W7100A Datasheet

Version 1.2.1

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

Table of Contents

List of Figures

List of Tables

1 Overview

1.1 Introduction

iMCU W7100A는 8051호환 마이크로 컨트롤러, 64KB SRAM과 hardwired TCP/IP Core 를 내장한 고성능 one-chip 인터넷 솔루션 이다. Hardwired TCP/IP 코어는 Ethernet MAC과 PHY 를 내장한 TCP/IP stack으로서 시장에서 그 성능을 증명해 왔다. Hardwired TCP/IP stack에 포함된 TCP, UDP, IPv4, ICMP, ARP, IGMP 그리고 PPPoE는 수년간 다양한 응용분야에 이용되어 왔다.

1.2 W7100A Features

- Fully software compatible with industrial standard 8051
- Pipelined architecture which enables execution of instructions 4~5 times faster than a standard 8051
- 10BaseT/100BaseTX Ethernet PHY embedded
- Power down mode supported for saving power consumption
- Hardwired TCP/IP Protocols: TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet
- 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
- 2 Data Pointers (DPTRs) for fast memory blocks processing
  - Advanced INC & DEC modes
  - Auto-switch of current DPTR
- 64KBytes Data Memory (RAM)
- 255Bytes data FLASH, 64KBytes Code Memory, 2KBytes Boot Code Memory
- Up to 16M bytes of external (off-chip) data 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
- High Product Endurance
  - Minimum 100,000 program/erase cycles
  - Minimum 10 years data retention

1.3 W7100A Block Diagram & Features

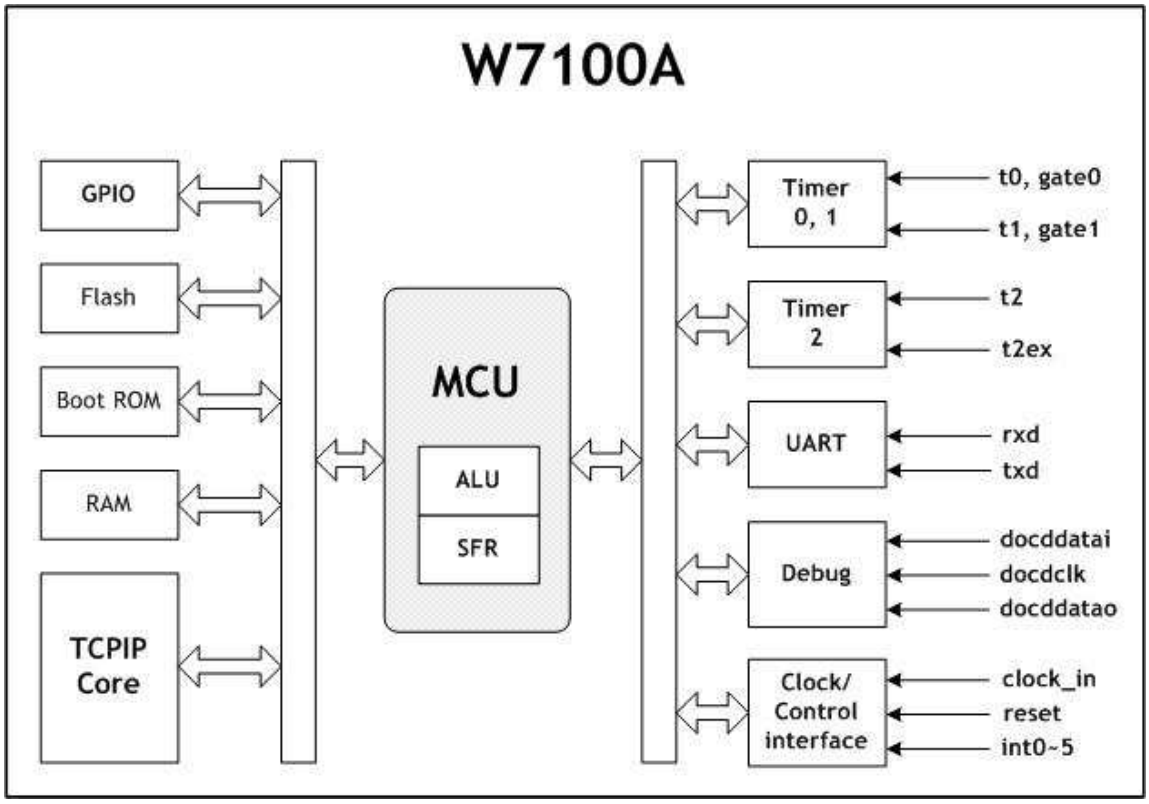


Figure 1.1 W7100A Block Diagram

W7100A 내부 블록 다이어그램을 그림1.1에 나타내었다.

