pin, capture is activated when EXEN2=1.

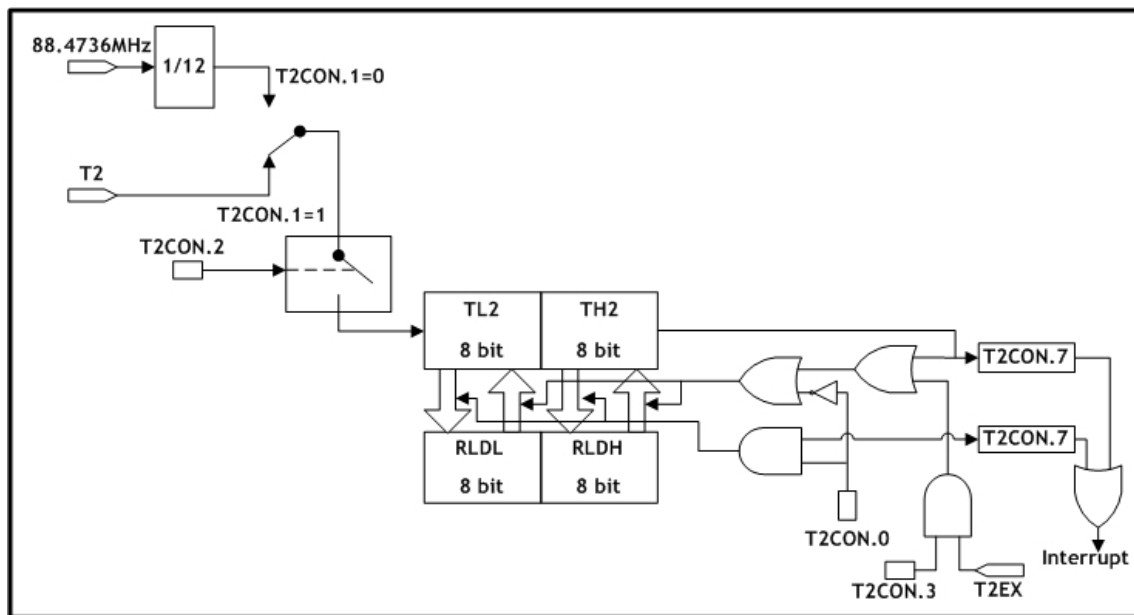


Figure 5.14 Timer/Counter2, 16-Bit Timer/Counter with Auto-Reload

5.2.2 Interrupts

The interrupt bits for Timer2 are shown below. An interrupt can be toggled by the IE register, and priorities can be configured by the IP register.

IE (0xA8)								
7	6	5	4	3	2	1	0	Reset
EA	-	ET2	ES	ET1	EX1	ET0	EX0	0x00

Figure 5.15 Interrupt Enable Register – Timer2

Note: EA - Enable global interrupts

ET2 - Enable Timer2 interrupts

IP (0xB8)								Reset
7	6	5	4	3	2	1	0	
-	-	PT2	PS	PT1	PX1	PT0	PX0	0x00

Figure 5.16 Interrupt Priority Register – Timer2

Note: PT2 - Timer2 interrupt priority level control (high level at 1)

Unimplemented bit - Read as 0 or 1

T2CON (0xC8)								Reset
7	6	5	4	3	2	1	0	
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	CT2	CPRL2	0x00

Figure 5.17 Timer2 Configuration Register – TF2

Note: TF2 - Timer2 interrupt (overflow) flag. It must be cleared by software. This flag will not be set when either RCLK or TCLK is set.

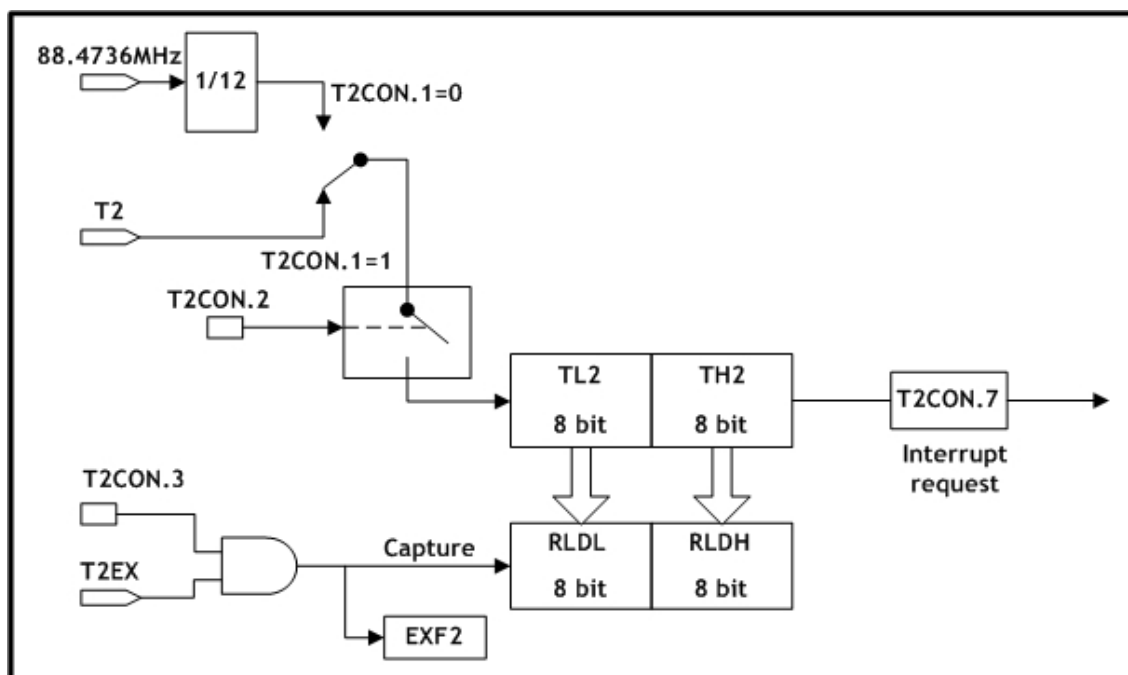


Figure 5.18 Timer/Counter2, 16-Bit Timer/Counter with Capture Mode

All of the bits that generate interrupts can be set or cleared by software, with the same result by hardware. That is, interrupts can be generated or cancelled by software.

Table 5.6 Timer2 Interrupt

Interrupt Flag	Function	Active Level/Edge	Flag Resets	Vector	Natural Priority
TF2	Internal, Timer2	-	Software	0x2B	6

Using the 0x2B vector, EXF2 is set by this interrupt, but the TF2 flag remains unchanged.

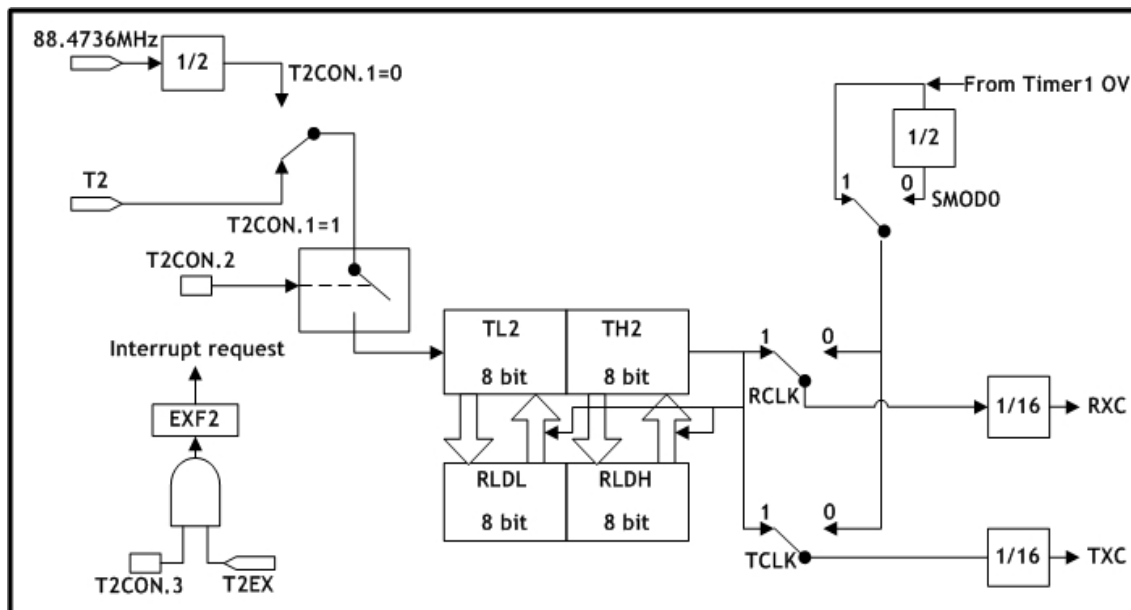


Figure 5.19 Timer2 for Baud Rate Generator Mode

6 UART

The UART of W7100 operates in full duplex mode which is capable of receiving and transmitting at the same time. Since the W7100 is double-buffered, the receiver is capable of receiving data while the first byte of the buffer is not read. During a read operation, the SBUF reads from the receive register. On the other hand, SBUF loads the data into the transmit register during a send operation. The UART has 4 different modes which include one in synchronous mode and three in asynchronous modes. Modes 2 and 3 include a special feature for multiprocessor communication. This feature is enabled by setting the SM2 bit in the SCON register. The master processor sends out the first address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte, while an address byte will interrupt all slaves. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that were not being addressed leave their SM2 set and ignore the incoming data.

The pin functionalities of UART are described in the following table.

Table 6.1 UART Pin Description

Pin	Active	Type	Pu/Pd	Description
RXD	-	IO	Pu	Serial receiver input / output
TXD	-	O	-	Serial transmitter

The UART of W7100 is fully compatible with the standard 8051 UART. The UART related registers are: SBUF (0x99), SCON (0x98), PCON (0x87), IE (0xA8) and IP (0xB8). The UART data buffer (SBUF) consists of two registers: transmit and receive. When data is written into the SBUF transmit register, the sending process begins. Similarly, data is read from the receive register during the receiving process.

SBUF (0x99)

7	6	5	4	3	2	1	0	Reset
SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0	0x00

Figure 6.1 UART Buffer Register

SCON (0x98)

7	6	5	4	3	2	1	0	Reset
SM0	SM1	SM2	REN	TB8	RB8	TI	RI	0x00

Figure 6.2 UART Configuration Register

Note: SM2 - Enable a multiprocessor communication feature
SM1 - Set baud rate

SM0 - Set baud rate

REN - '1' : enable serial receive

'0': disable serial receive

TB8 - The 9th transmitted data bit in Modes 2 and 3. This bit is enabled depending on the MCU's operation (parity check, multiprocessor communication, etc.),

RB8 - In Modes 2 and 3, it is the 9th bit of data received. In Mode1, if SM2 is 0, RB08 is a stop bit. In Mode0, this bit is not used.

The UART modes are presented in the table below.

Table 6.2 UART Modes

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift register	$f_{osc}/12$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$f_{osc}/32$ or $/64$
1	1	3	9-bit UART	Variable

The UART baud rates are presented below.

Table 6.3 UART Baud Rates

Mode	Baud Rate
Mode0	$f_{osc}/12$
Mode1,3	Time1 overflow rate or Timer2 overflow rate
Mode2	SMOD0 = 0 $f_{osc}/64$ SMOD0 = 1 $f_{osc}/32$

The SMOD0 bit is located in the PCON register.

PCON (0x87)

7	6	5	4	3	2	1	0	Reset
SMOD0	SMOD1	-	PWE	-	0	0	0	0x00

Figure 6.3 UART Bits in Power Configuration Register

Note: SMOD0 - Bit for UART baud rate

Unimplemented bit - Read as 0 or 1

Bits 2-0 must be written as 0

6.1 Interrupts

UART interrupt related bits are shown below. An interrupt can be toggled by the IE register,

and priorities can be configured by the IP register.

IE (0xA8)								
7	6	5	4	3	2	1	0	Reset
EA	-	ET2	ES	ET1	EX1	ET0	EX0	0x00

Figure 6.4 UART Bits in Interrupt Enable Register

Note: ES - RI & TI interrupt enable flag

IP (0xB8)								
7	6	5	4	3	2	1	0	Reset
-	-	PT2	PS	PT1	PX1	PT0	PX0	0x00

Figure 6.5 UART Bits in Interrupt Priority Register

Note: SMOD0 - Bit for UART baud rate

Unimplemented bit - Read as 0 or 1

SCON (0x98)								
7	6	5	4	3	2	1	0	Reset
SM0	SM1	SM2	REN	TB08	RB08	TI	RI	0x00

Figure 6.6 UART Configuration Register

Note: TI - Transmit interrupt flag, set by hardware after completion of a serial transfer. It must be cleared by software.

RI - Receive interrupt flag, set by hardware after completion of a serial reception. It must be cleared by software.

All of the bits that generate interrupts can be set or cleared by software, with the same result by hardware. That is, interrupts can be generated or cancelled by software.

Table 6.4 UART Interrupt

Interrupt Flag	Function	Active Level/Edge	Flag Resets	Vector	Natural Priority
TI & RI	Internal, UART	-	software	0x23	5

6.2 Mode0, Synchronous

TXD output is a shift clock. The baud rate is fixed at 1/12 of the CLK clock frequency. Eight bits are transmitted with LSB first. Reception is initialized by setting the flags in SCON as follows: RI = 0 and REN = 1.

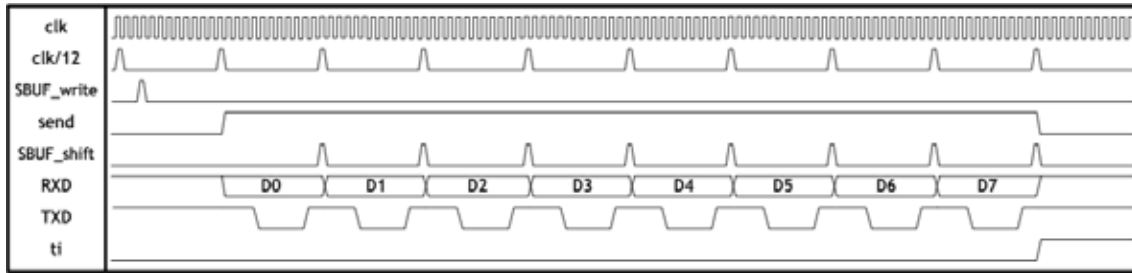


Figure 6.7 Timing Diagram for UART Transmission Mode0 (clk = 88.4736 MHz)

6.3 Mode1, 8-Bit UART, Variable Baud Rate, Timer 1 or 2 Clock Source

The pin RXD serves as an input while TXD serves as an output for the serial communication. 10 bits are transmitted in the following sequence: a start bit (always 0), 8 data bits (LSB first), and a stop bit (always 1). During data reception, a start bit synchronizes the transmission. Next, the 8 data bits can be accessed by reading SBUF, and the stop bit triggers the flag RB08 in SFR SCON (0x98). The baud rate is variable and dependent on Timer 1 or Timer 2 mode. To enable Timer 2 clocking, set the TCLK and RCLK bits which are located in the T2CON (0xC8) register.



Figure 6.8 Timing Diagram for UART Transmission Mode1

6.4 Mode2, 9-Bit UART, Fixed Baud Rate

This mode is almost identical to Mode1 except that the baud rate is fixed at 1/32 or 1/64 of CLK clock frequency, and 11 bits are transmitted or received in the following sequence: A start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). The 9th bit can be used to control the parity of the UART interface. During a transmission, the TB08 bit in SCON is outputted as the 9th bit. While receiving data, the 9th bit changes the RB08 bit in SCON.



Figure 6.9 Timing Diagram for UART Transmission Mode2

6.5 Mode3, 9-Bit UART, Variable Baud Rate, Timer1 or 2 Clock Source

The only difference between Mode2 and Mode3 is the baud rate in Mode3 is variable. Data reception is enabled when REN = 1. The baud rate is variable and dependent on Timer 1 or Timer 2 mode. To enable Timer 2 clocking, set the TCLK and RCLK bits which are located in the T2CON (0xC8) register.



Figure 6.10 Timing Diagram for UART Transmission Mode3

6.6 Examples of Baud Rate Setting

Table 6.5 Examples of Baud Rate Setting

Baud Rate(bps)	Timer 1 / Mode2		Timer 2
	TH1(0x8D)		RLDH(0xCB), RLDL(0xCA)
	SMOD = ‘0’	SMOD = ‘1’	
2400	160(0xA0)	64(0x40)	64384(0XFB80)
4800	208(0xD0)	160(0xA0)	64960(0xFDC0)
9600	232(0xE8)	208(0xD0)	65248(0xFEE0)
14400	240(0xF0)	224(0xE0)	65344(0XFF40)
19200	244(0xF4)	232(0xE8)	65392(0XFF70)
28800	248(0xF8)	240(0xF0)	65440(0XFFA0)
38400	250(0xFA)	244(0xF4)	65464(0XFFB8)
57600	252(0xFC)	248(0xF8)	65488(0XFFD0)
115200	254(0xFE)	252(0xFC)	65512(0XFFE8)
230400	255(0xFF)	254(0xFE)	65524(0XFFF4)

Note: Baud Rate calculation formula

Using Timer1 - Baud Rate = $(2^{SMOD} / 32) * (\text{Clock Frequency} / 12(256 - TH1))$

Using Timer2 - Baud Rate = $\text{Clock Frequency} / (32 * (65536 - (RLDH, RLDL)))$

7 Watchdog Timer

7.1 Overview

The Watchdog Timer is driven by the main system clock that is supplied by a series of dividers as shown in the Figure below. The divider output is selectable and determines the timeout intervals. When the timeout is reached, an interrupt flag will be set, and if enabled, a reset will be occurred. When interrupt enable bit and global interrupt are enabled, the interrupt flag will activate the interrupts. The reset and interrupt are completely discrete functions that may be acknowledged separately, together or even ignored depending on the application.

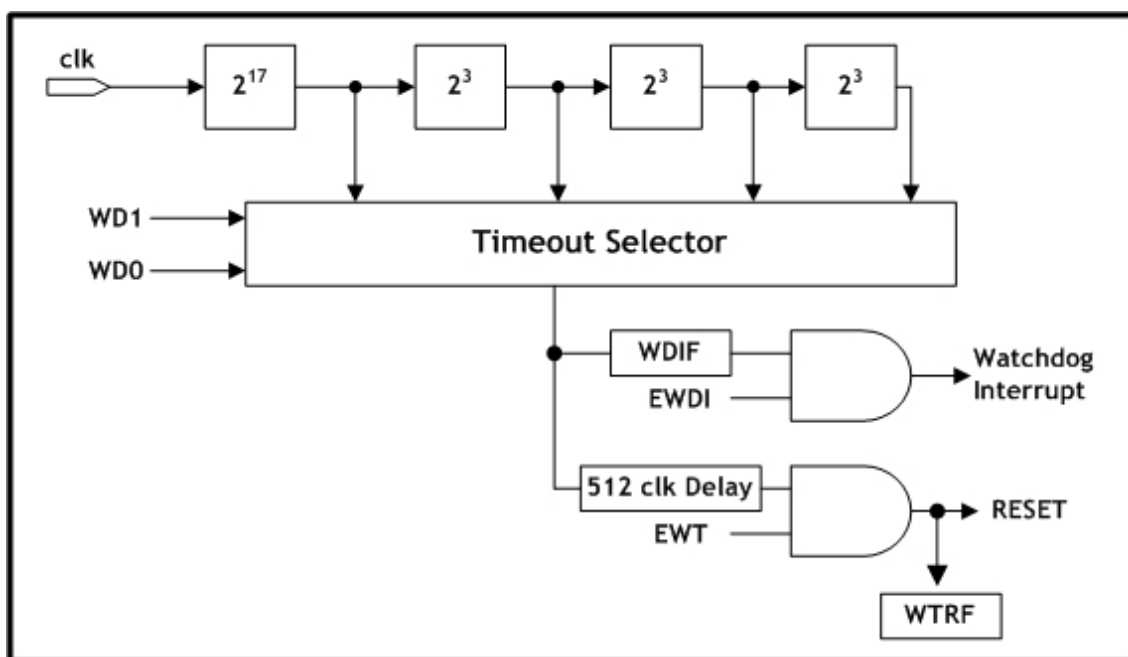


Figure 7.1 Watchdog Timer Structure

7.2 Interrupts

Watchdog interrupt related bits are shown below. An interrupt can be turned on/off by the IE (0xA8) and EIE (0xE8) registers, and high/low priorities can be set in the EIP EIP (0xF8) register. The IE contains global interrupt system disable (0) / enable (1) bit called EA.

IE (0xA8)								
7	6	5	4	3	2	1	0	Reset
EA	-	ET2	ES	ET1	EX1	ET0	EX0	0x00

Figure 7.2 Interrupt Enable Register

EIE (0xE8)								
7	6	5	4	3	2	1	0	Reset
-	-	-	EWDI	EINT5	EINT4	EINT3	EINT2	0x00

Figure 7.3 Extended Interrupt Enable Register

Note: EA - Enable global interrupt

EWDI - Enable Watchdog interrupt

EIP (0xF8)								
7	6	5	4	3	2	1	0	Reset
-	-	-	PWDI	PINT5	PINT4	PINT3	PINT2	0x00

Figure 7.4 Extended interrupt Priority Register

Note: PWDI - Watchdog priority level control (high level at 1)

Unimplemented bit - Read as 0 or 1

WDCON (0xD8)								
7	6	5	4	3	2	1	0	Reset
-	-	-	-	WDIF	WTRF	EWT	RWT	0x00

Figure 7.5 Watchdog Control Register

Note: WDIF - Watchdog Interrupt Flag. WDIF in conjunction with the EWDI (Enable WatchDog Interrupt) bit (EIE.4) and EWT indicates if a Watchdog Timer event has occurred and the action to be taken. This bit must be cleared by software before exiting the interrupt service routine, or another interrupt is generated. Setting WDIF in software will generate a Watchdog interrupt if enabled. This bit can be modified by using the **Timed Access Registers**.

All of the bits that generate interrupts can be set or cleared by software, with the same result by hardware. That is, interrupts can be generated or cancelled by software.

Table 7.1 Watchdog Interrupt

Interrupt Flag	Function	Active Level/Edge	Flag Reset	Vector	Natural Priority
WDIF	Internal, Watchdog	-	Software	0x63	11

7.3 Watchdog Timer Reset

The Watchdog Timer reset operates as follows. Once the timeout interval is initialized, the system restarts the Watchdog first by using RWT. Then, the reset mode is enabled by the EWT (Enable Watchdog Timer reset = WDCON.1) bit. Before the timer reaches the user selected terminal value, the software can set the RWT (Reset Watchdog Timer = WDCON.0) bit. If RWT is set before the timeout is reached, the timer will start over. If the timeout is reached without RWT being set, the Watchdog will reset the MCU. The Hardware automatically clears RWT after sets the RWT by software. When a reset occurs, the WTRF (Watchdog Timer reset Flag = WDCON.2) will automatically set to indicate the cause of the reset; however, software must clear this bit manually.

7.4 Simple Timer

The Watchdog Timer is a free running timer. In timer mode with reset disabled (EWT = 0) and interrupt functions disabled (EWDI = 0), the timer counts up to pre-programmed interval in WD[1:0] which will enable the Watchdog interrupt flag. By resetting the RWT bit, this timer can operate in polled timeout mode. The WDIF bit can be cleared by software or reset. The Watchdog interrupt is available for application which requires a long timer. The interrupt is enabled by using the EWDI (Enable WatchDog timer Interrupt = EIE.4) bit. When a timeout occurs, the Watchdog Timer will set the WDIF bit (WDCON.3), and an interrupt will occur if the global interrupt enable (EA) is set. **Note that WDIF is set to 512 clocks before a potential Watchdog reset.** The Watchdog interrupt flag indicates the source of the interrupt, and must be cleared by software. When the Watchdog interrupt is used properly, the Watchdog reset allows the interrupt software to monitor the system for any errors.

7.5 System Monitor

When using the Watchdog Timer as a system monitor, the Watchdog reset function should be used. If the Interrupt function was used, the purpose of the Watchdog would be defeated. For example, assume the system is executing errant code prior to the Watchdog interrupt. The interrupt would temporarily force the system back into control by vectoring the MCU to the interrupt service routine. Restarting the Watchdog and exiting by an RETI or RET, would return the processor to the lost position prior to the interrupt. But using the Watchdog reset function, the processor is restarted from the beginning of the program, and therefore placed into a known state.

7.6 Watchdog Related Registers

The Watchdog Timer has several SFR bits that are used during its operation. These bits can be utilized as a reset source, interrupt source, software polled timer or any combination of the three. Both the reset and interrupt have status flags. The Watchdog also has a bit which restarts the timer. The table below shows the bit locations with descriptions.

Table 7.2 Summary for Watchdog Related Bits

Bit Name	Register	Bit Position	Description
EWDI	EIE	EIE.4	Enable Watchdog Timer Interrupt
PWDI	EIP	EIP.4	Priority of Watchdog Timer Interrupt
WD[1:0]	CKCON	CKCON.7-6	Watchdog Interval
RWT	WDCON	WDCON.0	Reset Watchdog Timer
EWT		WDCON.1	Enable Watchdog Timer reset
WTRF		WDCON.2	Watchdog Timer reset flag
WDIF		WDCON.3	Watchdog Interrupt flag

The Watchdog Timer is not disabled during a Watchdog timeout reset, but it restarts the timer. In general, the software should set the Watchdog to a desired state. Control bits that support Watchdog operation are described in next subsections.

7.7 Watchdog Control

Watchdog control bits are described below. Please note that access (write) to this register has to be performed using '7.8 Timed Access Registers' procedure.

WDCON (0xD8)								
7	6	5	4	3	2	1	0	Reset
-	-	-	-	WDIF	WTRF	EWT	RWT	0x00

Figure 7.6 Watchdog Control Register

Note: WDIF - Watchdog Interrupt Flag. WDIF in conjunction with the Enable Watchdog Interrupt bit (EIE.4) and EWT indicates if a Watchdog Timer event has occurred and the action to be taken. This bit must be cleared by software before exiting the interrupt service routine, or another interrupt is generated. Setting WDIF in software will generate a Watchdog interrupt if enabled. This bit can be modified by using the **Timed Access Registers**.

WTRF - Watchdog Timer reset Flag. A Watchdog Timer reset has occurred when this flag is enabled by hardware; however, when using software to enable this flag, the Watchdog Timer reset is not triggered. During a reset, this flag is cleared otherwise it should be cleared by software. The Watchdog Timer has no effect on this bit if EWT bit is cleared.

EWT - Enable the Watchdog Timer reset. This bit controls the Watchdog Timer to reset the microcontroller, and has no effect on the ability of the Watchdog Timer to generate a Watchdog interrupt. Timed Access procedure must be used to modify this bit.

0 : Watchdog Timer timeout does not reset microcontroller

1 : Watchdog Timer timeout resets microcontroller

RWT - Reset the Watchdog Timer. Setting RWT resets the Watchdog Timer count. Timed Access procedure must be followed to enable this bit before the Watchdog Timer expires, reset or interrupt will be generated if the RWT is enabled.

Unimplemented bit - Read as 0 or 1

The table below summarizes Watchdog control bits and the functions.

Table 7.3 Watchdog Bits and Actions

EWT	EWDI	WDIF	Result
X	X	0	No Watchdog event.
0	0	1	Watchdog timeout has expired. No interrupt has been generated.
0	1	1	Watchdog interrupt has occurred.
1	0	1	Watchdog timeout has expired. No interrupt has been generated. Watchdog Timer reset will occur in 512 clock periods (CLK pin) if RWT is not strobed.
1	1	1	Watchdog interrupt has occurred. Watchdog Timer reset will occur in 512 clock periods (CLK pin) if RWT is not set using Timed Access procedure.

7.7.1 Clock Control

The Watchdog timeout selection is made using bits WD[1:0] as shown in the Figure below.

CKCON (0x8E)

7	6	5	4	3	2	1	0	Reset
WD1	WD0	-	-	-	MD2	MD1	MD0	0x03

Figure 7.7 Clock Control register - Watchdog bits

Clock control register CKCON(0x8E) contains WD[1:0] bits to select Watchdog Timer timeout period. The Watchdog is clocked directly from the CLK pin, and the PMM mode directly affects its timeout period.

The Watchdog has four timeout selections based on the input CLK clock frequency as shown in the Figure 7.1. The selections are a pre-selected number of clocks. Therefore, the actual timeout interval is dependent on the CLK frequency.

Table 7.4 Watchdog Intervals

WD[1:0]	Watchdog Interval	Number of Clocks
00	2^{17}	131072
01	2^{20}	1048576
10	2^{23}	8388608
11	2^{26}	67108864

Note that the time period shown above is for the interrupt events. When the reset is

enabled, it will activate 512 clocks later regardless of the interrupt. Therefore, the actual Watchdog timeout is the number of clocks chosen from Watchdog intervals plus 512 clocks (always CLK pin).

7.8 Timed Access Registers

Timed Access registers have a built-in mechanism that prevents them from accidental writes. TA is located at 0xC7 SFR address. To properly write to such registers, the following sequence has to be used :

MOV TA, #0xAA

MOV TA, #0x55

;Any direct addressing instruction writing timed access register

Any number of program wait states is allowed for the time elapsed between first, second, and third operations. Only correct sequence is required. Any third instruction causes protection mechanism to be turned on. This means that time protected register is opened for write only for a single instruction. Reading from such register is never protected. Timed Access registers are listed in the table below.

Table 7.5 Timed Access Registers

Register name	Description
WDCON(0xD8)	Watchdog configuration

8 TCPIPCore

8.1 Memory Map

TCPIPCore is composed of Common Register, SOCKET Register, TX Memory, and RX Memory as shown below.

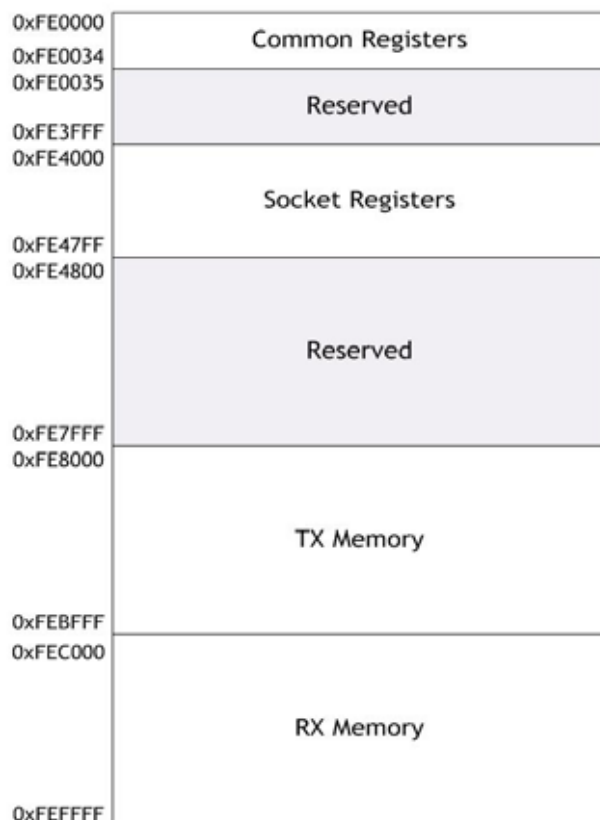


Figure 8.1 TCPIPCore Memory Map

8.2 TCPIPCore Registers

8.2.1 Common Registers

Address offset	Symbol	Description
0xFE0000	MR	Mode Register
0xFE0001	GAR0	GAR (Gateway Address Register)
0xFE0002	GAR1	
0xFE0003	GAR2	
0xFE0004	GAR3	
0xFE0005	SUBR0	SUBR (Subnet Mask Register)

0xFE0006	SUBR1	SUBR (Subnet Mask Register)
0xFE0007	SUBR2	
0xFE0008	SUBR3	
0xFE0009	SHAR0	SHAR (Source Hardware Address Register)
0xFE000A	SHAR1	
0xFE000B	SHAR2	
0xFE000C	SHAR3	
0xFE000D	SHAR4	
0xFE000E	SHAR5	
0xFE000F	SIPR0	SIPR (Source IP Address Register)
0xFE0010	SIPR1	
0xFE0011	SIPR2	
0xFE0012	SIPR3	
0xFE0013		Reserved
0xFE0014		
0xFE0015	IR	Interrupt Register
0xFE0016	IMR	Interrupt Mask Register
0xFE0017	RTR0	RTR (Retransmission Timeout-value Register)
0xFE0018	RTR1	
0xFE0019	RCR	RCR (Retransmission Retry-count Register)
0xFE001A		Reserved
0xFE001B		
0xFE001C	PATR0	PART (PPPoE Authentication Register)
0xFE001D	PATR1	
0xFE001E	PPPALGO	PPPoE Authentication Algorithm Register
0xFE001F	VERSIONR	W7100 Version Register
0xFE0020		Reserved
0xFE0027		
0xFE0028	PTIMER	PPP Link Control Protocol Request Timer Register
0xFE0029	PMAGIC	PPP LCP Magic Number Register
0xFE002A	UIPR0	UIPR (Unreachable IP Address Register in UDP mode)
0xFE002B	UIPR1	

0xFE002C	UIPR2	
0xFE002D	UIPR3	
0xFE002E	UPORT0	UPORT (Unreachable Port Register in UDP mode)
0xFE002F	UPORT1	
0xFE0030 ~ 0xFE0033		Reserved
0xFE0034	IR2	SOCKET Interrupt Register

8.2.2 SOCKET Registers

Address offset	Symbol	Description
0xFE4000	S0_MR	SOCKET 0 Mode Register
0xFE4001	S0_CR	SOCKET 0 Command Register
0xFE4002	S0_IR	SOCKET 0 Interrupt Register
0xFE4003	S0_SR	SOCKET 0 SOCKET Status Register
0xFE4004	S0_PORT0	S0_PORT (SOCKET 0 Source Port Register)
0xFE4005	S0_PORT1	
0xFE4006	S0_DHAR0	S0_DHAR (SOCKET 0 Destination Hardware Address Register)
0xFE4007	S0_DHAR1	
0xFE4008	S0_DHAR2	
0xFE4009	S0_DHAR3	
0xFE400A	S0_DHAR4	
0xFE400B	S0_DHAR5	
0xFE400C	S0_DIPR0	S0_DIPR (SOCKET 0 Destination IP Address Register)
0xFE400D	S0_DIPR1	
0xFE400E	S0_DIPR2	
0xFE400F	S0_DIPR3	
0xFE4010	S0_DPORT0	S0_DPORT (SOCKET 0 Destination Port Register)
0xFE4011	S0_DPORT1	
0xFE4012	S0_MSSR0	S0_MSSR (SOCKET 0 Maximum Segment Size Register)
0xFE4013	S0_MSSR1	
0xFE4014	S0_PROTO	SOCKET 0 Protocol of IP Header Field Register in IP raw mode

0xFE4015	S0_TOS	SOCKET 0 IP Type of Service(TOS) Register
0xFE4016	S0_TTL	SOCKET 0 IP Time to Live(TTL) Register
0xFE4017 ~ 0xFE401D		Reserved
0xFE401E	S0_RXMEM_SIZE	SOCKET 0 Receive Memory Size Register
0xFE401F	S0_TXMEM_SIZE	SOCKET 0 Transmit Memory Size Register
0xFE4020	S0_TX_FSR0	S0_TX_FSR (SOCKET 0 Transmit Free Memory Size Register)
0xFE4021	S0_TX_FSR1	
0xFE4022	S0_TX_RD0	S0_TX_RD0 (SOCKET 0 Transmit Memory Read Pointer Register)
0xFE4023	S0_TX_RD1	
0xFE4024	S0_TX_WR0	S0_TX_WR (SOCKET 0 Transmit Memory Write Pointer Register)
0xFE4025	S0_TX_WR1	
0xFE4026	S0_RX_RSR0	S0_RX_RSR (SOCKET 0 Received Data Size Register)
0xFE4027	S0_RX_RSR1	
0xFE4028	S0_RX_RD	S0_RX_RD (SOCKET 0 Receive Memory Read Pointer Register)
0xFE4029	S0_RX_RD1	
0xFE402A	S0_RX_WR	S0_RX_WR (SOCKET 0 Receive Memory Write Pointer Register)
0xFE402B	S0_RX_WR1	
0xFE402C	S0_IMR	SOCKET 0 Interrupt Mask Register
0xFE402D	S0_FRAG0	S0_FRAG (SOCKET 0 Fragment Field Value in IP Header Register)
0xFE402E	S0_FRAG1	
0xFE402F ~ 0xFE40FF		Reserved
0xFE4100	S1_MR	SOCKET 1 Mode Register
0xFE4101	S1_CR	SOCKET 1 Command Register
0xFE4102	S1_IR	SOCKET 1 Interrupt Register
0xFE4103	S1_SR	SOCKET 1 SOCKET Status Register
0xFE4104	S1_PORT0	S1_PORT (SOCKET 1 Source Port Register)
0xFE4105	S1_PORT1	

0xFE4106	S1_DHAR0	S1_DHAR (SOCKET 1 Destination Hardware Address Register)
0xFE4107	S1_DHAR1	
0xFE4108	S1_DHAR2	
0xFE4109	S1_DHAR3	
0xFE410A	S1_DHAR4	
0xFE410B	S1_DHAR5	
0xFE410C	S1_DIPR0	S1_DIPR (SOCKET 1 Destination IP Address Register)
0xFE410D	S1_DIPR1	
0xFE410E	S1_DIPR2	
0xFE410F	S1_DIPR3	
0xFE4110	S1_DPORT0	S1_DPORT (SOCKET 1 Destination Port Register)
0xFE4111	S1_DPORT1	
0xFE4112	S1_MSSR0	S1_MSSR (SOCKET 1 Maximum Segment Size Register)
0xFE4113	S1_MSSR1	
0xFE4114	S1_PROTO	SOCKET 1 Protocol of IP Header Field Register in IP raw mode
0xFE4115	S1_TOS	SOCKET 1 IP Type of Service(TOS) Register
0xFE4116	S1_TTL	SOCKET 1 IP Time to Live(TTL) Register
0xFE4117 ~ 0xFE411D		Reserved
0xFE411E	S1_RXMEM_SIZE	SOCKET 1 Receive Memory Size Register
0xFE411F	S1_TXMEM_SIZE	SOCKET 1 Transmit Memory Size Register
0xFE4120	S1_TX_FSR0	S1_TX_FSR (SOCKET 1 Transmit Free Memory Size Register)
0xFE4121	S1_TX_FSR1	
0xFE4122	S1_TX_RD0	S1_TX_RD
0xFE4123	S1_TX_RD1	(SOCKET 1 Transmit Memory Read Pointer Register)
0xFE4124	S1_TX_WR0	S1_TX_WR
0xFE4125	S1_TX_WR1	(SOCKET 1 Transmit Memory Write Pointer Register)
0xFE4126	S1_RX_RSR0	S1_RX_RSR
0xFE4127	S1_RX_RSR1	(SOCKET 1 Received Data Size Register)
0xFE4128	S1_RX_RD0	S1_RX_RD
0xFE4129	S1_RX_RD1	(SOCKET 1 Receive Memory Read Pointer Register)
0xFE412A	S1_RX_WR0	S1_RX_WR (SOCKET 1 Receive Memory Write Pointer

0xFE412B	S1_RX_WR1	Register)
0xFE412C	S1_IMR	SOCKET 1 Interrupt Mask Register
0xFE412D	S1_FRAG0	S1_FRAG (SOCKET 1 Fragment Field Value in IP Header Register)
0xFE412E	S1_FRAG1	
0xFE412F ~ 0xFE41FF		Reserved
0xFE4200	S2_MR	SOCKET 2 Mode Register
0xFE4201	S2_CR	SOCKET 2 Command Register
0xFE4202	S2_IR	SOCKET 2 Interrupt Register
0xFE4203	S2_SR	SOCKET 2 SOCKET Status Register
0xFE4204	S2_PORT0	S2_PORT (SOCKET 2 Source Port Register)
0xFE4205	S2_PORT1	
0xFE4206	S2_DHAR0	S2_DHAR (SOCKET 2 Destination Hardware Address Register)
0xFE4207	S2_DHAR1	
0xFE4208	S2_DHAR2	
0xFE4209	S2_DHAR3	
0xFE420A	S2_DHAR4	
0xFE420B	S2_DHAR5	
0xFE420C	S2_DIPR0	S2_DIPR (SOCKET 2 Destination IP Address Register)
0xFE420D	S2_DIPR1	
0xFE420E	S2_DIPR2	
0xFE420F	S2_DIPR3	
0xFE4210	S2_DPORT0	S2_DPORT (SOCKET 2 Destination Port Register)
0xFE4211	S2_DPORT1	
0xFE4212	S2_MSSR0	S2_MSSR (SOCKET 2 Maximum Segment Size Register)
0xFE4213	S2_MSSR1	
0xFE4214	S2_PROTO0	S2_PROTO (SOCKET 2 Protocol of IP Header Field Register in IP raw mode)
	S2_PROTO1	
0xFE4215	S2_TOS	SOCKET 2 IP Type of Service(TOS) Register
0xFE4216	S2_TTL	SOCKET 2 IP Time to Live(TTL) Register

0xFE4217 ~ 0xFE421D		Reserved
0xFE421E	S2_RXMEM_SIZE	SOCKET 2 Receive Memory Size Register
0xFE421F	S2_TXMEM_SIZE	SOCKET 2 Transmit Memory Size Register
0xFE4220	S2_TX_FSR0	S2_TX_FSR (SOCKET 2 Transmit Free Memory Size Register)
0xFE4221	S2_TX_FSR1	
0xFE4222	S2_TX_RD0	S2_TX_RD (SOCKET 2 Transmit Memory Read Pointer Register)
0xFE4223	S2_TX_RD1	
0xFE4224	S2_TX_WR0	S2_TX_WR (SOCKET 2 Transmit Memory Write Pointer Register)
0xFE4225	S2_TX_WR1	
0xFE4226	S2_RX_RSR0	S2_RX_RSR (SOCKET 2 Received Data Size Register)
0xFE4227	S2_RX_RSR1	
0xFE4228	S2_RX_RD0	S2_RX_RD (SOCKET 2 Receive Memory Read Pointer Register)
0xFE4229	S2_RX_RD1	
0xFE422A	S2_RX_WR0	S2_RX_WR (SOCKET 2 Receive Memory Write Pointer Register)
0xFE422B	S2_RX_WR1	
0xFE422C	S2_IMR	SOCKET 2 Interrupt Mask Register
0xFE422D	S2_FRAG0	SOCKET 2 Fragment Field Value in IP Header Register
0xFE422E	S2_FRAG1	
0xFE422F ~ 0xFE42FF		Reserved
0xFE4300	S3_MR	SOCKET 3 Mode Register
0xFE4301	S3_CR	SOCKET 3 Command Register
0xFE4302	S3_IR	SOCKET 3 Interrupt Register
0xFE4303	S3_SR	SOCKET 3 SOCKET Status Register
0xFE4304	S3_PORT0	S3_PORT (SOCKET 3 Source Port Register)
0xFE4305	S3_PORT1	
0xFE4306	S3_DHAR0	S3_DHAR (SOCKET 3 Destination Hardware Address Register)
0xFE4307	S3_DHAR1	

0xFE4308	S3_DHAR2	S3_DHAR (SOCKET 3 Destination Hardware Address Register)
0xFE4309	S3_DHAR3	
0xFE430A	S3_DHAR4	
0xFE430B	S3_DHAR5	
0xFE430C	S3_DIPR0	S3_DIPR (SOCKET 3 Destination IP Address Register)
0xFE430D	S3_DIPR1	
0xFE430E	S3_DIPR2	
0xFE430F	S3_DIPR3	
0xFE4310	S3_DPORT0	S3_DPORT (SOCKET 3 Destination Port Register)
0xFE4311	S3_DPORT1	
0xFE4312	S3_MSSR0	S3_MSSR (SOCKET 3 Maximum Segment Size Register)
0xFE4313	S3_MSSR1	
0xFE4314	S3_PROTO	SOCKET 3 Protocol of IP Header Field Register in IP raw mode
0xFE4315	S3_TOS	SOCKET 3 IP Type of Service(TOS) Register
0xFE4316	S0_TTL	SOCKET 3 IP Time to Live(TTL) Register
0xFE4317 ~ 0xFE431D		Reserved
0xFE431E	S3_RXMEM_SIZE	SOCKET 3 Receive Memory Size Register
0xFE431F	S3_TXMEM_SIZE	SOCKET 3 Transmit Memory Size Register
0xFE4320	S3_TX_FSR0	S3_TX_FSR (SOCKET 3 Transmit Free Memory Size Register)
0xFE4321	S3_TX_FSR1	
0xFE4322	S3_TX_RD0	S3_TX_RD (SOCKET 3 Transmit Memory Read Pointer Register)
0xFE4323	S3_TX_RD1	
0xFE4324	S3_TX_WR0	S3_TX_WR (SOCKET 3 Transmit Memory Write Pointer Register)
0xFE4325	S3_TX_WR1	
0xFE4326	S3_RX_RSR0	S3_RX_RSR (SOCKET 3 Received Data Size Register)
0xFE4327	S3_RX_RSR1	
0xFE4328	S3_RX_RD0	S3_RX_RD (SOCKET 3 Receive Memory Read Pointer Register)
0xFE4329	S3_RX_RD1	
0xFE432A	S3_RX_WR0	S3_RX_WR (SOCKET 3 Receive Memory Write Pointer Register)
0xFE432B	S3_RX_WR1	
0xFE432C	S3_IMR	SOCKET 3 Interrupt Mask Register

0xFE432D	S3_FRAG0	SOCKET 3 Fragment Field Value in IP Header Register
0xFE432E	S3_FRAG1	
0xFE432F ~ 0xFE43FF		Reserved
0xFE4400	S4_MR	SOCKET 4 Mode Register
0xFE4401	S4_CR	SOCKET 4 Command Register
0xFE4402	S4_IR	SOCKET 4 Interrupt Register
0xFE4403	S4_SR	SOCKET 4 SOCKET Status Register
0xFE4404	S4_PORT0	S4_PORT (SOCKET 4 Source Port Register)
0xFE4405	S4_PORT1	
0xFE4406	S4_DHAR0	S4_DHAR (SOCKET 4 Destination Hardware Address Register)
0xFE4407	S4_DHAR1	
0xFE4408	S4_DHAR2	
0xFE4409	S4_DHAR3	
0xFE440A	S4_DHAR4	
0xFE440B	S4_DHAR5	
0xFE440C	S4_DIPR0	S4_DIPR (SOCKET 4 Destination IP Address Register)
0xFE440D	S4_DIPR1	
0xFE440E	S4_DIPR2	
0xFE440F	S4_DIPR3	
0xFE4410	S4_DPORT0	S4_DPORT (SOCKET 4 Destination Port Register)
0xFE4411	S4_DPORT1	
0xFE4412	S4_MSSR0	S4_MSSR (SOCKET 4 Maximum Segment Size Register)
0xFE4413	S4_MSSR1	
0xFE4414	S4_PROTO	SOCKET 4 Protocol of IP Header Field Register in IP raw mode
0xFE4415	S4_TOS	SOCKET 4 IP Type of Service(TOS) Register
0xFE4416	S4_TTL	SOCKET 4 IP Time to Live(TTL) Register
0xFE4417 ~ 0xFE441D		Reserved
0xFE441E	S4_RXMEM_SIZE	SOCKET 4 Receive Memory Size Register

0xFE441F	S4_TXMEM_SIZE	SOCKET 4 Transmit Memory Size Register
0xFE4420	S4_TX_FSR0	S4_TX_FSR (SOCKET 4 Transmit Free Memory Size Register)
0xFE4421	S4_TX_FSR1	
0xFE4422	S4_TX_RD0	S4_TX_RD (SOCKET 4 Transmit Memory Read Pointer Register)
0xFE4423	S4_TX_RD1	
0xFE4424	S4_TX_WR0	S4_TX_WR (SOCKET 4 Transmit Memory Write Pointer Register)
0xFE4425	S4_TX_WR1	
0xFE4426	S4_RX_RSR0	S4_RX_RSR (SOCKET 4 Received Data Size Register)
0xFE4427	S4_RX_RSR1	
0xFE4428	S4_RX_RD0	S4_RX_RD (SOCKET 4 Receive Memory Read Pointer Register)
0xFE4429	S4_RX_RD1	
0xFE442A	S4_RX_WR0	S4_RX_WR (SOCKET 4 Receive Memory Write Pointer Register)
0xFE442B	S4_RX_WR1	
0xFE442C	S4_IMR	SOCKET 4 Interrupt Mask Register
0xFE442D	S4_FRAG0	SOCKET 4 Fragment Field Value in IP Header Register
0xFE442E	S4_FRAG1	
0xFE442F ~ 0xFE44FF		Reserved
0xFE4500	S5_MR	SOCKET 5 Mode Register
0xFE4501	S5_CR	SOCKET 5 Command Register
0xFE4502	S5_IR	SOCKET 5 Interrupt Register
0xFE4503	S5_SR	SOCKET 5 SOCKET Status Register
0xFE4504	S5_PORT0	S5_PORT (SOCKET 5 Source Port Register)
0xFE4505	S5_PORT1	
0xFE4506	S5_DHAR0	S5_DHAR (SOCKET 5 Destination Hardware Address Register)
0xFE4507	S5_DHAR1	
0xFE4508	S5_DHAR2	
0xFE4509	S5_DHAR3	
0xFE450A	S5_DHAR4	
0xFE450B	S5_DHAR5	
0xFE450C	S5_DIPR0	S5_DIPR (SOCKET 5 Destination IP Address Register)

0xFE450D	S5_DIPR1	S5_DIPR (SOCKET 5 Destination IP Address Register)
0xFE450E	S5_DIPR2	
0xFE450F	S5_DIPR3	
0xFE4510	S5_DPORT0	S5_DPORT (SOCKET 5 Destination Port Register)
0xFE4511	S5_DPORT1	
0xFE4512	S5_MSSR0	S5_MSSR (SOCKET 5 Maximum Segment Size Register)
0xFE4513	S5_MSSR1	
0xFE4514	S5_PROTO	SOCKET 5 Protocol of IP Header Field Register in IP raw mode
0xFE4515	S5_TOS	SOCKET 5 IP Type of Service(TOS) Register
0xFE4516	S5_TTL	SOCKET 5 IP Time to Live(TTL) Register
0xFE4517 ~ 0xFE451D		Reserved
0xFE451E	S5_RXMEM_SIZE	SOCKET 5 Receive Memory Size Register
0xFE451F	S5_TXMEM_SIZE	SOCKET 5 Transmit Memory Size Register
0xFE4520	S5_TX_FSR0	S5_TX_FSR (SOCKET 5 Transmit Free Memory Size Register)
0xFE4521	S5_TX_FSR1	
0xFE4522	S5_TX_RD0	S5_TX_RD (SOCKET 5 Transmit Memory Read Pointer Register)
0xFE4523	S5_TX_RD1	
0xFE4524	S5_TX_WR0	S5_TX_WR (SOCKET 5 Transmit Memory Write Pointer Register)
0xFE4525	S5_TX_WR1	
0xFE4526	S5_RX_RSR0	S5_RX_RSR (SOCKET 5 Received Data Size Register)
0xFE4527	S5_RX_RSR1	
0xFE4528	S5_RX_RD0	S5_RX_RD (SOCKET 5 Receive Memory Read Pointer Register)
0xFE4529	S5_RX_RD1	
0xFE452A	S5_RX_WR0	S5_RX_WR (SOCKET 5 Receive Memory Write Pointer Register)
0xFE452B	S5_RX_WR1	
0xFE452C	S5_IMR	SOCKET 5 Interrupt Mask Register
0xFE452D	S5_FRAG0	S5_FRAG (SOCKET 5 Fragment Field Value in IP Header Register)
0xFE452E	S5_FRAG1	

0xFE452F ~ 0xFE45FF		Reserved
0xFE4600	S6_MR	SOCKET 6 Mode Register
0xFE4601	S6_CR	SOCKET 6 Command Register
0xFE4602	S6_IR	SOCKET 6 Interrupt Register
0xFE4603	S6_SR	SOCKET 6 SOCKET Status Register
0xFE4604	S6_PORT0	S6_PORT (SOCKET 6 Source Port Register)
0xFE4605	S6_PORT1	
0xFE4606	S6_DHAR0	S6_DHAR (SOCKET 6 Destination Hardware Address Register)
0xFE4607	S6_DHAR1	
0xFE4608	S6_DHAR2	
0xFE4609	S6_DHAR3	
0xFE460A	S6_DHAR4	
0xFE460B	S6_DHAR5	
0xFE460C	S6_DIPR0	S6_DIPR (SOCKET 6 Destination IP Address Register)
0xFE460D	S6_DIPR1	
0xFE460E	S6_DIPR2	
0xFE460F	S6_DIPR3	
0xFE4610	S6_DPORT0	S6_DPORT (SOCKET 6 Destination Port Register)
0xFE4611	S6_DPORT1	
0xFE4612	S6_MSSR0	S6_MSSR (SOCKET 6 Maximum Segment Size Register)
0xFE4613	S6_MSSR1	
0xFE4614	S6_PROTO	SOCKET 6 Protocol of IP Header Field Register in IP raw mode
0xFE4615	S6_TOS	SOCKET 6 IP Type of Service(TOS) Register
0xFE4616	S6_TTL	SOCKET 6 IP Time to Live(TTL) Register
0xFE4617 ~ 0xFE461D		Reserved

0xFE461E	S6_RXMEM_SIZE	SOCKET 6 Receive Memory Size Register
0xFE461F	S6_TXMEM_SIZE	SOCKET 6 Transmit Memory Size Register
0xFE4620	S6_TX_FSR0	S6_TX_FSR (SOCKET 6 Transmit Free Memory Size Register)
0xFE4621	S6_TX_FSR1	
0xFE4622	S6_TX_RD0	S6_TX_RD (SOCKET 6 Transmit Memory Read Pointer Register)
0xFE4623	S6_TX_RD1	
0xFE4624	S6_TX_WR0	S6_TX_WR (SOCKET 6 Transmit Memory Write Pointer Register)
0xFE4625	S6_TX_WR1	
0xFE4626	S6_RX_RSR0	S6_RX_RSR (SOCKET 6 Received Data Size Register)
0xFE4627	S6_RX_RSR1	
0xFE4628	S6_RX_RD0	S6_RX_RD (SOCKET 6 Receive Memory Read Pointer Register)
0xFE4629	S6_RX_RD1	
0xFE462A	S6_RX_WR0	S6_RX_WR (SOCKET 6 Receive Memory Write Pointer Register)
0xFE462B	S6_RX_WR1	
0xFE462C	S6_IMR	SOCKET 6 Interrupt Mask Register
0xFE462D	S6_FRAG0	S6_FRAG (SOCKET 6 Fragment Field Value in IP Header Register)
0xFE462E	S6_FRAG1	
0xFE462F ~ 0xFE46FF		Reserved
0xFE4700	S7_MR	SOCKET 7 Mode Register
0xFE4701	S7_CR	SOCKET 7 Command Register
0xFE4702	S7_IR	SOCKET 7 Interrupt Register
0xFE4703	S7_SR	SOCKET 7 SOCKET Status Register
0xFE4704	S7_PORT0	S7_PORT (SOCKET 7 Source Port Register)
0xFE4705	S7_PORT1	
0xFE4706	S7_DHAR0	S7_DHAR (SOCKET 7 Destination Hardware Address Register)
0xFE4707	S7_DHAR1	
0xFE4708	S7_DHAR2	
0xFE4709	S7_DHAR3	
0xFE470A	S7_DHAR4	
0xFE470B	S7_DHAR5	

0xFE470C	S7_DIPR0	S7_DIPR (SOCKET 7 Destination IP Address Register)
0xFE470D	S7_DIPR1	
0xFE470E	S7_DIPR2	
0xFE470F	S7_DIPR3	
0xFE4710	S7_DPORT0	S7_DPORT (SOCKET 7 Destination Port Register)
0xFE4711	S7_DPORT1	
0xFE4712	S7_MSSR0	S7_MSSR (SOCKET 7 Maximum Segment Size Register)
0xFE4713	S7_MSSR1	
0xFE4714	S0_PROTO	SOCKET 7 Protocol of IP Header Field Register in IP raw mode
0xFE4715	S7_TOS	SOCKET 7 IP Type of Service(TOS) Register
0xFE4716	S7_TTL	SOCKET 7 IP Time to Live(TTL) Register
0xFE4717 ~ 0xFE471D		Reserved
0xFE471E	S7_RXMEM_SIZE	SOCKET 7 Receive Memory Size Register
0xFE471F	S7_TXMEM_SIZE	SOCKET 7 Transmit Memory Size Register
0xFE4720	S7_TX_FSR0	S7_TX_FSR (SOCKET 7 Transmit Free Memory Size Register)
0xFE4721	S7_TX_FSR1	
0xFE4722	S7_TX_RD0	S7_TX_RD
0xFE4723	S7_TX_RD1	(SOCKET 7 Transmit Memory Read Pointer Register)
0xFE4724	S7_TX_WR0	S7_TX_WR
0xFE4725	S7_TX_WR1	(SOCKET 7 Transmit Memory Write Pointer Register)
0xFE4726	S7_RX_RSR0	S7_RX_RSR (SOCKET 7 Received Data Size Register)
0xFE4727	S7_RX_RSR1	
0xFE4728	S7_RX_RD0	S7_RX_RD
0xFE4729	S7_RX_RD1	(SOCKET 7 Receive Memory Read Pointer Register)
0xFE472A	S7_RX_WR0	S7_RX_WR
0xFE472B	S7_RX_WR1	(SOCKET 7 Receive Memory Write Pointer Register)
0xFE472C	S7_IMR	SOCKET 7 Interrupt Mask Register
0xFE472D	S7_FRAG0	S7_FRAG
0xFE472E	S7_FRAG1	(SOCKET 7 Fragment Field Value in IP Header Register)

0xFE472F ~ 0xFE47FF		Reserved
---------------------------	--	----------

8.3 Register Description

8.3.1 Mode Register

MR (Mode Register) [R/W] [0xFE0000] [0x00]

This register is used for S/W reset, ping block mode and PPPoE mode.

7	6	5	4	3	2	1	0
RST			PB	PPPoE			

Bit	Symbol	Description
7	RST	S/W Reset If this bit is '1', internal register will be initialized. It will be automatically cleared after reset.
6	Reserved	Reserved
5	Reserved	Reserved
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 a user uses ADSL without router or etc, the bit should be set as '1' to connect to ADSL Server. For more detail, refer to the application note, "How to connect ADSL".
2	Reserved	Reserved
1	Reserved	Reserved
0	Reserved	Reserved

GAR (Gateway IP Address Register) [R/W] [0xFE0001 - 0xFE0004] [0x00]

This Register sets up the default gateway address.

Ex) In case of "192.168.0.1"

0xFE0001	0xFE0002	0xFE0003	0xFE0004
192 (0xC0)	168 (0xA8)	0 (0x00)	1 (0x01)

SUBR (Subnet Mask Register) [R/W] [0xFE0005 - 0xFE0008] [0x00]

This register sets up the subnet mask address.

Ex) In case of "255.255.255.0"

0xFE0005	0xFE0006	0xFE0007	0xFE0008
255 (0xFF)	255 (0xFF)	255 (0xFF)	0 (0x00)

SHAR (Source Hardware Address Register) [R/W] [0xFE0009 - 0xFE000E] [0x00]

This register sets up the Source Hardware address.

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

0xFE0009	0xFE000A	0xFE000B	0xFE000C	0xFE000D	0xFE000E
0x00	0x08	0xDC	0x01	0x02	0x03

SIPR (Source IP Address Register) [R/W] [0xFE000F - 0xFE0012] [0x00]

This register sets up the Source IP address.

Ex) In case of "192.168.0.2"

0xFE000F	0xFE0010	0xFE0011	0xFE0012
192 (0xC0)	168 (0xA8)	0 (0x00)	2 (0x02)

IR (Interrupt Register) [R] [0xFE0015] [0x00]

This register is accessed by the MCU of W7100 to determine the cause of an interrupt. As long as any signal is set, the /INT signal retain low, and it will not go high until all bits is cleared in the **Interrupt Register**. It occurs the interrupt signal (nINT5: TCIPCore interrupt) to MCU.

7	6	5	4	3	2	1	0
CONFLICT	UNREACH	PPPoE	Reserved	Reserved	Reserved	Reserved	Reserved

Bit	Symbol	Description
7	CONFLICT	IP Conflict When the ARP request has the same IP address as the Source IP address, this bit is set as '1'. It can be cleared to '0' by writing '1' to this bit.
6	UNREACH	Destination unreachable W7100 will receive ICMP(Destination Unreachable) packet if non-existing destination IP address is transmitted during a UDP data transmission. In this case, the IP address and the port number will be saved in Unreachable IP Address (UIPR) and Unreachable Port Register (UPORT). The UNREACH bit will be set as '1'. This bit can be cleared to '0' by writing '1' to this bit.

5	PPPoE	PPPoE Connection Close In the PPPoE Mode, '1' is set if the PPPoE connection is closed. This bit can be cleared to '0' by writing '1' to this bit.
4	Reserved	Reserved
3	Reserved	Reserved
2	Reserved	Reserved
1	Reserved	Reserved
0	Reserved	Reserved

IMR (Interrupt Mask Register) [R/W] [0xFE0016] [0x00]

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Register2 (IR2). If an interrupt mask bit is set, an interrupt will be issued whenever the corresponding bit in the IR2 is set. If the bit of IMR is set as '0', corresponding interrupt will not be triggered by the enabled bit in the IR2.

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

RTR (Retry Time-value Register) [R/W] [0xFE0017 - 0xFE0018] [0x07D0]

This register sets the period of timeout. Value 1 means 100us. The default timeout is 200ms which has a value of 2000 (0x07D0).

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

0xFE0017	0xFE0018
0x0F	0xA0

Re-transmission will occur if there is no response or response is delayed from the remote peer.

RCR (Retry Count Register) [R/W] [0xFE0019] [0x08]

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

PATR (Authentication Type in PPPoE mode) [R] [0xFE001C-0xFE001D] [0x0000]

This register notifies the type of authentication used to establish the PPPoE connection. W7100 supports two types of Authentication method - PAP and CHAP.

Value	Authentication Type
0xC023	PAP
0xC223	CHAP

PPPALGO (Authentication Algorithm in PPPoE mode)[R][0xFE001E][0x00]

This register notifies the authentication algorithm used for the PPPoE connection. For detail information, please refer to PPPoE application note.

PTIMER (PPP Link Control Protocol Request Timer Register) [R/W] [0xFE0028] [0x28]

This register indicates the duration of LCP Echo Request being sent. Value 1 is about 25ms.

Ex) in case that PTIMER is 200,

$$200 * 25(\text{ms}) = 5000(\text{ms}) = 5 \text{ seconds}$$

PMAGIC (PPP Link Control Protocol Magic number Register) [R/W] [0xFE0029][0x00]

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

VERSIONR (W7100 Chip Version Register)[R][0xFE001F][0x02]

This register is W7100 chip version register.

UIPR (Unreachable IP Address Register) [R] [0xFE002A - 0xFE002D] [0x00]

When using UDP to transfer data, ICMP (Destination Unreachable) packet will be received if the destination IP address is non-existing. In this case, the IP address and port number will be saved in the Unreachable IP Address Register(UIPR) and Unreachable Port Register(UPORT) respectively.

Ex) For the case of "192.168.0.11"

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

UPORT (Unreachable Port Register) [R] [0xFE002E - 0xFE002F] [0x0000]

Refer to Unreachable IP Address Register (UIPR)

Ex) For the case of 5000(0x1388)

0xFE002E	0xFE002F
0x13	0x88

IR2 (W7100 SOCKET Interrupt Register)[R/W][0xFE0034][0x00]

IR2 is a Register which notifies the host that a W7100 SOCKET interrupt has occurred. When an interrupt occurs, the related bit in IR2 is enabled. In this case, the /INT signal is asserted low until all of the bits of IR2 is '0'. Once the IR2 register is cleared out by using the Sn_IR bits, the /INT signal is asserted high. It occurs the interrupt signal (nINT5: TCIPCore interrupt) to MCU.

7	6	5	4	3	2	1	0
S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT

Bit	Symbol	Description
7	S7_INT	Occurrence of SOCKET 7 Interrupt When an interrupt occurs at SOCKET 7, it becomes '1'. This interrupt information is applied to S7_IR2. This bit is automatically cleared when S7_IR2 is cleared to 0x00 by host.
6	S6_INT	Occurrence of SOCKET 6 Interrupt When an interrupt occurs at SOCKET 6, it becomes '1'. This interrupt information is applied to S6_IR2. This bit is automatically cleared when S6_IR2 is cleared to 0x00 by host.
5	S5_INT	Occurrence of SOCKET 5 Interrupt When an interrupt occurs at SOCKET 5, it becomes '1'. This interrupt information is applied to S5_IR2. This bit is automatically cleared when S5_IR2 is cleared to 0x00 by host.
4	S4_INT	Occurrence of SOCKET 4 Interrupt When an interrupt occurs at SOCKET 4, it becomes '1'. This interrupt information is applied to S4_IR2. This bit is automatically cleared when S4_IR2 is cleared to 0x00 by host.
3	S3_INT	Occurrence of SOCKET 3 Interrupt When an interrupt occurs at SOCKET 3, it becomes '1'. This interrupt information is applied to S3_IR2. This bit is automatically cleared when S3_IR2 is cleared to 0x00 by host.
2	S2_INT	Occurrence of SOCKET 2 Interrupt

		When an interrupt occurs at SOCKET 2, it becomes '1'. This interrupt information is applied to S2_IR2. This bit is automatically cleared when S2_IR2 is cleared to 0x00 by host.
1	S1_INT	Occurrence of SOCKET 1 Interrupt When an interrupt occurs at SOCKET 1, it becomes '1'. This interrupt information is applied to S1_IR2. This bit is automatically cleared when S1_IR2 is cleared to 0x00 by host.
0	S0_INT	Occurrence of SOCKET 0 Interrupt When an interrupt occurs at SOCKET 0, it becomes '0'. This interrupt information is applied to S0_IR2. This bit is automatically cleared when S0_IR2 is cleared to 0x00 by host.

8.3.2 SOCKET Registers

Sn_MR (SOCKET *n* Mode Register)[R/W][0xFE4000 + 0x100n][0x0000]

This register configures the protocol type or option of SOCKET *n*.

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

Bit	Symbol	Description
7	MULTI	Multicasting 0 : disable Multicasting 1 : enable Multicasting This only applies to UDP case(P3-P0 : "0010") To use multicasting, write multicast group address and port number to SOCKET <i>n</i> destination IP and port register respectively before using the OPEN command.
6	Reserved	Reserved
5	ND/MC	Use No Delayed ACK 0 : Disable No Delayed ACK option 1 : Enable No Delayed ACK option, This only applies to TCP case (P3-P0 : "0001") If this bit is set as '1', ACK packet is immediately transmitted after receiving data packet from a peer. If this bit is cleared, ACK packet is transmitted according to internal timeout mechanism. Multicast 0 : using IGMP version 2 1 : using IGMP version 1

		This bit is valid when MULTI bit is enabled and UDP mode is used (P3-P0 : “0010”). In addition, multicast can be used to send out the version number in IGMP messages such as Join/Leave/Report to multicast-group																																										
4	Reserved	Reserved																																										
3	P3	Protocol Sets up corresponding SOCKET as TCP, UDP, or IP RAW mode																																										
2	P2	<table><tr><th>Symbol</th><th>P3</th><th>P2</th><th>P1</th><th>P0</th><th>Meaning</th></tr><tr><td>Sn_MR_CLOSE</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Closed</td></tr><tr><td>Sn_MR_TCP</td><td>0</td><td>0</td><td>0</td><td>1</td><td>TCP</td></tr><tr><td>Sn_MR_UDP</td><td>0</td><td>0</td><td>1</td><td>0</td><td>UDP</td></tr><tr><td>Sn_MR_IPRAW</td><td>0</td><td>0</td><td>1</td><td>1</td><td>IPRAW</td></tr><tr><td>S0_MR_MACRAW</td><td>0</td><td>1</td><td>0</td><td>0</td><td>MAC RAW</td></tr><tr><td>S0_MR_PPPOE</td><td>0</td><td>1</td><td>0</td><td>1</td><td>PPPoE</td></tr></table>	Symbol	P3	P2	P1	P0	Meaning	Sn_MR_CLOSE	0	0	0	0	Closed	Sn_MR_TCP	0	0	0	1	TCP	Sn_MR_UDP	0	0	1	0	UDP	Sn_MR_IPRAW	0	0	1	1	IPRAW	S0_MR_MACRAW	0	1	0	0	MAC RAW	S0_MR_PPPOE	0	1	0	1	PPPoE
Symbol	P3	P2	P1	P0	Meaning																																							
Sn_MR_CLOSE	0	0	0	0	Closed																																							
Sn_MR_TCP	0	0	0	1	TCP																																							
Sn_MR_UDP	0	0	1	0	UDP																																							
Sn_MR_IPRAW	0	0	1	1	IPRAW																																							
S0_MR_MACRAW	0	1	0	0	MAC RAW																																							
S0_MR_PPPOE	0	1	0	1	PPPoE																																							
1	P1																																											
0	P0	S0_MR_MACRAW and S0_MR_PPPOE are valid only in SOCKET 0. S0_MR_PPPOE is temporarily used for PPPoE server connection/Termination. After connection is established, it can be utilized as another protocol.																																										

Sn_CR (SOCKET *n* Command Register)[R/W][0xFE4001 + 0x100n][0x00]

This is used to set the command for SOCKET *n* such as OPEN, CLOSE, CONNECT, LISTEN, SEND, and RECEIVE. After W7100 identifies the command, the Sn_CR register is automatically cleared to 0x00. Even though Sn_CR is cleared to 0x00, the command is still being processed. To verify whether the command is completed or not, please check the Sn_IR or Sn_SR registers.

Value	Symbol	Description														
0x01	OPEN	SOCKET <i>n</i> is initialized and opened according to the protocol selected in Sn_MR (P3:P0). The table below shows the value of Sn_SR corresponding to Sn_MR														
		<table><tr><th>Sn_MR(P3:P0)</th><th>Sn_SR</th></tr><tr><td>Sn_MR_CLOSE</td><td>-</td></tr><tr><td>Sn_MR_TCP</td><td>SOCK_INIT</td></tr><tr><td>Sn_MR_UDP</td><td>SOCK_UDP</td></tr><tr><td>Sn_MR_IPRAW</td><td>SOCK_IPRAW</td></tr><tr><td>S0_MR_MACRAW</td><td>SOCK_MACRAW</td></tr><tr><td>S0_MR_PPPOE</td><td>SOCK_PPPOE</td></tr></table>	Sn_MR(P3:P0)	Sn_SR	Sn_MR_CLOSE	-	Sn_MR_TCP	SOCK_INIT	Sn_MR_UDP	SOCK_UDP	Sn_MR_IPRAW	SOCK_IPRAW	S0_MR_MACRAW	SOCK_MACRAW	S0_MR_PPPOE	SOCK_PPPOE
		Sn_MR(P3:P0)	Sn_SR													
		Sn_MR_CLOSE	-													
		Sn_MR_TCP	SOCK_INIT													
		Sn_MR_UDP	SOCK_UDP													
		Sn_MR_IPRAW	SOCK_IPRAW													
		S0_MR_MACRAW	SOCK_MACRAW													
S0_MR_PPPOE	SOCK_PPPOE															

0x02	LISTEN	This is valid only in TCP mode (Sn_MR(P3:P0) = Sn_MR_TCP). In this mode, the SOCKET n is configured as a TCP server which is waiting for connection-request (SYN packet) from any "TCP CLIENT". The Sn_SR register changes the state from SOCK_INIT to SOCKET_LISTEN. When a client's connection request is successfully established, the Sn_SR changes from SOCK_LISTEN to SOCK_ESTABLISHED and the Sn_IR(0) becomes '1'. On the other hand, Sn_IR(3) is set as '1' and Sn_SR changes to SOCK_CLOSED during a connection failure(SYN/ACK packet failed to transfer) cf> If the destination port of the TCP Client does not exist during a connection request, W7100 will transmit a RST packet and Sn_SR is unchanged.
0x04	CONNECT	This mode is only valid in TCP mode and operates the SOCKET n as a TCP client. A connect-request (SYN packet) is sent to the TCP server by connecting to the IP address and port stored in destination address and port registers (Sn_DIPR0 and Sn_DPORT0) When a client's connection request is successfully established, the Sn_SR register is changed to SOCK_ESTABLISHED and the Sn_IR(0) becomes '1'. In the following cases, the connect-request fails - When a ARP_{TO} occurs (Sn_IR(s)='1') because the Destination Hardware Address is not acquired through the ARP process - When a SYN/ACK packet is not received and TCP_{TO}(Sn_IR(3)) is '1' - When a RST packet is received instead of a SYN/ACK packet Above three cases, Sn_SR is changed to SOCK_CLOSED.
0x08	DISCON	Only valid in TCP mode Regardless of "TCP SERVER" or "TCP CLIENT", this disconnect the process - Active close : it transmits disconnect-request(FIN packet) to the connected peer - Passive close : When FIN packet is received from peer, a FIN packet is replied back to the peer when FIN/ACK packet is received, Sn_SR is changed to SOCK_CLOSED. When a disconnect request is not received, TCP_{TO} occurs (Sn_IR(3)='1')

		and Sn_SR is changed to SOCK_CLOSED. cf> If CLOSE is used instead of DISCON, only Sn_SR is changed to SOCK_CLOSED without disconnect-process(disconnect-request). If a RST packet is received from a peer during communication, Sn_SR is unconditionally changed to SOCK_CLOSED.
0x10	CLOSE	Closes SOCKET <i>n</i> . Sn_SR is changed to SOCK_CLOSED.
0x20	SEND	SEND transmits all the data buffered in the TX memory. For more details, please refer to SOCKET <i>n</i> TX Free Size Register (Sn_TX_FSR0), SOCKET <i>n</i> TX Write Pointer Register(Sn_TX_WR0), and SOCKET <i>n</i> TX Read Pointer Register(Sn_TX_RD0).
0x21	SEND_MAC	Used in UDP mode only The basic operation is same as SEND. Normally SEND operation needs Destination Hardware Address which can be retrieved by the ARP (Address Resolution Protocol) process. SEND_MAC uses SOCKET <i>n</i> Destination Hardware Address(Sn_DHAR0) that is chosen by the user without going through the ARP process.
0x22	SEND_KEEP	Used in TCP mode It checks the connection status by sending 1byte data. If the connection has no response from peers or is terminated, the Timeout interrupt will occur.
0x40	RECV	RECV processes the data received by using a RX read pointer register(Sn_RX_RD). For more detail, please refer to 9.2.1.1 SERVER mode Receiving Process with SOCKET <i>n</i> RX Received Size Register (Sn_RX_RSR0), SOCKET <i>n</i> RX Write Pointer Register(Sn_RX_WR), and SOCKET <i>n</i> RX Read Pointer Register(Sn_RX_RD).

Below commands are only valid for SOCKET 0 and S0_MR(P3:P0) = S0_MR_PPpOE.

For more detail refer to the “How to use ADSL”.

Value	Symbol	Description
0x23	PCON	ADSL connection begins by transmitting PPpOE Discovery Packet
0x24	PDISCON	Closes ADSL connection
0x25	PCR	In each phase, it transmits REQ message
0x26	PCN	In each phase, it transmits NAK message
0x27	PCJ	In each phase, it transmits REJECT message

Sn_IR (SOCKET *n* Interrupt Register)[R/W][0xFE4002 + 0x100n][0x00]

Sn_IR register provides information such as the type of interrupt (establishment, termination, receiving data, timeout) used in SOCKET *n*.

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

In order to clear the Sn_IR bit, the host should write the bit as '1'. When all the bits of Sn_IR is cleared ('0'), IR(*n*) is automatically cleared. It occurs the interrupt signal (nINT5: TCPIPCore interrupt) to MCU.

7	6	5	4	3	2	1	0
PRECV	PFAIL	PNEXT	SEND_OK	TIMEOUT	RCV	DISCON	CON
Bit	Symbol	Description					
7	PRECV	PPP Receive Interrupts when the option which is not supported is received					
6	PFAIL	PPP Fail Interrupts when PAP Authentication is failed					
5	PNEXT	PPP Next Phase Interrupts when the phase is changed during ADSL connection process					
4	SENDOK	SEND OK Interrupts when the SEND command is completed					
3	TIMEOUT	TIMEOUT Interrupts when ARP _{TO} or TCP _{TO} occurs					
2	RCV	Receive Interrupts whenever data packet is received from a peer					
1	DISCON	Disconnect Interrupts when FIN of FIN/ACK packet is received from a peer					
0	CON	Connect Interrupts when a connection is established with a peer					

Sn_SR (SOCKET *n* Status Register)[R][0xFE4003 + 0x100n][0x00]

This register provides the status of SOCKET *n*. SOCKET status are changed when using the Sn_CR register or during packet transmission/reception.

The table below describes the different states of SOCKET *n*

Value	Symbol	Description
0x00	SOCK_CLOSED	When DISCON or CLOSE command is used, or ARP _{TO} , or TCP _{TO} occurs, the state changes to SOCK_CLOSED regardless of previous value.
0x13	SOCK_INIT	In this state, the SOCKET <i>n</i> is opened in TCP mode and initialized the first step of TCP connection establishment. Now, the user can use the LISTEN and CONNECT commands. When Sn_MR(P3:P0) is Sn_MR_TCP and the OPEN command is

		used, the stage changes to SOCK_INIT.
0x14	SOCK_LISTEN	SOCKET n operates in TCP Server Mode and waits for a connection-request (SYN packet) from a "TCP CLIENT". When the LISTEN command is used, the stage changes to SOCK_LISTEN Once the connection is established, the SOCKET state changes from SOCK_LISTEN to SOCK_ESTABLISHED; however, if the connection fails, TCP_{TO} occurs ($Sn_IR(TIME_OUT) = '1'$) and the state changes to SOCK_CLOSED.
0x17	SOCK_ESTABLISHED	When a SYN packet is received from a TCP client, the socket changes from the SOCK_LISTEN or CONNECTS state to the SOCK_ESTABLISHED state. At this stage, DATA packets can be exchanged by using the SEND or RECV command.
0x1C	SOCK_CLOSE_WAIT	In this case, a disconnect-request (FIN packet) is received from a peer. Although the TCP connection is half-closed, data packet can still be transferred. In order to complete the TCP disconnection, the DISCON command should be used. When a socket is closed without going through the disconnection-process, the CLOSE command should be used.
0x22	SOCK_UDP	The socket is opened in UDP mode. The SOCKET status is changed to SOCK_UDP when $Sn_MR(P3:P0)$ is Sn_MR_UDP and OPEN command is used. Unlike TCP mode, the SOCKET in UDP mode can transfer data without establishing a connection (3 way handshake)
0x32	SOCK_IPRAW	The socket is opened in IPRAW mode. The SOCKET status is change to SOCK_IPRAW when $Sn_MR(P3:P0)$ is Sn_MR_IPRAW and OPEN command is used. IP Packet can be transferred without a connection similar to the UDP mode.
0x42	SOCK_MACRAW	SOCKET 0 is opened in MACRAW mode. The SOCKET status is change to SOCK_MACRAW when $S0_IMR(P3:P0)$ is $S0_MR_MACRAW$ and $S0_CR = OPEN$. MAC packet(Ethernet frame) can be transferred similar to UDP mode.
0x5F	SOCK_PPPOE	SOCKET 0 is opened in PPPoE mode. The SOCKET status is changed to SOCK_PPPOE when $S0_MR(P3:P0) = S0_MR_PPPoE$ and $S0_CR = OPEN$.

Below table shows the temporary status which can be observed when the Sn_SR is changed.

Value	Symbol	Description
0x15	SOCK_SYSENT	This status indicates that a connect-request(SYN packet) is sent to a "TCP SERVER". SYSENT is an intermediate state between SOCK_INT and SOCK_ESTABLISHED. If connect-accept(SYN/ACK packet) is received from a "TCP SERVER", the SOCKET status automatically changes to SOCK_ESTBLISHED. However, if SYN/ACK packet is not received before TCP timeout occurs (Sn_IR(TIMEOUT)='1'), the status is changed to SOCK_CLOSED.
0x16	SOCK_SYNRCV	This status indicate that a connect-request(SYN packet) is received from a "TCP CLIENT". The socket status changes to SOCK_ESTABLISHED when W7100 successfully transmits connect-accept (SYN/ACK packet) to a "TCP CLIENT". If W7100 fails to send and TCP_{TO} occurs (Sn_IR(TIMEOUT)='1'), the status is changed to SOCK_CLOSED.
0x18	SOCK_FIN_WAIT	These statues shows the process of terminating a connection. If the termination succeeds or Timeout interrupt is asserted, the socket status is changed to SOCK_CLOSED.
0x1A	SOCK_CLOSING	
0X1B	SOCK_TIME_WAIT	
0X1D	SOCK_LAST_ACK	
0x11 0x21 0x31	SOCK_ARP	This status indicates an ARP-request is being transmitted to a peer in order to acquire destination hardware address. It appears when the SEND command is used in UDP, IP RAW, and TCP mode, the socket status changes to SOCK_ARP. If the hardware address is successfully acquired from the destination (ARP-response is received), the socket status changes to SOCK_UDP, SOCK_IPRAW or SOCK_SYSENT. On the other hand, when W7100 fails to acquire the hardware address and an ARP timeout occurs (Sn_IR(TIMEOUT)= '1'), the socket status returns to the previous state in UDP mode and IP RAW mode. In TCP mode, the socket status goes to the SOCK_CLOSED state. cf> In UDP and IP RAW mode, the Sn_DIPR register compares the previous and current values. ARP is only used if the two values are different.

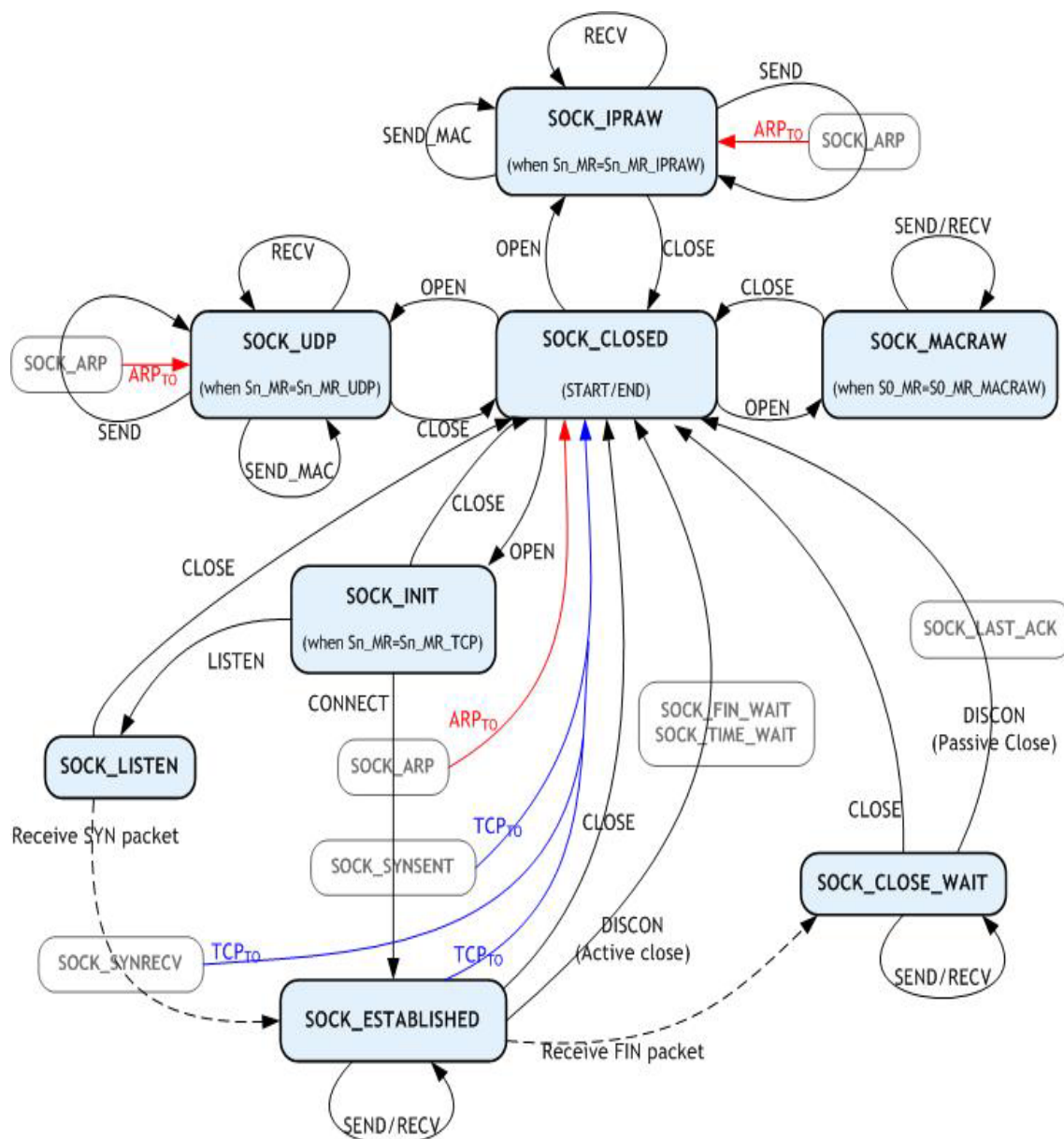


Figure 8.2 SOCKET *n* Status transition

Sn_PORT (SOCKET *n* Source Port Register)[R/W] [(0xFE4004 + 0x100*n*) - (0xFE4005 + 0x100*n*)] [0x0000]

It sets source port number.

It is valid when SOCKET *n* is used as TCP or UDP mode, and ignored when used as other modes.

It should be set before OPEN command.

Ex) In case of SOCKET 0 port = 5000(0x1388), configure as below,

0xFE4004	0xFE4005
0x13	0x88

Sn_DHAR (SOCKET *n* Destination Hardware Address Register)[R/W] [(0xFE4006 + 0x100n) - (0xFE400B + 0x100n)][FF.FF.FF.FF.FF]

It sets or is set as destination hardware address of SOCKET *n*. Also, if SOCKET 0 is used for PPPoE mode, S0_DHAR sets as PPPoE server hardware an address that is already known.

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

When using PPPoE-process of W7100, PPPoE server hardware address is not required to be set.

However, even if PPPoE-process of W7100 is not used, but implemented by yourself with MACRAW mode, in order to transmit or receive the PPPoE packet, PPPoE server hardware address(acquired by your PPPoE-process), PPPoE server IP address, and PPP session ID should be set, and MR(PPPoE) also should be set as '1'.

S0_DHAR sets the PPPoE server hardware address before the OPEN command. PPPoE server hardware address which is set by S0_DHAR is applied to PDHAR after performing the OPEN command. The configured PPPoE information is internally valid even after the CLOSE command.

EX) In case of SOCKET 0 Destination Hardware address = 00.08.DC.01.02.10, configuration is as below,

0xFE4006	0xFE4007	0xFE4008	0xFE4009	0xFE400A	0xFE400B
0x00	0x08	0xDC	0x01	0x02	0x10

Sn_DIPR (SOCKET *n* Destination IP Address Register)[R/W] [(0xFE400C + 0x100n) - (0xFE400F + 0x100n)][00.00.00.00]

It sets or is set as destination IP address of SOCKET *n*. If SOCKET 0 is used as PPPoE mode, S0_DIPR0 sets PPPoE server with an IP address that is already known.

It is valid only in TCP, UDP, IPRAW or PPPoE mode, but ignored in MACRAW mode.

In the TCP mode, when operating as "TCP CLIENT" it sets the IP address of the "TCP SERVER" before performing the CONNECT command and when operating as "TCP SERVER", it internally sets the IP address of the "TCP CLIENT" after successfully establishing connection.

In UDP or IPRAW mode, set the destination IP address in the Sn_DIPR for transmitting UDP or IPRAW DATA packets before performing SEND or SEND_MAC command. At the PPPoE mode, S0_DIPR sets as PPPoE server IP address that is already known.

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

0xFE400C	0xFE400D	0xFE400E	0xFE400F
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

Sn_DPORT (SOCKET *n* Destination Port Register)[R/W] [(0xFE4010 + 0x100*n*) - (0xFE4011 + 0x100*n*)] [0x0000]

The destination port number is set in the Sn_DPORT of SOCKET *n*. If SOCKET 0 is used as PPPoE mode, S0_DPORT0 sets PPP session ID that is already known.

It is valid only in TCP, UDP or PPPoE mode, and ignored in other modes.

At the TCP mode, when operating as "TCP CLIENT", it listens for the port number of the "TCP SERVER" before performing the CONNECT command.

At the UDP mode, the destination port number is set in the Sn_DPORT to be used for transmitting UDP DATA packets before performing SEND or SEND_MAC command.

At the PPPoE mode, the PPP session ID that is already known is set in the S0_DPORT. PPP session ID (set by S0_DPORT0) is applied to PSIDR after performing the OPEN command.

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

0xFE4010	0xFE4011
0x13	0x88

Sn_MSSR (SOCKET *n* Maximum Segment Size Register)[R/W] [(0xFE4012 + 0x100*n*) - (0xFE4013 + 0x100*n*)] [0x0000]

It sets the MTU (Maximum Transfer Unit) of SOCKET *n* or notifies the MTU that is already set.

If the host does not set the Sn_MSSR, it is set as default MTU.

It supports TCP or UDP mode. When using PPPoE (MR(PPPoE)='1'), the MTU of the TCP or UDP mode is assigned in the range of the MTU of PPPoE.

At the IPRAW or MACRAW, MTU is not processed internally, but the default MTU is used. Therefore, when transmitting data bigger than the default MTU, the host should manually divide the data into the default MTU unit.

At the TCP or UDP mode, if transmitting data is bigger than the MTU, W7100 automatically divides the data into the MTU unit.

MTU is known as MSS in the TCP mode. By selecting from the Host-Written-Value and the peer's MSS, MSS is automatically set as the smaller value through the TCP connection process.

At the UDP mode, there is no connection-process of TCP mode, and Host-Written-Value is just used. When communicating with the peer having a different MTU, W7100 is able to receive ICMP (Fragment MTU) packets. So, the user should close the SOCKET, set FMTU as Sn_MSSR and retry the communication with the OPEN command.

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

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

0xFE4012	0xFE4013
0x05	0xB4

Sn_PROTO (SOCKET *n* Protocol Number Register)[R/W][0xFE4014 + 0x100*n*][0x00]

It is a 1 byte register that sets the protocol number field of the IP header at the IP layer. It is valid only in IPRAW mode, and ignored in other modes. Sn_PROTO is set before OPEN command. When SOCKET *n* is opened in IPRAW mode, it transmits and receives the data of the protocol number set in Sn_PROTO. Sn_PROTO can be assigned in the range of 0x00 ~ 0xFF, but W7100 does not support TCP(0x06) and UDP(0x11) protocol number

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

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

Sn_TOS (SOCKET *n* TOS Register)[R/W][0xFE4015 + 0x100*n*][0x00]

It sets the TOS(Type of Service) field of the IP header at the IP layer. It should be set before the OPEN command. Refer to <http://www.iana.org/assignments/ip-parameters>.

Sn_TTL (SOCKET *n* TTL Register)[R/W][0xFE4016 + 0x100*n*][0x80]

It sets the TTL(Time To Live) field of the IP header at the IP layer. It should be set before the OPEN command. Refer to <http://www.iana.org/assignments/ip-parameters>.

Sn_RXMEM_SIZE (SOCKET *n* Receive Memory Size Register)[R/W][0xFE401E + 0x100*n*][0x02]

It configures the internal RX Memory size of each SOCKET. RX Memory size of each SOCKET is configurable in the size of 1, 2, 4, 8Kbytes. 2Kbytes is assigned when reset. Sn_RXMEM_SIZE_{SUM}(sum of Sn_RXMEM_SIZE) of each SOCKET should be 16KB.

Ex1) SOCKET 0 : 8KB, SOCKET 1 : 2KB

0xFE401E	0xFE411E
0x08	0x02

Ex2) SOCKET 2 : 1KB, SOCKET 3 : 1KB

0xFE421E	0xFE431E
0x01	0x01

Ex3) SOCKET 4 : 1KB, SOCKET 5 : 1KB

0xFE441E	0xFE451E
0x01	0x01

Ex4) SOCKET 6 : 1KB, SOCKET 7 : 1KB

0xFE461E	0xFE471E
0x01	0x01

As shown above ex1) ~ ex4), total size of each SOCKET's RX memory ($S_n_RXMEM_SIZE_{SUM}$) is 16Kbytes.

$S_n_TXMEM_SIZE$ (SOCKET n Transmit Memory Size Register)[R/W][0xFE401F + 0x100n][0x02]

It configures the internal TX Memory size of each SOCKET. TX Memory size of each SOCKET is configurable in the size of 1, 2, 4, 8Kbytes. 2Kbytes is assigned when reset. $S_n_TXMEM_SIZE_{SUM}$ (summation of $S_n_TXMEM_SIZE$) of each SOCKET should be 16KB.

Ex5) SOCKET 0 : 4KB, SOCKET 1 : 1KB

0xFE401F	0xFE411F
0x04	0x01

Ex6) SOCKET 2 : 2KB, SOCKET 3 : 1KB

0xFE421F	0xFE431F
0x02	0x01

Ex7) SOCKET 4 : 2KB, SOCKET 5 : 2KB

0xFE441F	0xFE451F
0x02	0x02

Ex8) SOCKET 6 : 2KB, SOCKET 7 : 2KB

0xFE461F	0xFE471F
0x02	0x02

As shown above ex5) ~ ex8), total size of each SOCKET's TX memory (Sn_TXMEM_SIZE_{SUM}) is 16Kbytes.

Sn_TX_FSR (SOCKET *n* TX Free Size Register)[R] [(0xFE4020 + 0x100*n*) - (0xFE4021 + 100*n*)] [0x0000]

It notifies the available size of the internal TX memory (the byte size of transmittable data) of SOCKET *n*. The host can't write data as a size bigger than Sn_TX_FSR. Therefore, be sure to check Sn_TX_FSR before transmitting data, and if your data size is smaller than or the same as Sn_TX_FSR, transmit the data with SEND or SEND_MAC command after copying the data.

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

Ex) In case of 2048(0x8000) in S0_TX_FSR0

0xFE4020	0xFE4021
0x08	0x00

Sn_TX_RD (SOCKET *n* TX Read Pointer Register)[R] [(0xFE4022 + 0x100*n*) - (0xFE4023 + 0x100*n*)] [0x0000]

This register shows the address of the last transmission finishing in the TX memory. With the SEND command of SOCKET *n* Command Register, it transmits data from the current Sn_TX_RD to the Sn_TX_WR and automatically updates after transmission is finished. Therefore, after transmission is finished, Sn_TX_RD and Sn_TX_WR will have the same value. When reading this register, the user should read the upper bytes (0xFE4022, 0xFE4122, 0xFE4222, 0xFE4322, 0xFE4422, 0xFE4522, 0xFE4622, 0xFE4722) first and lower bytes (0xFE4023, 0xFE4123, 0xFE4223, 0xFE4323, 0xFE4423, 0xFE4523, 0xFE4623, 0xFE4723) later to get the correct value.

Sn_TX_WR (SOCKET *n* TX Write Pointer Register)[R/W] [(0xFE4024 + 0x100*n*) - (0xFE4025 + 0x100*n*)] [0x0000]

This register offers the location information of where the transmission data should be written. When reading this register, the user should read the upper bytes (0xFE4024, 0xFE4124, 0xFE4224, 0xFE4324, 0xFE4424, 0xFE4524, 0xFE4624, 0xFE4724) first and the lower bytes (0xFE4025, 0xFE4125, 0xFE4225, 0xFE4325, 0xFE4425, 0xFE4525, 0xFE4625, 0xFE4725) later to get the correct value.

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

0xFE4024	0xFE4025
0x08	0x00

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

1. SOCKET n TX Base Address (SBUFBASEADDRESS(n)) and SOCKET n TX Mask Address (SMASK(n)) are calculated on Sn_TXMEM_SIZE(n) value. Refer to the Pseudo code of the Initialization if detail is needed.
2. The bitwise-AND operation of two values and Sn_TX_WR and SMASK(n) gives the result of the offset address (dst_mask) in TX memory range of the SOCKET.
3. Two values dst_mask and SBUFBASEADDRESS(n) are added together to give the result of the physical address (dst_ptr).

Now, write the transmission data to dst_ptr as large as the user wants. (* There may be a case where it exceeds the TX memory of the 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 SBUFBASEADDRESS(n). Next, write the rest of the transmission data.)

After that, be sure to increase the Sn_TX_WR value by the size of writing data. Finally, give the SEND command to Sn_CR (SOCKET n Command Register). Refer to the pseudo code of the transmission part on TCP Server mode if the detail is needed.

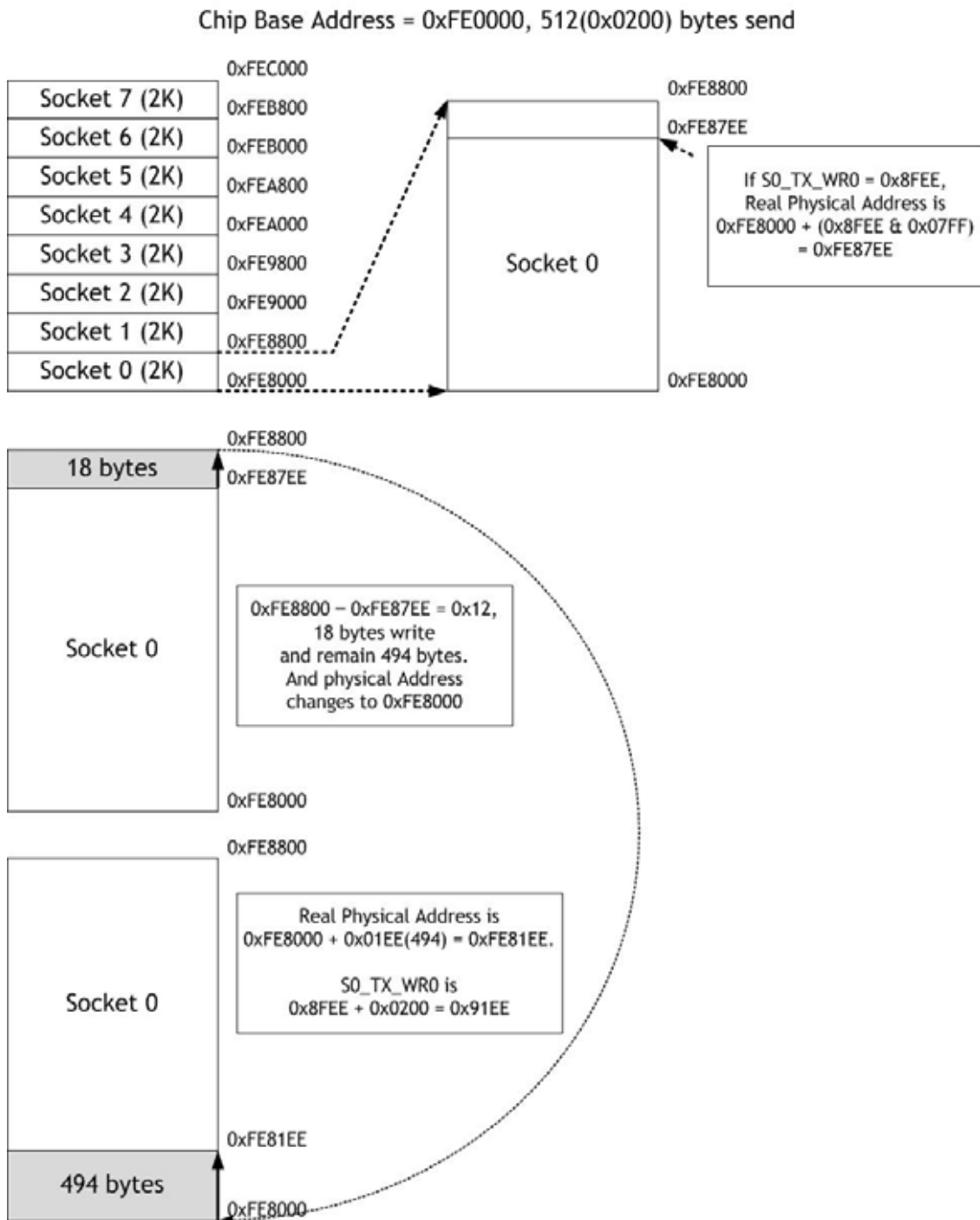


Figure 8.3 Calculate Physical Address

Sn_RX_RSR (SOCKET *n* RX Received Size Register)[R] [(0xFE4026 + 0x100*n*) - (0xFE4027 + 0x100*n*)] [0x0000]

It informs the user of the byte size of the received data in Internal RX Memory of SOCKET *n*. As this value is internally calculated with the values of Sn_RX_RD and Sn_RX_WR, it is automatically changed by RECV command of SOCKET *n* Command Register(Sn_CR) and receives data from the remote peer. When reading this register, the user should read the

upper byte(0xFE4026, 0xFE4126, 0xFE4226, 0xFE4326, 0xFE4426, 0xFE4526, 0xFE4626, 0xFE4726) first and lower byte(0xFE4027, 0xFE4127, 0xFE4227, 0xFE4327, 0xFE4427, 0xFE4527, 0xFE4627, 0xFE4727) later to get the correct value.

Ex) In case of 2048(0x0800) in S0_RX_RSR0,

0xFE4026	0xFE4027
0x08	0x00

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

Sn_RX_RD (SOCKET *n* Read Pointer Register)[R/W] [(0xFE4028 + 0x100n) - (0xFE4029 + 0x100n)][0x0000]

This register offers the location information to read the receiving data. When reading this register, user should read the upper byte (0xFE4028, 0xFE4128, 0xFE4228, 0xFE4328, 0xFE4428, 0xFE4528, 0xFE4628, 0xFE4728) first and lower byte (0xFE4029, 0xFE4129, 0xFE4229, 0xFE4329, 0xFE4429, 0xFE4529, 0xFE4629, 0xFE4729) later to get the correct value.

Ex) In case of 2048(0x0800) in S0_RX_RD,

0x0428	0x0429
0x08	0x00

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

1. SOCKET *n* RX Base Address (RBUFBASEADDRESS(*n*)) and SOCKET *n* RX Mask Address (RMASK(*n*)) are calculated on Sn_RXMEM_SIZE(*n*) value.
2. The bitwise-AND operation of two values, Sn_RX_RD and RMASK(*n*) gives the result of the offset address (src_mask), in the RX memory range of the SOCKET.
3. Two values src_mask and RBUFBASEADDRESS(*n*) are added together to give the result of the physical address(src_ptr).

Now, read the receiving data from src_ptr as large as the user wants. (* There may be a case where 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 RBUFBASEADDRESS(*n*). Next, read the rest of the receiving data.)

After that, be sure to increase the Sn_RX_RD value by the size of the 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 pseudo code of the receiving part on TCP Server mode if the detail is needed.

Sn_RX_WR (SOCKET *n* RX Write Pointer Register)[R/W] [(0xFE402A + 0x100n) - (0xFE402B + 0x100n)][0x0000]

This register offers the location information to write the receive data. When reading this register, the user should read upper bytes (0xFE402A, 0xFE412A, 0xFE422A, 0xFE432A, 0xFE442A, 0xFE452A, 0xFE462A, 0xFE472A) first and lower bytes (0xFE402B, 0xFE412B, 0xFE422B, 0xFE432B, 0xFE442B, 0xFE452B, 0xFE462B, 0xFE472B) later to get the correct value.

Ex) In case of 2048(0x0800) in S0_RX_WR,

0xFE402A	0xFE402B
0x08	0x00

Sn_IMR (SOCKET *n* Interrupt Mask Register)[R/W][0xFE402C + 0x100n][0x00]

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

7	6	5	4	3	2	1	0
PRECV	PFAIL	PNEXT	SEND_OK	TIMEOUT	RECV	DISCON	CON

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

Sn_FRAG (SOCKET *n* Fragment Register)[R/W][(0xFE402D + 0x100n) - (0xFE402E + 0x100n)][0x4000]

It sets the Fragment field of the IP header at the IP layer. W7100 does not support the packet fragment at the IP layer. Even though Sn_FRAG is configured, IP data is not fragmented, and not recommended either. It should be configured before performing OPEN command.

Ex) Sn_FRAG0 = 0x4000 (Don't Fragment)

0xFE402D

0xFE402E

0x40	0x00
------	------

9 Functional Description

Since the W7100 internally contains the 8051 compatible **MCU** and TCP/IP core, it can run standalone without other devices to Ethernet application. In this section, both the initialization of the W7100 and the communication method for each protocol (TCP, UDP, IPRAW and MACRAW) based on Pseudo code will be introduced.

9.1 Initialization

The initialization of W7100 has three steps which setup the 8051 MCU, the network information and the internal TX/RX memory.

● STEP 1 : Initializes MCU

1. Interrupt setting

Set the enable / disable state of interrupt such as the general 8051. Detail information of the setting refers to the section 3 'Interrupt'.

2. Memory Access timing setting

The memory access timing can be set by using two registers which are CKCON (0x8E) and WTST (0x92) registers. The CKCON (0x8E) can control the data memory access timing and the WTST (0x92) can control the code memory access timing. Both two registers can set their value from 0 to 7. But in the W7100, CKCON can set the value 1~7 and WTST can set the value 4~7 only. The other values of both registers are not used. If the user sets the value to an unused value, the W7100 cannot run properly. Detail information can be found in the section 2.4 'SFR definition'.

Ex) Setting: interrupt disabled, 2 clocks access time with data memory, 7 clocks access time with code memory.

```
EA = 0;           // Disable all interrupts
CKCON = 0x01;     // Set data memory access time
WTST = 0x06;      // Set code memory access time
```

3. Baud rate, register and interrupt setting for serial communication

1) For the serial communication, related registers of W7100 should be set.

The registers of W7100 for serial communication are TMOD, PCON and SCON as below.

① TMOD(89H): Decide the timer/counter mode for serial communication.

GATE	C/T	M ₁	M ₀	GATE	C/T	M ₁	M ₀
------	-----	----------------	----------------	------	-----	----------------	----------------

Table 12.1 Timer / Counter Mode

M ₁	M ₀	Mode
0	0	0
0	1	1
1	0	2
1	1	3

- ② PCON(87H): Decide the SMOD bit which is the control flag of the serial transmission rate.

SMOD	-	-	-	-	-	-	-
------	---	---	---	---	---	---	---

Table 12.2 Baud rate

Mode	SMOD = '0'	SMOD = '1'
1, 3	A half of Overflow of the Timer/Counter 1	Overflow of the Timer/Counter 1
2	A quarter of the XTAL	A half of the XTAL

- ③ SCON(98H): For control and observe the UART.

SM ₀	SM ₁	SM ₂	REN	TB ₈	RB ₈	TI	RI
-----------------	-----------------	-----------------	-----	-----------------	-----------------	----	----

Table 12.3 Mode of UART

SM ₀	SM ₁	Mode
0	0	0
0	1	1
1	0	2
1	1	3

SM₂: Used in Mode2, 3. Assume that this bit set to 1, if the 9th bit of received data bit is '1', receive the data. Or the bit is '0' ignore the data.

REN: Receive enable bit ('1'; Receive enable).

TB₈: In the mode2, 3, 8th bit of transmitted data.

RB₈: In the mode2, 3, 8th bit of received data.

TI: Transmission complete interrupt flag.

RI: Reception complete interrupt flag.

- 2) Interrupt state should be set when initializing the serial communication.

Since the serial communication uses interrupt, user must disable the related interrupts when initializing the serial communication.

- 3) The baud rate should be set to the value which the user will use. Baud rate value for the

timer of W7100 refers to the section 6.6 'Examples of Baud Rate Setting'.

The calculation of baud rate for the timer is as below

① Calculation formula of timer1

$$TH1 = 256 - ((K * 88.4736\text{MHz}) / (384 * \text{baud rate}))$$

K = '1' at SMOD = '0', K = '2' at SMOD = '1'

② Calculation formula of timer2

$$(RCAP2H, RCAP2L) = 65536 - (88.4736\text{MHz} / (32 * \text{baud rate}))$$

Ex) Using timer mode2, SMOD = 1, Clock speed = 88.4736MHz, Baud rate = 115200.

```
ET1= 0;           // Timer1 INT disable
TMOD = 0x20;      // TIMER MODE2
PCON |= 0x80;     // SMOD = 1
TH1 = 0xFC;       // x2 115200(SMOD = 1) at 88.4736MHz
TR1 = 1;          // Start the TIMER1
SCON = 0x50;      // Serial MODE1, REN = 1, TI = 0, RI = 0
ES = 0;           // Serial interrupt disable
RI = 0;           // Receive interrupt disable
TI = 0;           // Transmit interrupt disable
```

● STEP 2 : Setting Network Information

1. Basic network information setting for communication:

It must be set the basic network information.

① SHAR(Source Hardware Address Register)

It is prescribed that the source hardware addresses, which is set by SHAR, use unique hardware addresses (Ethernet MAC address) in the Ethernet MAC layer. The IEEE manages the MAC address allocation. The manufacturer which produces the network device allocates the MAC address to product.

Details on MAC address allocation refer to the website as below.

<http://www.ieee.org/>, <http://standards.ieee.org/regauth/oui/index.shtml>

② GAR(Gateway Address Register)

③ SUBR(Subnet Mask Register)

④ SIPR(Source IP Address Register)

2. Set the retransmission time & count when the packet transmission fails.

To set the retransmission time, the registers should be set as below.

① RTR(Retry Time-value Register), In the RTR, '1' means '100us'.

② RCR(Retry Count Register)

● STEP 3 : Allocation Internal TX/RX Memory for SOCKET *n*

Each configurable maximum size of TX, RX memory is 16 Kbytes. User can freely set the memory size to 1KB, 2KB, 4KB, and 8KB within 16 Kbytes in 8 sockets. If user assigns the all 16Kbytes memory to established SOCKET, other SOCKETS have no usable memory to use to transmission and reception.

```

In case of, assign 2KB rx, tx memory per SOCKET
{
gS0_RX_BASE = 0xFE0000(Chip base address) + 0xFEC000(Internal RX buffer address); // Set
base address of RX memory for SOCKET 0
Sn_RXMEM_SIZE(ch) = (uint8 *) 2; // Assign 2K rx memory per SOCKET
gS0_RX_MASK = 2K - 1; // 0x07FF, for getting offset address within assigned SOCKET 0 RX
memory
gS1_RX_BASE = gS0_RX_BASE + (gS0_RX_MASK + 1);
gS1_RX_MASK = 2K - 1;
gS2_RX_BASE = gS1_RX_BASE + (gS1_RX_MASK + 1);
gS2_RX_MASK = 2K - 1;
gS3_RX_BASE = gS2_RX_BASE + (gS2_RX_MASK + 1);
gS3_RX_MASK = 2K - 1;
gS4_RX_BASE = gS3_RX_BASE + (gS3_RX_MASK + 1);
gS4_RX_MASK = 2K - 1;
gS5_RX_BASE = gS4_RX_BASE + (gS4_RX_MASK + 1);
gS5_RX_MASK = 2K - 1;
gS6_RX_BASE = gS5_RX_BASE + (gS5_RX_MASK + 1);
gS6_RX_MASK = 2K - 1;
gS7_RX_BASE = gS6_RX_BASE + (gS6_RX_MASK + 1);
gS7_RX_MASK = 2K - 1;
gS0_TX_BASE = 0xFE0000(Chip base address) + 0xFE8000(Internal TX buffer address); // Set
base address of TX memory for SOCKET 0
Sn_TXMEM_SIZE(ch) = (uint8 *) 2; // Assign 2K rx memory per SOCKET
gS0_TX_MASK = 2K - 1;
Same method, set gS1_TX_BASE, gS1_TX_MASK, gS2_TX_BASE, gS2_TX_MASK, gS3_TX_BASE,
gS3_TX_MASK, gS4_TX_BASE, gS4_TX_MASK, gS5_TX_BASE, gS5_TX_MASK, gS6_TX_BASE,
gS6_tx_MASK, gS7_TX_BASE, gS7_TX_MASK.
}
    
```

Sn_TXMEM_SIZE(ch) = 2K,
Chip base address = 0xFE0000

Socket 7	0xFEC000	gS7_TX_BASE = 0xFEB800 gS7_TX_MASK = 0x07FF
Socket 6	0xFEB800	gS6_TX_BASE = 0xFEB000 gS6_TX_MASK = 0x07FF
Socket 5	0xFEB000	gS5_TX_BASE = 0xFE800 gS5_TX_MASK = 0x07FF
Socket 4	0xFE800	gS4_TX_BASE = 0xFE400 gS4_TX_MASK = 0x07FF
Socket 3	0xFE400	gS3_TX_BASE = 0xFE9800 gS3_TX_MASK = 0x07FF
Socket 2	0xFE9800	gS2_TX_BASE = 0xFE9000 gS2_TX_MASK = 0x07FF
Socket 1	0xFE9000	gS1_TX_BASE = 0xFE8800 gS1_TX_MASK = 0x07FF
Socket 0	0xFE8800	gS0_TX_BASE = 0xFE8000 gS0_TX_MASK = 0x07FF

(a) Tx

Sn_RXMEM_SIZE(ch) = 2K,
Chip base address = 0xFE0000

Socket 7	0xFE800	gS7_RX_BASE = 0xFE7800 gS7_RX_MASK = 0x07FF
Socket 6	0xFE7800	gS6_RX_BASE = 0xFE7000 gS6_RX_MASK = 0x07FF
Socket 5	0xFE7000	gS5_RX_BASE = 0xFEE800 gS5_RX_MASK = 0x07FF
Socket 4	0xFEE800	gS4_RX_BASE = 0xFEE000 gS4_RX_MASK = 0x07FF
Socket 3	0xFEE000	gS3_RX_BASE = 0xFED800 gS3_RX_MASK = 0x07FF
Socket 2	0xFED800	gS2_RX_BASE = 0xFED000 gS2_RX_MASK = 0x07FF
Socket 1	0xFED000	gS1_RX_BASE = 0xFEC800 gS1_RX_MASK = 0x07FF
Socket 0	0xFEC800	gS0_RX_BASE = 0xFEC000 gS0_RX_MASK = 0x07FF

(b) Rx

Figure 9.1 Allocation Internal TX/RX memory of SOCKET *n*

If the three steps of W7100 initialization process are finished, the W7100 can perform data communication through Ethernet. From this point, the W7100 can transmit the ping-reply of the request packet which is received from network.

9.2 Data Communication

After the initialization process, W7100 can transmit and receive the data with others by 'open' the SOCKET of TCP, UDP, IPRAW and MACRAW mode. The W7100 supports the independently and simultaneously usable 8 SOCKETS. In this section, the communication method for each mode will be introduced.

9.2.1 TCP

The TCP is a connection-oriented protocol. The TCP make the connection SOCKET by using its own IP address, port number and destination IP address, port number. Then transmits and receives the data by using this SOCKET.

Methods of making the connection to SOCKET are "TCP SERVER" and "TCP CLIENT". It is divided by transmitting the connect-request (SYN packet).

The "TCP SERVER" listens to the connect-request from the "TCP CLIENT", and makes connection SOCKET by accepting the transmitted connect-request (Passive-open).

The "TCP CLIENT" transmits the connect-request first to "TCP SERVER" to make the connection (Active-open).

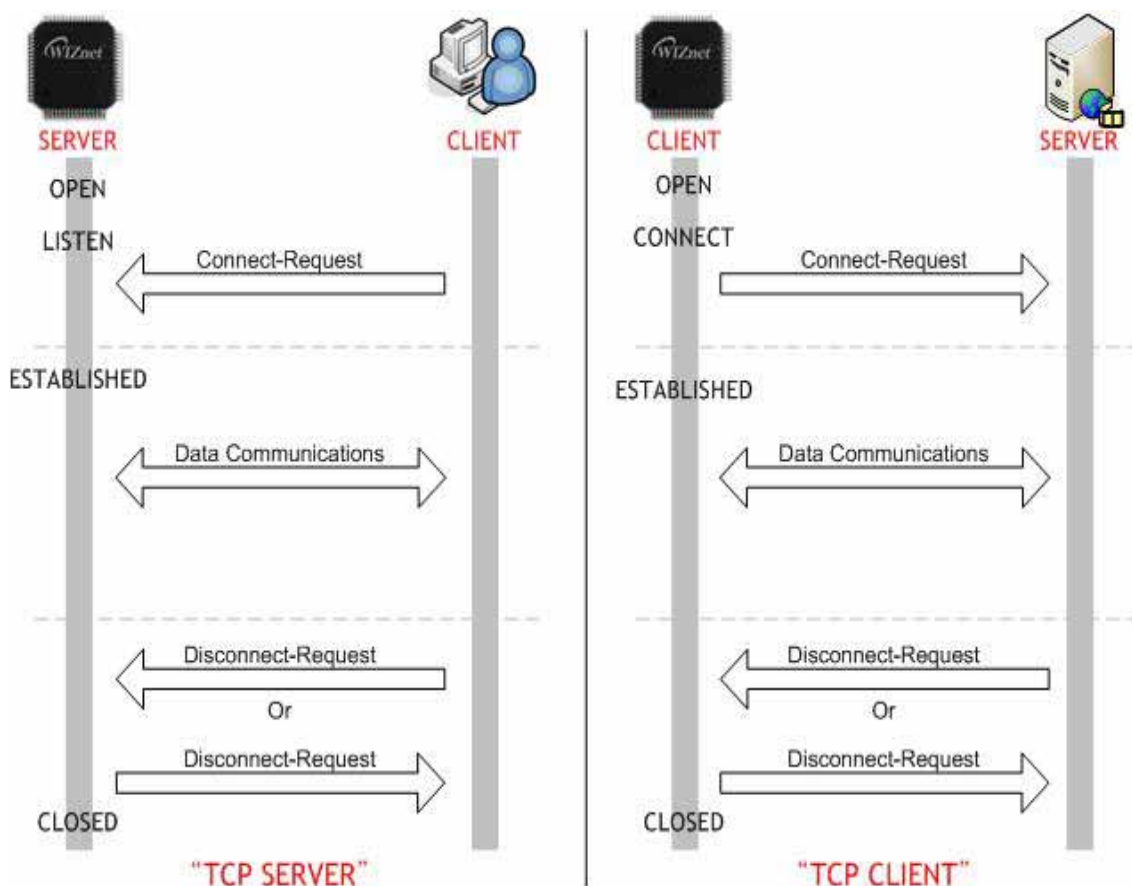


Figure 9.2 TCP SERVER & TCP CLIENT

9.2.1.1 TCP SERVER

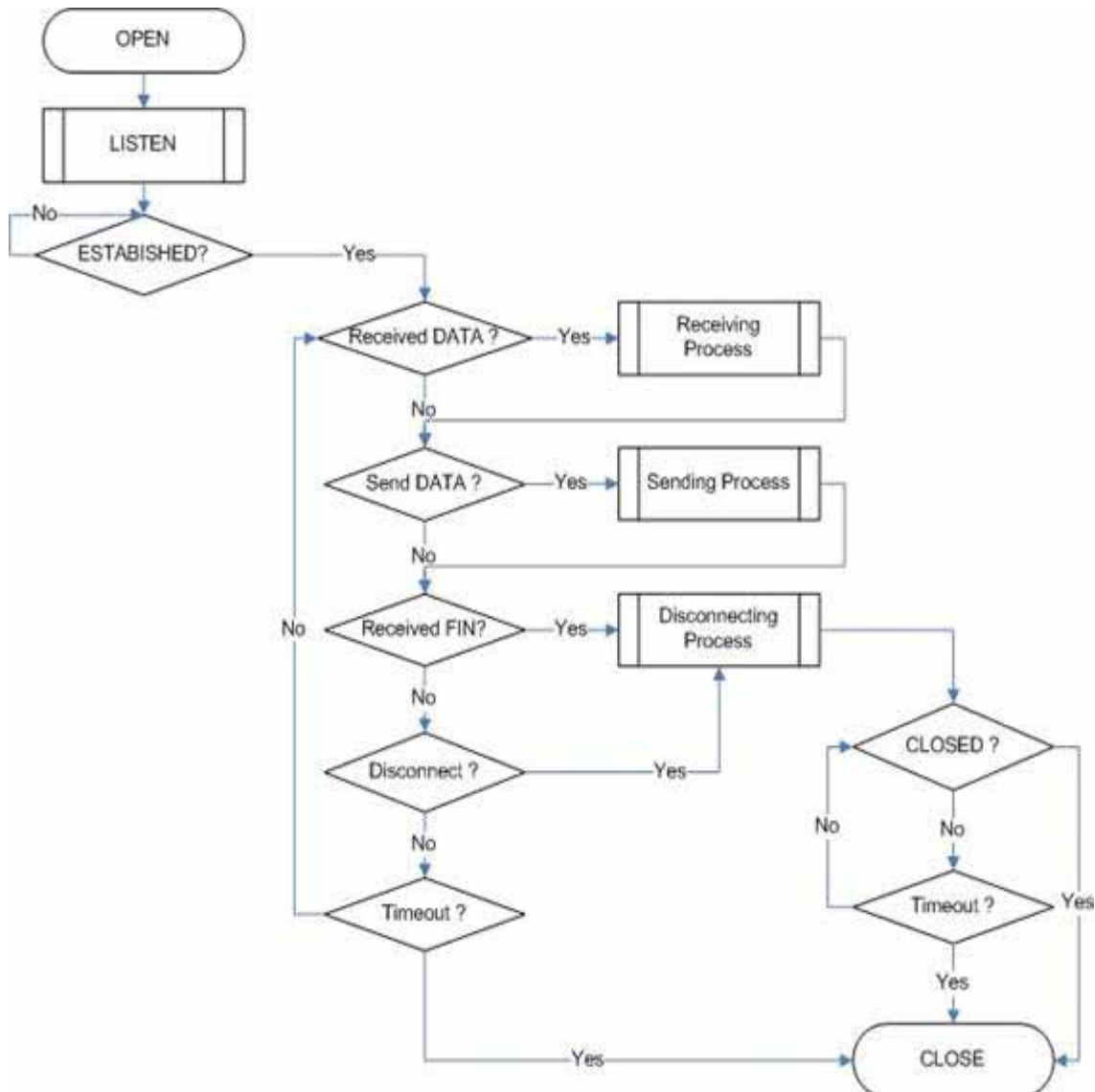


Figure 9.3 “TCP SERVER” Operation Flow

■ SOCKET Initialization

SOCKET initialization is required for TCP data communication. The initialization is opening the SOCKET. The SOCKET opening process selects one SOCKET from 8 SOCKETS of the W7100, and sets the protocol mode (Sn_MR(P3:P0)) and Sn_PORT0 which is source port number (Listen port number in “TCP SERVER”) in the selected SOCKET, and then executes OPEN command. After the OPEN command, if the status of Sn_SR is changed to SOCK_INIT, the SOCKET initialization process is completed.

The SOCKET initialization process is identically applied in “TCP SEVER” and “TCP CLIENT”. The Initialization process of SOCKET *n* in TCP mode is shown below.

```
{
START:
    Sn_MR = 0x0001;           // sets TCP mode
    Sn_PORT0 = source_port;   // sets source port number
    Sn_CR = OPEN;            // sets OPEN command
    /* wait until Sn_SR is changed to SOCK_INIT */
    if (Sn_SR != SOCK_INIT) Sn_CR = CLOSE; goto START;
}
```

■ LISTEN

Run as “TCP SERVER” by LISTEN command.

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

■ ESTABLISHMENT

When the status of Sn_SR is SOCK_LISTEN, if it receives a SYN packet, the status of Sn_SR is changed to SOCK_SYNRCV and transmits the SYN/ACK packet. After that, the SOCKET *n* makes a connection. After it makes the connection of SOCKET *n*, it enables the data communication. There are two methods to confirm the connection of SOCKET *n*.

First method :

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

Second method :

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

■ ESTABLISHMENT : Check received data

Confirm the reception of the TCP data.

First method :

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

Second Method :

```
{
    if (Sn_RX_RSR0 != 0x00000000) goto Receiving Process stage;
}
```

The First method: set the Sn_IR(RECV) to '1' whenever you receive a DATA packet. If the host receives the next DATA packet without setting the Sn_IR(RECV) as '1' in the prior DATA packet, it cannot recognize the Sn_IR(RECV) of the next DATA packet. This is due to the prior Sn_IR(RECV) and next Sn_IR(RECV) being overlapped. So this method is not recommended if the host cannot perfectly process the DATA packets of each Sn_IR(RECV).

■ ESTABLISHMENT : Receiving process

In this process, it processes the TCP data which was received in the Internal RX memory. At the TCP mode, the W7100 cannot receive the data if the size of received data is larger than the RX memory free size of SOCKET *n*. If the prior stated condition is happened, the W7100 holds on to the connection (pauses), and waits until the RX memory's free size is larger than the size of the received data.

Since the wizmemcpy function, using Receive / Send process, is included in boot of W7100 for fast processing, there is no more redefinition to use. If user does not use the wizmemcpy function, user can use other defined memory copy function. Since the W7100 internally has data memory and TCIPCore internal memory, user should classify it by address. So user must pad '0xFE' to the top level address of TCIPCore internal memory with, or DPX0 register set to '0xFE' when memory copying from TCIPCore memory to data memory. More detail about the wizmemcpy please refers to the 'W7100 Driver Guide'.

```
{
    /* first, get the received size */
    len = Sn_RX_RSR0;    // len is received size
    /* calculate offset address */
    src_mask = Sn_RX_RD & gSn_RX_MASK;    // src_mask is offset address
    /* calculate start address(physical address) */
    src_ptr = gSn_RX_BASE + src_mask;    // src_ptr is physical start address
}
```

```

/* if overflow SOCKET RX memory */
If((src_mask + len) > (gSn_RX_MASK + 1))
{
    /* copy upper_size bytes of get_start_address to destination_address */
    upper_size = (gSn_RX_MASK + 1) - src_mask;
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), upper_size);
    /* update destination_address */
    destination_address += upper_size;
    /* copy left_size bytes of gSn_RX_BASE to destination_address */
    left_size = len - upper_size;
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), left_size);
}
else
{
    /* copy len bytes of src_ptr to destination_address */
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), len);
}

/* increase Sn_RX_RD as length of len */
Sn_RX_RD += len;

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

```

■ ESTABLISHMENT : Check send data / Send process

The size of the transmit data cannot be larger than assigned internal TX memory of SOCKET *n*. If the size of transmit data is larger than configured MSS, it is divided by size of MSS and transmits.

To transmit the next data, user must check the completion of prior SEND command. An error may occur if the SEND command executes before completion of prior SEND command. The larger the data size, the more time to complete the SEND command. So the user should properly divide the data to transmit.

At the send process, user must pad '0xFE' to top-level address of TCIPCore internal memory as in the receive process.

```

{
    /* first, get the free TX memory size */
    FREESIZE:
    freesize = Sn_TX_FSR0;
}

```

```

if (freesize < len) goto FREESIZE;    // len is send size
/* calculate offset address */
dst_mask= Sn_TX_WRO & gSn_TX_MASK;    // dst_mask is offset address
/* calculate start address(physical address) */
dst_ptr = gSn_TX_BASE + dst_mask;    // dst_ptr is physical start address
/* if overflow SOCKET TX memory */
if ( (dst_mask + len) > (gSn_TX_MASK + 1) )
{
    /* copy upper_size bytes of source_addr to dst_ptr */
    upper_size = (gSn_TX_MASK + 1) - dst_mask;
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + dst_ptr), upper_size);
    /* update source_addr */
    source_addr += upper_size;
    /* copy left_size bytes of source_addr to gSn_TX_BASE */
    left_size = len - upper_size;
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + gSn_TX_BASE), left_size);
}
else
{
    /* copy len bytes of source_addr to dst_ptr */
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + dst_ptr), len);
}
/* increase Sn_TX_WR as length of len */
Sn_TX_WRO += send_size;
/* set SEND command */
Sn_CR = SEND;
}

```

■ ESTABLISHMENT : Check disconnect-request(FIN packet)

Check if the Disconnect-request(FIN packet) has been received. User can confirm the reception of FIN packet as below.

```

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

```

Second method :

```
{
    if (Sn_SR == SOCK_CLOSE_WAIT) goto CLOSED stage;
}
```

■ ESTABLISHMENT : Check disconnect / disconnecting process

When the user does not need data communication with others, or receives a FIN packet, disconnect the connection SOCKET.

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

■ ESTABLISHMENT : Check closed

Confirm that the SOCKET *n* is disconnected or closed by DISCON or close command.

First method :

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

Second method :

```
{
    if (Sn_SR == SOCK_CLOSED) goto CLOSED stage;
}
```

■ ESTABLISHMENT : Timeout

The timeout can occur by Connect-request(SYN packet) or its response(SYN/ACK packet), the DATA packet or its response(DATA/ACK packet), the Disconnect-request(FIN packet) or its response(FIN/ACK packet) and transmission all TCP packet. If it cannot transmit the above packets within 'timeout' which is configured at RTR and RCR, the TCP final timeout(TCP_{T0}) occurs and the state of Sn_SR is set to SOCK_CLOSED. Confirming method of the TCP_{T0} is as below:

First method :

```
{
```

```

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

```

Second method :

```

{
    if (Sn_SR == SOCK_CLOSED) goto CLOSED stage;
}

```

■ SOCKET close

It can be used to close the SOCKET *n*, which disconnected by disconnect-process, or closed by TCP_{TO} or closed by host's need without disconnect-process.

```

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

```

9.2.1.2 TCP CLIENT

It is same as TCP server except 'CONNECT' state. User can refer to the "9.2.1.1 TCP SERVER".

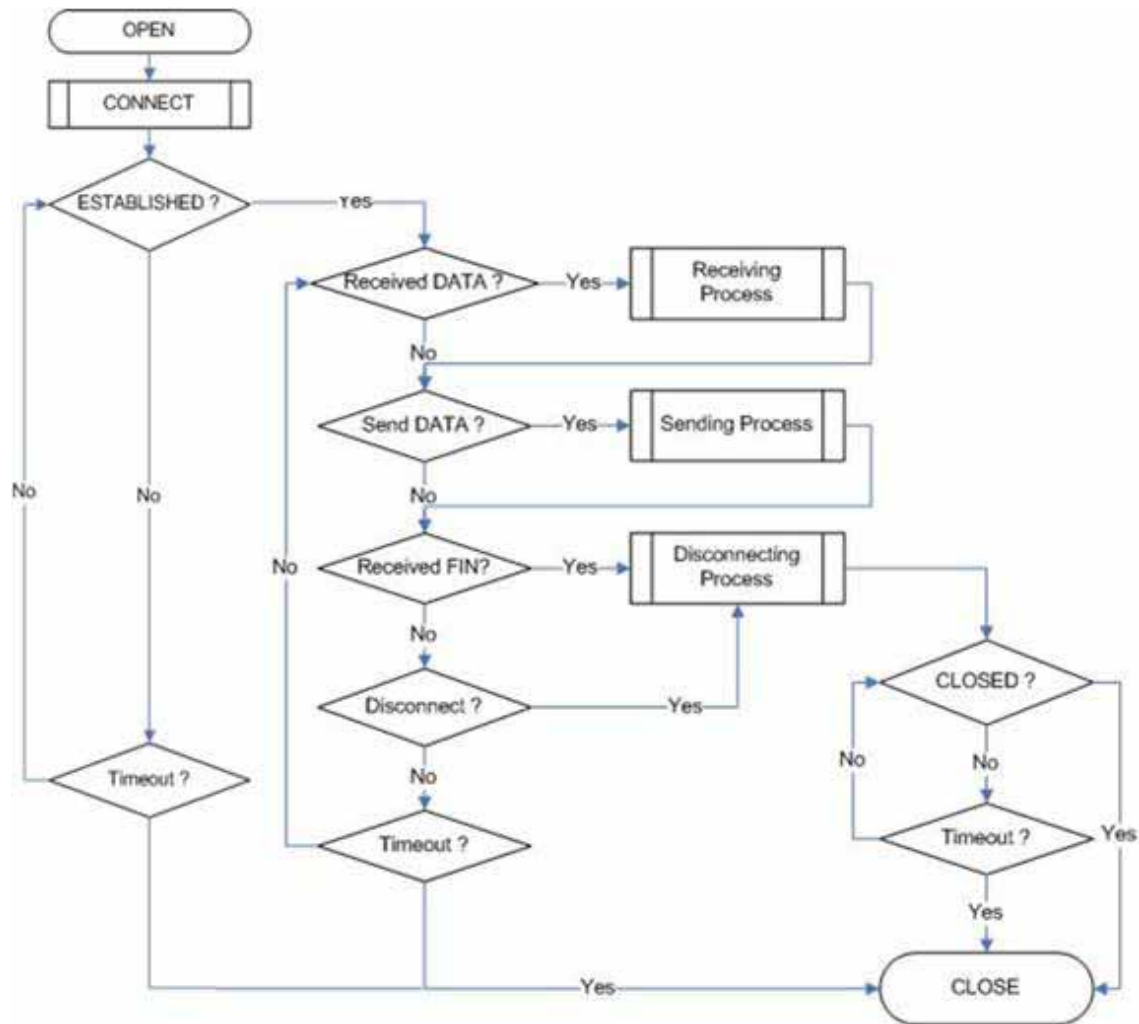


Figure 9.5 "TCP CLIENT" Operation Flow

■ CONNECT

Transmit the connect-request(SYN packet) to "TCP SERVER". It may occurs the timeout such as ARP_{TO} , TCP_{TO} when make the "connection SOCKET" with "TCP SERVER"

```

{
  Sn_DIPR0 = server_ip;          /* set TCP SERVER IP address*/
  Sn_DPORT0 = server_port;       /* set TCP SERVER listen port number*/
  Sn_CR = CONNECT;               /* set CONNECT command */
}
  
```

9.2.2 UDP

The UDP is a Connection-less protocol. It communicates without “connection SOCKET”. The TCP protocol guarantees reliable data communication, but the UDP protocol use datagram communication which has no guarantees of data communication. Because the UDP do not use “connection SOCKET”, it can communicate with many other devices with the known host IP address and port number. This is a great advantage; communication with many others by using just one SOCKET, but also it has many problems such as loss of transmitted data, unwanted data which received from others etc. In the UDP, to avoid these problems and guarantee the reliability, the host retransmits damaged data or ignores the unwanted data which is received from others. The UDP protocol supports unicast, broadcast, and multicast communication. It follows the below communication flow.

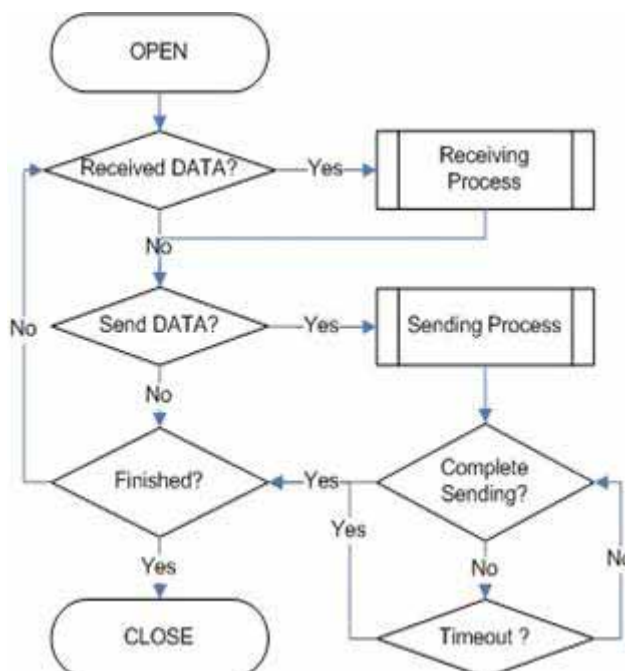


Figure 9.6 UDP Operation Flow

9.2.2.1 Unicast & Broadcast

The unicast is one method of UDP communication. It transmits data to one destination at one time. On the other hand, the broadcast communication transmits data to all receivable destinations by using ‘broadcast IP address (255.255.255.255)’. For example, suppose that the user transmits data to destination A, B and C. The unicast communication transmits each destination A, B and C at each time. At this time, the ARP_{TO} can also occur when the user gets the destination hardware address of destinations A, B and C. User cannot transmit data to destinations which have ARP_{TO}.

The broadcast communication can simultaneously transmit data to destination A, B and C at

one time by using “255.255.255.255” IP address. At this time, there is no need to get the destination hardware address about destination A, B and C, and also ARP_{TO} is not occurred.

■ SOCKET Initialization

For the UDP data communication, SOCKET initialization is needed. It is opening the SOCKET. The SOCKET open process is as follows. At first choose the one SOCKET among the 8 SOCKETS of W7100, then set the protocol mode(Sn_MR(P3:P0)) of the chosen SOCKET and set the source port number Sn_PORT0 for communication. Finally execute the OPEN command. After the OPEN command, the state of Sn_SR is changed to SOCK_UDP. Then the SOCKET initialization is complete.

```
{
START:
    Sn_MR = 0x02;                /* sets UDP mode */
    Sn_PORT0 = source_port;      /* sets source port number */
    Sn_CR = OPEN;                /* sets OPEN command */
    /* wait until Sn_SR is changed to SOCK_UDP */
    if (Sn_SR != SOCK_UDP) Sn_CR = CLOSE; goto START;
}
```

■ Check received data

Check the reception of UDP data from destination. User can also check for received data via TCP communication. It is strongly recommended that the TCP method is used because of the same reason ing TCP. Please refer to the “9.2.1.1 TCP SERVER”.

First method :

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

Second Method :

```
{
    if (Sn_RX_RS0 != 0x00000000) goto Receiving Process stage;
}
```

■ Receiving process

Process the received UDP data in Internal RX memory. The structure of received UDP data is as below.

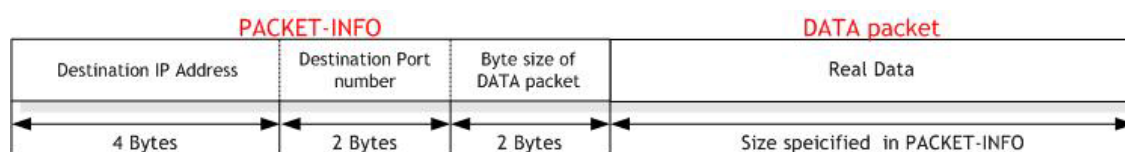


Figure 9.7 The received UDP data format

The received UDP data consists of 8bytes PACKET-INFO, and DATA packet. The PACKET-INFO contains transmitter's information (IP address, Port number) and the length of DATA packet. The UDP can receive UDP data from many others. User can classify the transmitter by transmitter's information of PACKET-INFO. It also receives broadcast SOCKET by using "255.255.255.255" IP address. So the host should ignore unwanted reception by analysis of transmitter's information.

If the DATA size of SOCKET n is larger than Internal RX memory free size, user cannot receive that DATA and also cannot receive fragmented DATA.

```
{
    /* first, get the received size */
    len = Sn_RX_RSR0;    // len is received size
    /* calculate offset address */
    src_mask = Sn_RX_RD & gSn_RX_MASK;    // src_mask is offset address
    /* calculate start address(physical address) */
    src_ptr = gSn_RX_BASE + src_mask;    // src_ptr is physical start address

    /* read head information (8 bytes) */
    header_size = 8;
    /* if overflow SOCKET RX memory */
    if ( (src_mask + header_size) > (gSn_RX_MASK + 1) )
    {
        /* copy upper_size bytes of src_ptr to header_addr */
        upper_size = (gSn_RX_MASK + 1) - src_mask;
        wizmemcpy((0xFE0000 + src_ptr), (0x000000 + 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;
        wizmemcpy((0xFE0000 + gSn_RX_BASE), (0x000000 + header_addr), left_size);
        /* update src_mask */
        src_mask = left_size;
    }
}
```

```

}
else
{
    /* copy header_size bytes of get_start_address to header_addr */
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + header_addr), header_size);
    /* update src_mask */
    src_mask += header_size;
}
/* update src_ptr */
src_ptr = gSn_RX_BASE + src_mask;
/* 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 ( (src_mask + len) > (gSn_RX_MASK + 1) )
{
    /* copy upper_size bytes of src_ptr to destination_addr */
    upper_size = (gSn_RX_MASK + 1) - src_mask;
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + 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;
    wizmemcpy((0xFE0000 + gSn_RX_BASE), (0x000000 + destination_addr), left_size);
}
else
{
    /* copy len bytes of src_ptr to destination_addr */
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_addr), get_size);
}
/* increase Sn_RX_RD as length of len + header_size */
Sn_RX_RD = Sn_RX_RD + len + header_size;
/* set RECV command */
Sn_CR = RECV;
}

```

■ Check send data / Sending process

The size of DATA which user wants to transmit cannot be larger than Internal TX memory. If it is larger than MTU, it is automatically divided by MTU unit and transmits.

The Sn_DIPR0 is set "255.255.255.255" when user wants to broadcast.

```
{
    /* first, get the free TX memory size */
FREESIZE:
    freesize = Sn_TX_FSR0;
    if (freesize < len) goto FREESIZE;    // len is send size

    /* 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_DIPR0 = remote_ip;
    Sn_DPORT0 = remote_port;

    /* calculate offset address */
    dst_mask = Sn_TX_WR0 & gSn_TX_MASK;    // dst_mask is offset address
    /* calculate start address(physical address) */
    dst_ptr = gSn_TX_BASE + dst_mask;    // dst_ptr is physical start address

    /* if overflow SOCKET TX memory */
    if ( (dst_mask + len) > (gSn_TX_MASK + 1) )
    {
        /* copy upper_size bytes of source_address to dst_ptr */
        upper_size = (gSn_TX_MASK + 1) - dst_mask;
        wizmemcpy((0x000000 + source_address), (0xFE0000 + dst_ptr), upper_size);
        /* update source_address */
        source_address += upper_size;
        /* copy left_size bytes of source_address to gSn_TX_BASE */
        left_size = send_size - upper_size;
        wizmemcpy((0x000000 + source_address), (0xFE0000 + gSn_TX_BASE), left_size);
    }
    else
    {
        /* copy len bytes of source_address to dst_ptr */
        wizmemcpy((0x000000 + source_address), (0xFE0000 + dst_ptr), len);
    }
}
```

```

/* increase Sn_TX_WR0 as length of len */
Sn_TX_WR0 += len;
/* set SEND command */
Sn_CR = SEND;
}

```

■ Check complete sending / Timeout

To transmit the next data, user must check that the prior SEND command is completed. The larger the data size, the more time to complete the SEND command. Therefore, the user must properly divide the data to transmit. The ARP_{TO} can occur when user transmits UDP data. If the ARP_{TO} is occurred, the UDP data transmission is failed.

First method :

```

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

```

Second method :

```

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

```

■ Check Finished / SOCKET close

If user doesn't need the communication any more, close the SOCKET *n*.

```

{
/* clear remained interrupts */
Sn_IR = 0x00FF;
IR(n) = '1';
}

```

```

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

```

9.2.2.2 Multicast

The broadcast communication communicates with many and unspecified others. But the multicast communication communicates with many but specified others who registered at multicast-group. Suppose that A, B and C are registered at specified multicast-group. If user transmits data to multicast-group (contains A), the B and C also receive the DATA for A. To use multicast communication, the destination list registers to multicast-group by using IGMP protocol. The multicast-group consists of 'Group hardware address', 'Group IP address' and 'Group port number'. User cannot change the 'Group hardware address' and 'Group IP address'. But the 'Group port number' can be changed to what user wants.

The 'Group hardware address' is selected at the assigned range (From "01:00:5e:00:00:00" to "01:00:5e:7f:ff:ff") and the 'Group IP address' is selected in D-class IP address (From "224.0.0.0" to "239.255.255.255", please refer to the website; <http://www.iana.org/assignments/multicast-addresses>). When selecting, the upper 23bit of 6bytes 'Group hardware address' and the 4bytes 'Group IP address' must be the same. For example, if the user selects the 'Group IP address' to "244.1.1.11", the 'Group hardware address' is selected to "01:00:5e:01:01:0b". Please refer to the "RFC1112" (<http://www.ietf.org/rfc.html>).

In the W7100, IGMP processing to register the multicast-group is internally (automatically) processed. When the user opens the SOCKET *n* with multicast mode, the "Join" message is internally transmitted. If the user closes it, the "Leave" message is internally transmitted. After the SOCKET opens, the "Report" message is periodically and internally transmitted when the user communicates.

The W7100 support IGMP version 1 and version 2 only. If user wants use an updated version, the host processes IGMP directly by using the IPRAW mode SOCKET.

■ SOCKET Initialization

Choose one SOCKET for multicast communication among 8 SOCKETS of W7100. Then set the Sn_DHAR0 to 'Multicast-group hardware address' and set the Sn_DIPRO to 'Multicast-group IP address'. Then set the Sn_PORT0 and Sn_DPORT0 to 'Multicast-group port number'. Then set the Sn_MR(P3:P0) to UDP and set the Sn_MR(MULTI) to '1'. Finally execute OPEN command. If the state of Sn_SR is changed to SOCK_UDP after the OPEN command, the SOCKET initialization is completed.

```

{
START:
    /* set Multicast-Group information */
    Sn_DHAR0 = 0x01;      /* set Multicast-Group H/W address(01:00:5e:01:01:0b) */
    Sn_DHAR1 = 0x00;
    Sn_DHAR2 = 0x5E;
    Sn_DHAR3 = 0x01;
    Sn_DHAR4 = 0x01;
    Sn_DHAR5 = 0x0B;
    Sn_DIPR0 = 211;      /* set Multicast-Group IP address(211.1.1.11) */
    Sn_DIPR1 = 1;
    Sn_DIPR2 = 1;
    Sn_DIRP3 = 11;
    Sn_DPORT0 = 0x0BB8; /* set Multicast-Group Port number(3000) */
    Sn_PORT0 = 0x0BB8;  /* set Source Port number(3000) */
    Sn_MR = 0x02 | 0x80; /* set UDP mode & Multicast on SOCKET n Mode Register */

    Sn_CR = OPEN;      /* set OPEN command */

    /* wait until Sn_SR is changed to SOCK_UDP */
    if (Sn_SR != SOCK_UDP) Sn_CR = CLOSE; goto START;
}

```

■ Check received data

Refer to the “9.2.2.1 Unicast & Broadcast”.

■ Receiving process

Refer to the “9.2.2.1 Unicast & Broadcast”.

■ Check send data / Sending Process

Since the user sets the information about multicast-group at SOCKET initialization, user does not need to set IP address and port number for destination any more. Therefore, copy the transmission data to internal TX memory and executes SEND command.

```

{
    /* first, get the free TX memory size */
FREESIZE:
    freesize = Sn_TX_FSR0;
    if (freesize < len) goto FREESIZE;    // len is send size
}

```

```

/* calculate offset address */
dst_mask = Sn_TX_WRO & gSn_TX_MASK;    // dst_mask is offset address
/* calculate start address(physical address) */
dst_ptr = gSn_TX_BASE + dst_mask;    // dst_ptr is physical start address
/* if overflow SOCKET TX memory */
if ( (dst_mask + len) > (gSn_TX_MASK + 1) )
{
    /* copy upper_size bytes of source_addr to dst_ptr */
    upper_size = (gSn_TX_MASK + 1) - dst_mask;
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + dst_ptr), upper_size);
    /* update source_addr */
    source_addr += upper_size;
    /* copy left_size bytes of source_addr to gSn_TX_BASE */
    left_size = len - upper_size;
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + gSn_TX_BASE), left_size);
}
else
{
    /* copy len bytes of source_addr to dst_ptr */
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + dst_ptr), len);
}
/* increase Sn_TX_WR as length of len */
Sn_TX_WRO += send_size;
/* set SEND command */
Sn_CR = SEND;
}

```

■ Check complete sending / Timeout

Since the host manages all protocol process for data communication, the timeout cannot occur.

```

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

```

■ Check finished / SOCKET close

Refer to the “9.2.2.1 Unicast & Broadcast”.

9.2.3 IPRAW

The IPRAW is data communication using TCP, UDP and IP layers which are the lower protocol layers. The IPRAW supports IP layer protocol such as ICMP (0x01) and IGMP (0x02) according to the protocol number. The 'ping' of ICMP or IGMP v1/v2 is already included in W7100 by hardware logic. But if the user needs, the host can directly process the IPRAW by opening the SOCKET *n* to IPRAW. In the case of using IPRAW mode, user must set the protocol number field of the IP header to what the user wants to use. The protocol number is defined by IANA. Refer to the web (<http://www.iana.org/assignments/protocol-numbers>). The protocol number must be configured to Sn_PROTO before 'SOCKET open'. In the IPRAW mode, the W7100 does not support TCP (0x06) or UDP (0x11) protocol number. The SOCKET communication of IPRAW mode only allows the communication of an assigned protocol number. The ICMP SOCKET cannot receive unassigned protocol data except assigned protocol data such as IGMP.

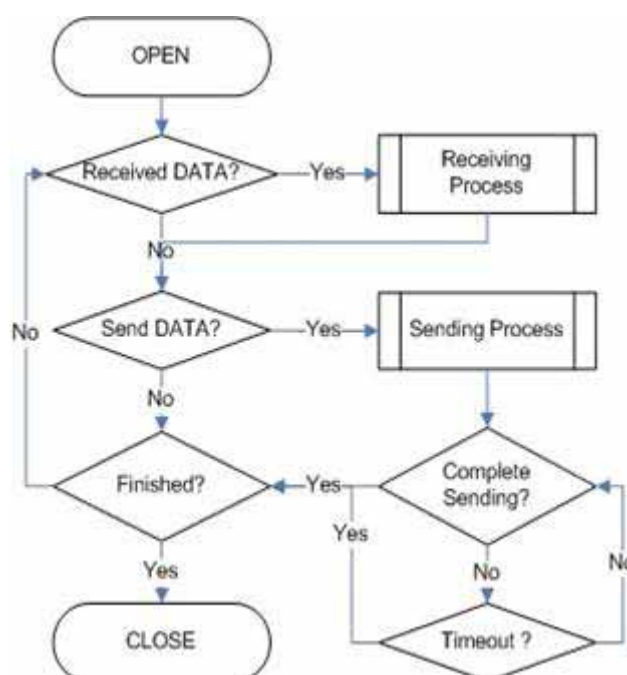


Figure 9.8 IPRAW Operation Flow

■ SOCKET Initialization

Select the SOCKET and set the protocol number. Then set the Sn_MR(P3:P0) to IPRAW mode and execute 'OPEN' command. If the Sn_SR is changed to SOCK_IPRAW after the 'OPEN' command, the SOCKET initialization is completed.

```

{
START:
    /* sets Protocol number */
    /* The protocol number is used in Protocol Field of IP Header. */

```

```

Sn_PROTO = protocol_num;
/* sets IP raw mode */
Sn_MR = 0x03;
/* sets OPEN command */
Sn_CR = OPEN;
/* wait until Sn_SR is changed to SOCK_IPRAW */
if (Sn_SR != SOCK_IPRAW) Sn_CR = CLOSE; goto START;
}

```

■ Check received data

Refer to the “9.2.2.1 Unicast & Broadcast”.

■ Receiving process

Process the IPRAW data which is received in internal RX memory. The structure of received IPRAW data is as below.

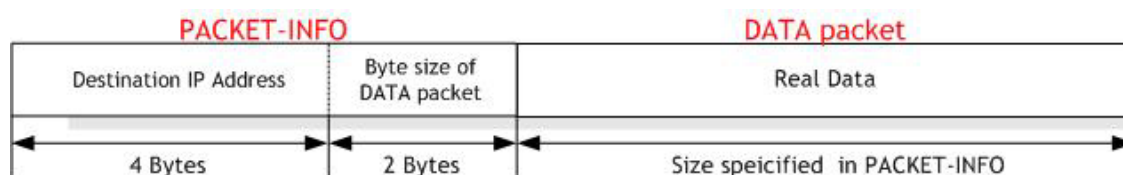


Figure 9.9 The received IPRAW data format

The IPRAW data consists of 6bytes PACKET-INFO and DATA packet. The PACKET-INFO contains information about the transmitter (IP address) and the length of the DATA-packet. The data reception of IPRAW is the same as UDP data reception except processing the port number of the transmitter in UDP PACKET-INFO. Refer to the “9.2.2.1 Unicast & Broadcast”.

If the transmitted DATA size is larger than RX memory free size of SOCKET n , user cannot receive that DATA and also cannot receive fragmented DATA.

■ Check send data / Sending process

The size of DATA which user wants to transmit cannot be larger than Internal TX memory and default MTU. The transmission of IPRAW data is the same as transmission of UDP data except setting ‘Destination port number’. Refer to the “9.2.2.1 Unicast & Broadcast”.

■ Complete sending / Timeout

Same as UDP, please refer to the “9.2.2 UDP”.

■ Check finished / SOCKET closed

Same as UDP, please refer to the “9.2.2 UDP”.

9.2.4 MACRAW

The MACRAW communication is based on Ethernet MAC, and it can flexibly use upper layer protocol to suit the host's needs.

The MACRAW mode can only be used with a SOCKET. If the user uses the SOCKET in MACRAW mode, not only can it use the SOCKET1~7 in the 'Hardwired TCP/IP stack', but it can also be used as a NIC (Network Interface Controller). Therefore, any SOCKET1~7 can be used with 'Software TCP/IP stack'. Since the W7100 supports 'Hardwired TCP/IP stack' and 'Software TCP/IP stack', it calls 'Hybrid TCP/IP stack'. If user wants more SOCKETS beyond the supported 8 SOCKETS, the SOCKET in which the user wants high performance should be utilizing the "Hardwired TCP/IP stack", and the others should be using 'Software TCP/IP stack' by MACRAW mode. So it overcomes the limited capacity of 8 SOCKETS. The SOCKET of MACRAW mode can process all protocols except using in SOCKET1~7. Since the MACRAW communication is pure Ethernet packet communication (there is no other processing), the MACRAW designer should use the 'Software TCP/IP stack' to process the protocol. The MACRAW data should basically contain the 6bytes of 'Source hardware address', 6bytes of 'destination hardware address' and 2bytes of 'Ethernet type' because it is based on Ethernet MAC.

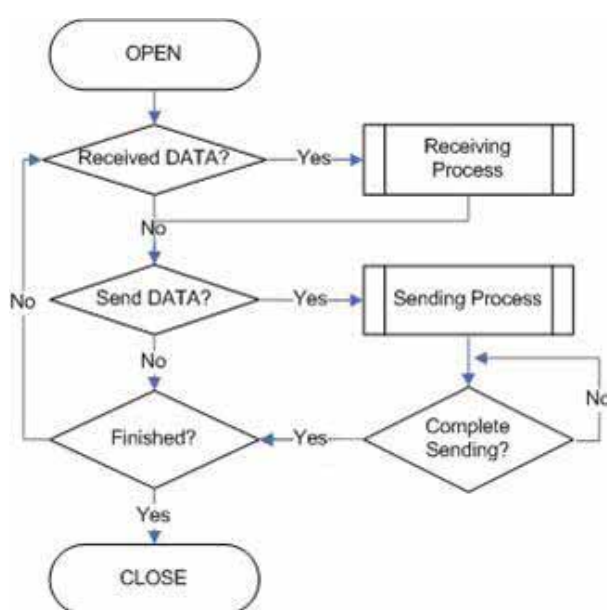


Figure 9.10 MACRAW Operation Flow

■ SOCKET Initialization

Select the SOCKET and set the SN_MR(P3:P0) to MACRAW mode. Then execute the 'OPEN' command. After the 'OPEN' command, if the Sn_SR is successfully changed to 'SOCK_MACRAW', the SOCKET initialization is completed. Since all information about

communication (Source hardware address, Source IP address, Source port number, Destination hardware address, Destination IP address, Destination port number, Protocol header, etc.) is in the 'MACRAW data', there is no more register setting.

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

■ Check received data

Refer to the "9.2.2.1 Unicast & Broadcast".

■ Receiving process

Process the MACRAW data of the SOCKET which received it in internal RX memory. The structure of the MACRAW data is as below:

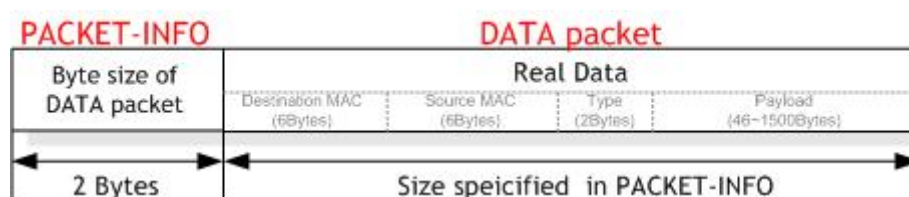


Figure 9.11 The received MACRAW data format

The MACRAW data consists of 'PACKET-INFO', 'DATA packet' and 4bytes CRC. The 'PACKET-INFO' is the length of the DATA packet. The 'DATA packet' consists of 6bytes 'Destination MAC address', 6bytes 'Source MAC address' and 2bytes 'Type', 46~1500 bytes 'Payload'. The 'Payload' of DATA packet consists of Internet protocol such as ARP, IP according to the 'Type'. The details of 'Type' please refer to the web:

(<http://www.iana.org/assignments/ethernet-numbers>)

```
{
    /* first, get the received size */
    len = Sn_RX_RSR0;    // len is received size
    /* calculate offset address */
    src_mask = Sn_RX_RD & gSn_RX_MASK;    // src_mask is offset address
    /* calculate start address(physical address) */
    src_ptr = gSn_RX_BASE + src_mask;    // src_ptr is physical start address
}
```

```

/* if overflow SOCKET RX memory */
If((src_mask + len) > (gSn_RX_MASK + 1))
{
    /* copy upper_size bytes of get_start_address to destination_address */
    upper_size = (gSn_RX_MASK + 1) - src_mask;
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), upper_size);
    /* update destination_address */
    destination_address += upper_size;
    /* copy left_size bytes of gSn_RX_BASE to destination_address */
    left_size = len - upper_size;
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), left_size);
}
else
{
    /* copy len bytes of src_ptr to destination_address */
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), len);
}

/* increase Sn_RX_RD as length of len */
Sn_RX_RD += len;

/* extract 4 bytes CRC from internal RX memory and then ignore it */
wizmemcpy((0xFE0000 + src_ptr), (0x000000 + dummy), len);

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

```

<Notice>

If the free size of the internal RX memory is smaller than the MACRAW data, a problem may occasionally occur where some parts of that PACKET-INFO and DATA packet are stored to the internal RX memory. Since the problem occurs as an analysis error for PACKET-INFO, it cannot process the MACRAW data correctly. The closer the internal RX memory is to being full, the higher the probability is for an error to occur. This problem can be resolved if user allows some loss of the MACRAW data.

The solution is as follows:

- Process the internal RX memory as fast as possible to prevent that it closes to full.
- Reduce the receiving load by reception only its MACRAW data by setting the MF (MAC Filter) bit of S0_MR in sample code of SOCKET initialization.

```

{
START:

```

```

/* sets MAC raw mode with enabling MAC filter */
SO_MR = 0x44;
/* sets OPEN command */
SO_CR = OPEN;
/* wait until Sn_SR is changed to SOCK_MACRAW */
if (Sn_SR != SOCK_MACRAW) SO_CR = CLOSE; goto START;
}

```

- If the free size of the internal RX memory is smaller than '1528 - Default MTU(1514)+PACKET-INFO(2) + DATA packet(8) + CRC(4)', close the SOCKET and process all received data. Then reopen the SOCKET. After closing the SOCKET, the received MACRAW data from closing time can be lost.

```

{
/* check the free size of internal RX memory */
if((Sn_RXMEM_SIZE(0) * 1024) - Sn_RX_RSR0(0) < 1528)
{
    recved_size = Sn_RX_RSR0(0);          /* backup Sn_RX_RSR */
    Sn_CR0 = CLOSE;                      /* SOCKET Closed */
    while(Sn_SR != SOCK_CLOSED);         /* wait until SOCKET is closed */
    /* process all data remained in internal RX memory */
    while(recved_size > 0)
    { /* calculate offset address */
        src_mask = Sn_RX_RD & gSn_RX_MASK;    // src_mask is offset address
        /* calculate start address(physical address) */
        src_ptr = gSn_RX_BASE + src_mask;    // src_ptr is physical start address
        /* if overflow SOCKET RX memory */
        If((src_mask + len) > (gSn_RX_MASK + 1))
        {
            /* copy upper_size bytes of get_start_address to destination_address */
            upper_size = (gSn_RX_MASK + 1) - src_mask;
            wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address),
            upper_size);
            /* update destination_address */
            destination_address += upper_size;
            /* copy left_size bytes of gSn_RX_BASE to destination_address */
            left_size = len - upper_size;
            wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), left_size);
        }
    }
}

```

```

    }
    else
    { /* copy len bytes of src_ptr to destination_address */
        wizmemcpy((0xFE0000 + src_ptr), (0x000000 + destination_address), len);
    }

    /* increase Sn_RX_RD as length of len */
    Sn_RX_RD += len;

    /* extract 4 bytes CRC from internal RX memory and then ignore it */
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + dummy), len);

    /* calculate the size of remained data in internal RX memory*/
    recved_size = recved_size - 2 - len - 4;
}

/* Reopen the SOCKET */
/* sets MAC raw mode with enabling MAC filter */
S0_MR = 0x44; /* or S0_MR = 0x04 */
/* sets OPEN command */
S0_CR = OPEN;

/* wait until Sn_SR is changed to SOCK_MACRAW */
while (Sn_SR != SOCK_MACRAW);
}
else /* process normally the DATA packet from internal RX memory */
{ /* This block is same as the code of "Receiving process" stage*/
}
}
}

```

■ Check send data / Sending process

The size of the data which the user wants to transmit cannot be larger than the internal TX memory and default MTU. The host generates the MACRAW data in the same format as the "Receiving process" data packet, and transmits it. At this time, if the size of the generated data is smaller than 60bytes, the transmitted Ethernet packet internally fills to 60bytes by "Zero padding" and then it is transmitted.

```

{
    /* first, get the free TX memory size */
FREESIZE:
    freesize = S0_TX_FSR0;
    if (freesize < send_size) goto FREESIZE;

    /* calculate offset address */

```

```

dst_mask = Sn_TX_WRO & gSn_TX_MASK;    // dst_mask is offset address
/* calculate start address(physical address) */
dst_ptr = gSn_TX_BASE + dst_mask;    // dst_ptr is physical start address

/* if overflow SOCKET TX memory */
if ( (dst_mask + len) > (gSn_TX_MASK + 1) )
{
    /* copy upper_size bytes of source_addr to dst_ptr */
    upper_size = (gSn_TX_MASK + 1) - dst_mask;
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + dst_ptr), upper_size);
    /* update source_addr */
    source_addr += upper_size;
    /* copy left_size bytes of source_addr to gSn_TX_BASE */
    left_size = len - upper_size;
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + gSn_TX_BASE), left_size);
}
else
{
    /* copy len bytes of source_addr to dst_ptr */
    wizmemcpy((0x000000 + source_addr), (0xFE0000 + dst_ptr), len);
}

/* increase Sn_TX_WR as length of len */
Sn_TX_WRO += send_size;

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

```

■ Check complete sending

Since the host manages all protocol processors to communicate, the timeout can not be occurred.

```

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

```

■ Check finished / SOCKET close

Refer to the “9.2.2.1 Unicast & Broadcast”.

10 Electrical Specification

Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	-0.5 to 3.6	V
V_{IN}	DC input voltage	-0.5 to 5.5 (5V tolerant)	V
V_{OUT}	DC output voltage	2 to 2.5 (GPIO)	V
		-0.5 to 3.6 (Others)	
I_{IN}	DC input current	± 5	mA
I_{OUT}	DC output current	2 to 8	mA
T_{OP}	Operating temperature	0 to 80	$^{\circ}\text{C}$
T_{STG}	Storage temperature	-55 to 125	$^{\circ}\text{C}$

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

Since the output driving voltage of GPIO is 2.5, if the user wants 3.3V, it needs additional pull-up register.

DC Characteristics

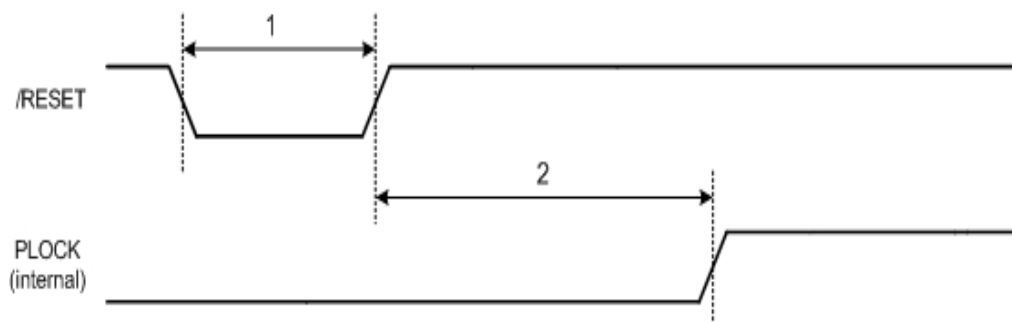
Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
V_{DD}	DC Supply voltage	Junction temperature is from -55°C to 125°C	3.0	3.3	3.6	V
V_{IH}	High level input voltage		2.0		5.5	V
V_{IL}	Low level input voltage		- 0.5		0.8	V
V_{OH}	High level output voltage	$I_{OH} = 2 \sim 16 \text{ mA}$	2.4			V
V_{OL}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

Power consumption(Driving voltage 3.3V)

Symbol	Parameter	Test Condition	Max	Unit
I_{Boot}	Current consumption	Booting	250	mA
I_{Idle}	Current consumption	Idle state	220	mA
I_{Active}	Current consumption	Whole 8 SOCKETS running	220	mA

AC Characteristics

Reset Timing



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

Data Memory Read Timing

The data memory can be accessed by MOVX instruction only. The memory access time is dependent on ready input and can be performed with minimal 2 wait states.

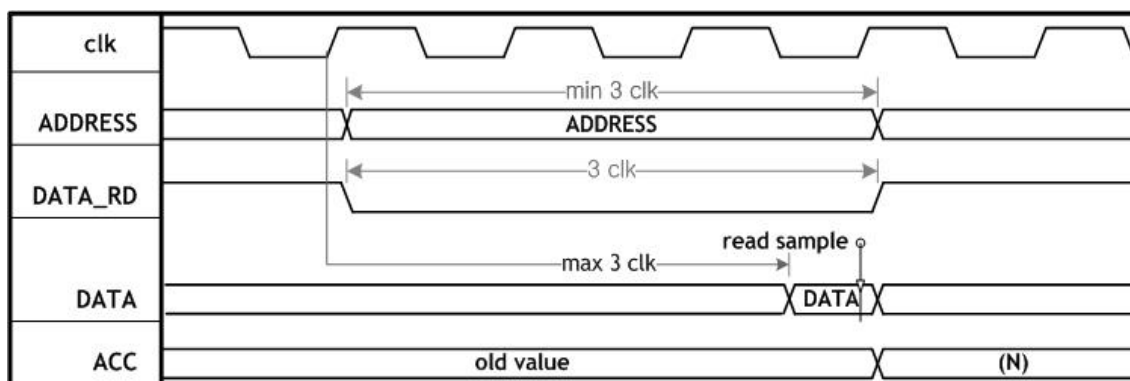


Figure 10.1 Waveform for data memory Read Cycle with Minimal Wait States(CKCON = '2')

- Note:**
1. clk - System clock frequency (clk = 88.4736 MHz)
 2. ADDRESS - Actually read the memory address
 3. DATA - Data read form address
 4. Read sample - Data of data memory has been read from (address) cell into ACC register
 5. DATA_RD - Read strobe signal

Data Memory Write Timing

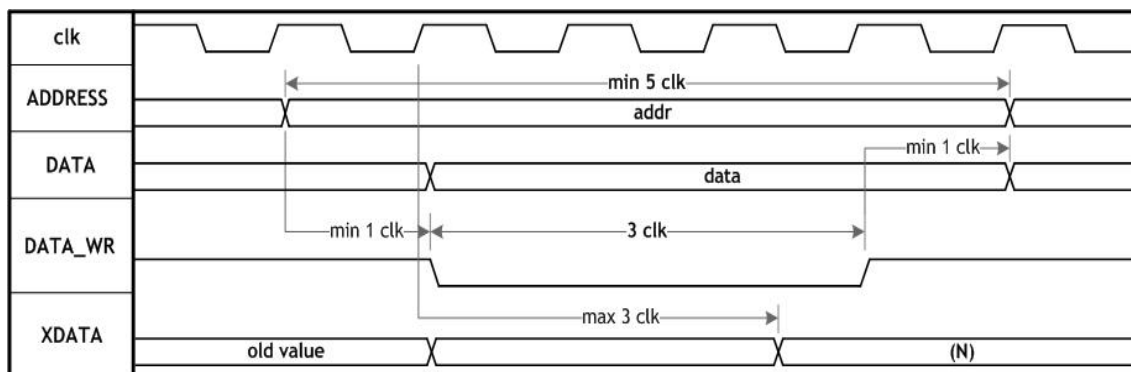


Figure 10.2 Waveform for data memory Write Cycle with Minimal Wait States(CKCON = '2')

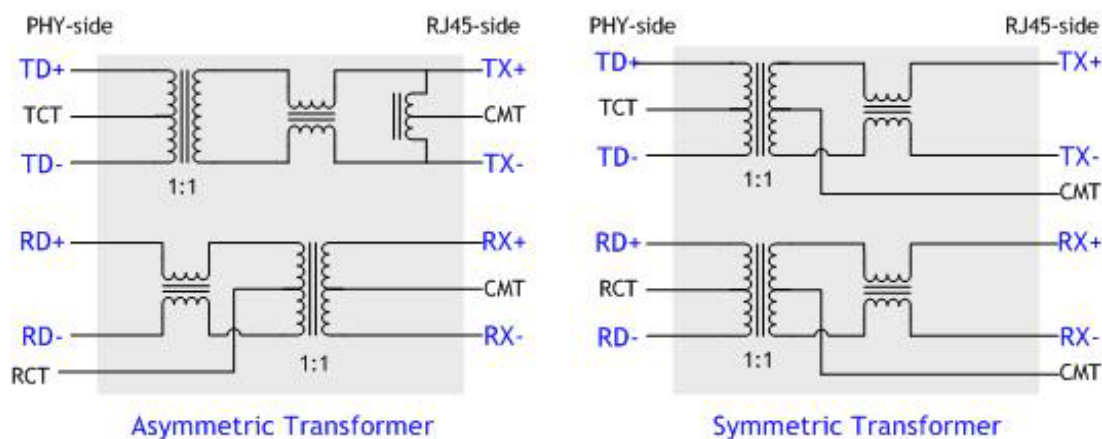
- Note:**
1. clk - System clock period (clk = 88.4736 MHz)
 2. ADDRESS - Written in the memory address
 3. DATA - Data written into address
 4. XDATA - data memory cell

Crystal Characteristics

Parameter	Range
Frequency	25 MHz
Frequency Tolerance (at 25°C)	±30 ppm
Shunt Capacitance	7pF Max
Drive Level	1 ~ 500uW (100uW typical)
Load Capacitance	18pF
Aging (at 25°C)	±3ppm / year Max

Transformer Characteristics

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



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

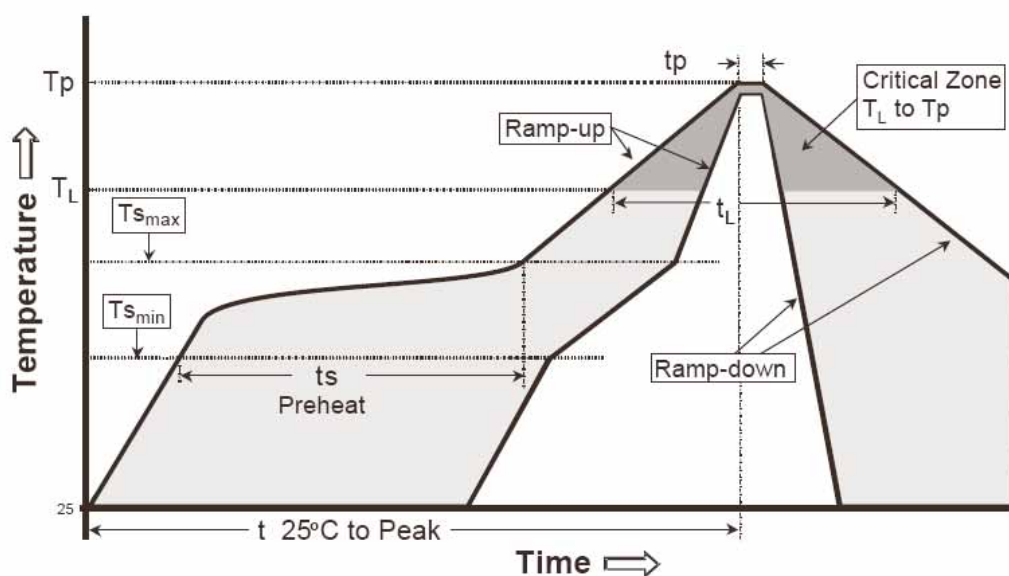
In the case of using the External PHY mode, use the transformer which is suitable for external PHY specification.

11 IR Reflow Temperature Profile (Lead-Free)

Moisture Sensitivity Level: 3

Dry Pack Required: Yes

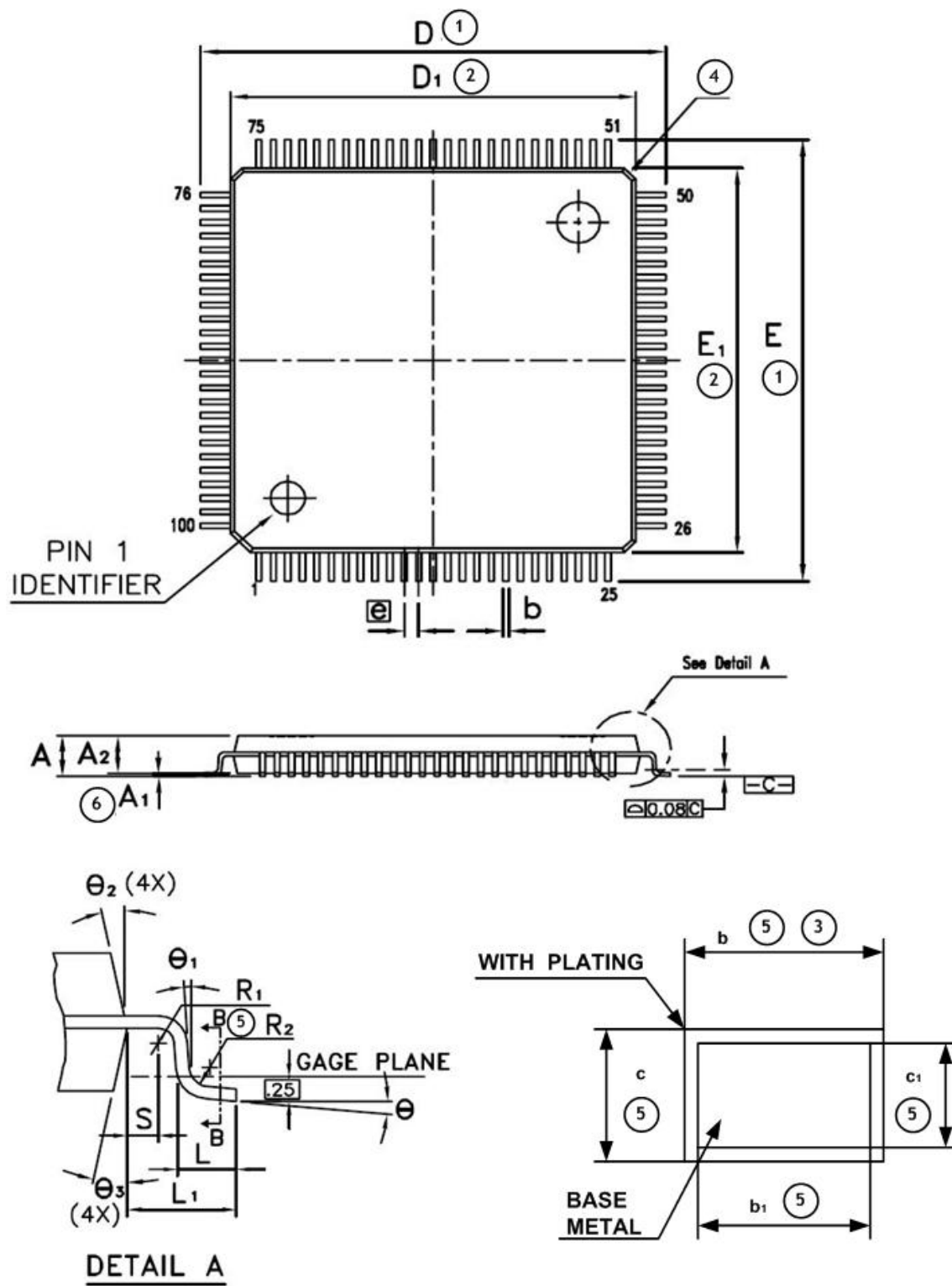
Average RAMP-Up Rate ($T_{s_{max}}$ to T_p)	3 ° C/second max.
Preheat - Temperature Min ($T_{s_{min}}$) - Temperature Max ($T_{s_{max}}$) - Time ($t_{s_{min}}$ to $t_{s_{max}}$)	150 ° C 200 ° C 60-180 seconds
Time maintained above: - Temperature (T_L) - Time (t_L)	217 ° C 60-150 seconds
Peak/Classification Temperature (T_p)	260 + 0 ° C
Time within 5 ° C of actual Peak Temperature (t_p)	20-40 seconds
RAMP-Down Rate	6 ° C/second max.
Time 25 ° C to Peak Temperature	8 minutes max.



IPC-0206-5-1

12 Package Descriptions

Package type: LQFP 100



SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	-	-	1.60	-	-	0.063
A ₁	0.05	-	0.15	0.002	-	0.006
A ₂	1.35	1.40	1.45	0.053	0.055	0.057
b	0.17	0.22	0.27	0.007	0.009	0.011
b ₁	0.17	0.20	0.23	0.007	0.008	0.009
c	0.09	-	0.20	0.004	-	0.008
c ₁	0.09	-	0.16	0.004	-	0.006
D	15.85	16.00	16.15	0.624	0.630	0.636
D ₁	13.90	14.00	14.10	0.547	0.551	0.555
E	15.85	16.00	16.15	0.624	0.630	0.636
E ₁	13.90	14.00	14.10	0.547	0.551	0.555
	0.50 BSC			0.020 BSC		
L	0.45	0.60	0.75	0.018	0.024	0.030
L ₁	1.00 REF			0.039 REF		
R ₁	0.08	-	-	0.003	-	-
R ₂	0.08	-	0.20	0.003	-	0.008
S	0.20	-	-	0.008	-	-
θ	0°	3.5°	7°	0°	3.5°	7°
θ ₁	0°	-	-	0°	-	-
θ ₂	12° TYP			12° TYP		
θ ₃	12° TYP			12° TYP		

Note :

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

13 Appendix: Performance Improvement about W7100

This section presents the benefits gained about calculation by using W7100 over standard 8051 family.

13.1 Summary

The 8-bit operation cycles of the 80C51 and W7100 with addition, subtraction, multiplication and division are as below. It is briefly shows its performance. The W7100 with 'wizmemcpy' (supported by WIZnet) function is almost 9 times faster than the 80C51.

	80C51 cycle	W7100 cycle / with user code	W7100 cycle / with wizmemcpy
ADD	36	12	4
SUB	36	12	4
MUL	96	12	6
DIV	96	20	10

In the succeeded section shows more detail performance.

13.2 8-Bit Arithmetic Functions

13.2.1 Addition

■ Immediate data

The following code performs immediate data (constant) addition to an 8-bit register.

$RX = RX + \#n$

Mnemonic	Opcode	Byte	80C51 Cycle	W7100 Cycle	
				ISP / wizmemcpy	FLASH / user code
MOV A, Rx	E8h - EFh	1	12	1	4
ADD A, #n	24h	2	12	2	4
MOV Rx, A	F8h - FFh	1	12	1	4
Sum :			36	4	12

Note. 'wizmemcpy' function are built-in inside Boot ROM in W7100. Refer to the 'Driver Guide.'

■ Direct addressing

The following code performs direct addressing addition to an 8-bit register.

$Rx = Rx + (dir)$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rx	E8h - EFh	1	12	1	4
ADD	A, dir	25h	2	12	2	4
MOV	Rx, A	F8h - FFh	1	12	1	4
Sum :				36	4	12

■ Indirect addressing

The following code performs indirect addressing addition to an 8-bit register.

 $Rx = Rx + (@Rx)$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rx	E8h - EFh	1	12	1	4
ADD	A, @Rx	26h - 27h	1	12	2	4
MOV	Rx, A	F8h - FFh	1	12	1	4
Sum :				36	4	12

■ Register addressing

The following code performs an 8-bit register to register addition.

 $Rx = Rx + Ry$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rx	E8h - EFh	1	12	1	4
ADD	A, Ry	28h - 2Fh	1	12	1	4
MOV	Rx, A	F8h - FFh	1	12	1	4
Sum :				36	3	12

13.2.2 Subtraction

■ Immediate data

The following code performs immediate data (constant) subtraction from an 8-bit register.

$Rx = Rx - \#n$

Mnemonic	Opcode	Byte	80C51 Cycle	W7100 Cycle	
				ISP / wizmemcpy	FLASH / user code
MOV A, Rx	E8h - EFh	1	12	1	4
SUBB A, #n	24h	2	12	2	4
MOV Rx, A	F8h - FFh	1	12	1	4
Sum :			36	4	12

■ Direct addressing

The following code performs direct addressing subtraction from an 8-bit register.

 $Rx = Rx - (\text{dir})$

Mnemonic	Opcode	Byte	80C51 Cycle	W7100 Cycle	
				ISP / wizmemcpy	FLASH / user code
MOV A, Rx	E8h - EFh	1	12	1	4
SUBB A, dir	25h	2	12	2	4
MOV Rx, A	F8h - FFh	1	12	1	4
Sum :			36	4	12

■ Indirect addressing subtraction

The following code performs indirect addressing subtraction from an 8-bit register.

 $Rx = Rx - (@Ry)$

Mnemonic	Opcode	Byte	80C51 Cycle	W7100 Cycle	
				ISP / wizmemcpy	FLASH / user code
MOV A, Rx	E8h - EFh	1	12	1	4
SUBB A, @Ry	26h - 27h	1	12	2	4
MOV Rx, A	F8h - FFh	1	12	1	4
Sum :			36	4	12

■ Register addressing subtraction

The following code performs an 8-bit register from register subtraction.

 $Rx = Rx - Ry$

Mnemonic	Opcode	Byte	80C51 Cycle	W7100 Cycle	
				ISP / wizmemcpy	FLASH / user code

MOV	A, Rx	E8h - EFh	1	12	1	4
SUBB	A, Ry	28h - 2Fh	1	12	1	4
MOV	Rx, A	F8h - FFh	1	12	1	4
Sum :				36	3	12

13.2.3 Multiplication

The following code performs the 8-bit register multiplication.

$Rx = Rx * Ry$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rx	E8h - EFh	1	12	1	4
MOV	B, Ry	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
MOV	Rx, A	F8h - FFh	1	12	1	4
Sum :				96	6	12

13.2.4 Division

The following code performs the 8-bit register division.

$Rx = Rx / Ry$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rx	E8h - EFh	1	12	1	4
MOV	B, Ry	88h - 8Fh	2	24	2	4
DIV	AB	84h	1	48	6	8
MOV	Rx, A	F8h - FFh	1	12	1	4
Sum :				96	10	20

13.3 16-Bit Arithmetic Functions

13.3.1 Addition

The following code performs 16-bit addition. The first operand and result are located in registers pair RaRb. The second operand is located in registers pair RxRy.

$$RaRb = RaRb + RxRy$$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rb	E8h - EFh	1	12	1	4
ADD	A, Ry	28h - 2Fh	1	12	1	4
MOV	Rb, A	F8h - FFh	1	12	1	4
MOV	A, Ra	E8h - EFh	1	12	1	4
ADDC	A, Rx	38h - 3Fh	1	12	1	4
MOV	Ra, A	F8h - FFh	1	12	1	4
Sum :				72	6	24

13.3.2 Subtraction

The following code performs 16-bit subtraction. The first operand and result are located in registers pair RaRb. The second operand is located in registers pair RxRy.

$$RaRb = RaRb - RxRy$$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
CLR	C	C3h	1	12	1	4
MOV	A, Rb	E8h - EFh	1	12	1	4
SUBB	A, Ry	98h - 9Fh	1	12	1	4
MOV	Rb, A	F8h - FFh	1	12	1	4
MOV	A, Ra	E8h - EFh	1	12	1	4
SUBB	A, Rx	98h - 9Fh	1	12	1	4
MOV	Ra, A	F8h - FFh	1	12	1	4
Sum :				84	7	28

13.3.3 Multiplication

The following code performs 16-bit multiplication. The first operand and result are located in registers pair RaRb. The second operand is located in registers pair RxRy.

$$RaRb = RaRb * RxRy$$

Mnemonic	Opcode	Byte	80C51 Cycle	W7100 Cycle	
				ISP / wizmemcpy	FLASH / user code

MOV	A, Rb	E8h - EFh	1	12	1	4
MOV	B, Ry	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
MOV	Rz, B	A8h - AFh	2	24	3	4
XCH	A, Rb	C8h - CFh	1	12	2	4
MOV	B, Rx	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
ADD	A, Rz	28h - 2Fh	1	12	1	4
XCH	A, Ra	C8h - CFh	1	12	2	4
MOV	B, Ry	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
ADD	A, Ra	28h - 2Fh	1	12	1	4
MOV	Ra, A	F8h - FFh	1	12	1	4
Sum :				312	23	52

13.4 32-bit Arithmetic Functions

13.4.1 Addition

The following code performs 32-bit addition. The first operand and result are located in four registers RaRbRcRd. The second operand is located in four registers RvRxRyRz.

RaRbRcRd = RaRbRcRd + RvRxRyRz

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, Rd	E8h - EFh	1	12	1	4
ADD	A, Rz	28h - 2Fh	1	12	1	4
MOV	Rd, A	F8h - FFh	1	12	1	4
MOV	A, Rc	E8h - EFh	1	12	1	4
ADDC	A, Ry	38h - 3Fh	1	12	1	4
MOV	Rc, A	F8h - FFh	1	12	1	4
MOV	A, Rb	E8h - EFh	1	12	1	4
ADDC	A, Rx	38h - 3Fh	1	12	1	4
MOV	Rb, A	F8h - FFh	1	12	1	4
MOV	A, Ra	E8h - EFh	1	12	1	4
ADDC	A, Rv	38h - 3Fh	1	12	1	4
MOV	Ra, A	F8h - FFh	1	12	1	4

Sum :	144	12	48
-------	-----	----	----

13.4.2 Subtraction

The following code performs 32-bit subtraction. The first operand and result are located in four registers RaRbRcRd. The second operand is located in four registers RvRxRyRz.

$RaRbRcRd = RaRbRcRd - RvRxRyRz$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
CLR	C	C3h	1	12	1	4
MOV	A, Rd	E8h - EFh	1	12	1	4
SUBB	A, Rz	98h - 9Fh	1	12	1	4
MOV	Rd, A	F8h - FFh	1	12	2	4
MOV	A, Rc	E8h - EFh	1	12	1	4
SUBB	A, Ry	98h - 9Fh	1	12	2	4
MOV	Rc, A	F8h - FFh	1	12	2	4
MOV	A, Rb	E8h - EFh	1	12	1	4
SUBB	A, Rx	98h - 9Fh	1	12	1	4
MOV	Rb, A	F8h - FFh	1	12	1	4
MOV	A, Ra	E8h - EFh	1	12	1	4
SUBB	A, Rv	98h - 9Fh	1	12	1	4
MOV	Ra, A	F8h - FFh	1	12	1	4
Sum :				156	13	52

13.4.3 Multiplication

The following code performs 32-bit multiplication. The first operand and result are located in four registers RaRbRcRd. The second operand is located in four registers RvRxRyRz.

$RaRbRcRd = RaRbRcRd * RvRxRyRz$

Mnemonic		Opcode	Byte	80C51 Cycle	W7100 Cycle	
					ISP / wizmemcpy	FLASH / user code
MOV	A, R0	E8h - EFh	1	12	1	4
MOV	B, R7	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
XCH	A, R4	C8h - CFh	1	12	2	4
MOV	B, R3	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4

ADD	A, R4	28h - 2Fh	1	12	1	4
MOV	R4, A	F8h - FFh	1	12	1	4
MOV	A, R1	E8h - EFh	1	12	1	4
MOV	B, R6	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
ADD	A, R4	28h - 2Fh	1	12	1	4
MOV	R4, A	F8h - FFh	1	12	1	4
MOV	B, R2	88h - 8Fh	2	24	2	4
MOV	A, R5	E8h - EFh	1	12	2	4
MUL	AB	A4h	1	48	2	4
ADD	A, R4	28h - 2Fh	1	12	1	4
MOV	R4, A	F8h - FFh	1	12	1	4
MOV	A, R2	E8h - EFh	1	12	1	4
MOV	B, R6	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
XCH	A, R5	C8h - CFh	1	12	2	4
MOV	R0, B	A8h - AFh	2	24	3	4
MOV	B, R3	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
ADD	A, R5	28h - 2Fh	1	12	1	4
XCH	A, R4	C8h - CFh	1	12	2	4
ADDC	A, R0	38h - 3Fh	1	12	1	4
ADD	A, B	25h	2	12	2	4
MOV	R5, A	F8h - FFh	1	12	1	4
MOV	A, R1	E8h - EFh	1	12	1	4
MOV	B, R7	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
ADD	A, R4	28h - 2Fh	1	12	1	4
XCH	A, R5	C8h - CFh	1	12	2	4
ADDC	A, B	35h	2	12	2	4
MOV	R4, A	F8h - FFh	1	12	1	4
MOV	A, R3	E8h - EFh	1	12	1	4
MOV	B, R6	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
MOV	R6, A	F8h - FFh	1	12	1	4
MOV	R1, B	A8h - AFh	2	24	3	4

MOV	A, R3	E8h - EFh	1	12	1	4
MOV	B, R7	88h - 8Fh	2	24	2	4
MUL	AB	A4h	1	48	2	4
XCH	A, R7	C8h - CFh	1	12	2	4
XCH	A, B	C5h	2	12	3	4
ADD	A, R6	28h - 2Fh	1	12	1	4
XCH	A, R5	C8h - CFh	1	12	2	4
ADDC	A, R1	38h - 3Fh	1	12	1	4
MOV	R6, A	F8h - FFh	1	12	1	4
CLR	A	E4h	1	12	1	4
ADDC	A, R4	38h - 3Fh	1	12	1	4
MOV	R4, A	F8h - FFh	1	12	1	4
MOV	A, R2	E8h - EFh	1	12	1	4
MUL	AB	A4h	1	48	2	4
ADD	A, R5	28h - 2Fh	1	12	1	4
XCH	A, R6	C8h - CFh	1	12	2	4
ADDC	A, B	38h - 3Fh	2	12	2	4
MOV	R5, A	F8h - FFh	1	12	1	4
CLR	A	E4h	1	12	1	4
ADDC	A, R4	38h - 3Fh	1	12	1	4
MOV	R4, A	F8h - FFh	1	12	1	4
Sum :				1248	99	252