ALU – 산술, 논리연산을 수행한다. Accumulator (ACC)와 Program Status Word (PSW), B 레지스터, 그리고 arithmetic unit, logic unit, multiplier, divider등 관련된 Logic을 포함하고 있다.

SFR – Special function register를 제어한다. 표준, 사용자 지정 register들과 관련된 logic을 포함하고 있다. 사용자가 정의한 외부 장비를 direct addressing mode 명령을 이용해서 빠르게 access (read, write, modify) 할 수 있다.

1.3.1 ALU (Arithmetic Logic Unit)

W7100A는 표준 8051 microcontroller와 완벽하게 호환된다. 그리고 관련된 모든 명령을 포함하고 있다. W7100A는 고속 명령어 수행을 가능하게 하는 많은 구조적 장점을 가지고 있다. W7100A MCU의 ALU는 광범위한 data 처리를 하고 이것은 8-bit ALU, ACC (0xE0) register, B (0xF0) register와 PSW (0xD0) register 로 구성된다.

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

B register는 곱셈연산과 나눗셈연산 중에 사용된다. 다른 경우에, B register는 보통 SFR처럼 동작한다.

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

ALU는 덧셈, 뺄셈, 곱셈, 나눗셈 과 같은 산술연산과 증가 (increment), 감소 (decrement), BCD-decimal-add-adjust, 비교 (compare) 연산 등을 수행한다. 논리 unit은 다른 연산들을 수행하기 위해 AND, OR, Exclusive OR, complement, rotation을 이용한다. Boolean 프로세서는 set, clear, complement, jump-if-not-set, jump-if-set-and-clear 그리고 move to/from carry와 같은 bit연산을 수행한다.

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

PSW register는 MCU의 현재 상태를 반영할 수 있는 몇몇 bit들을 포함하고 있다.

1.3.2 TCPIPCore

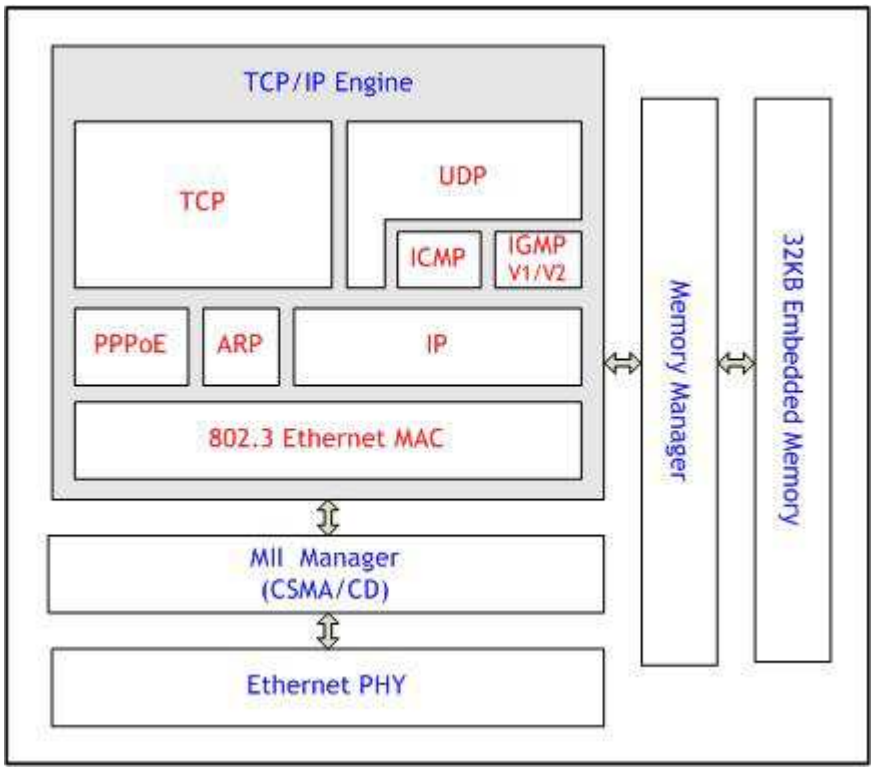


Figure 1.6 TCPIPCore Block Diagram

Ethernet PHY

W7100A는 10BaseT/100BaseTX Ethernet PHY를 내장하고 있다. Half-duplex와 Full-duplex를 모두 지원하며 Auto-negotiation과 Auto-MDI/MDIX를 지원한다. 또한 Link, TX, RX 상태, Collision, Speed와 Duplex 6개의 네트워크 indicator LED 출력을 지원한다.

TCPIP Engine

TCPIP 엔진은 WIZnet의 네트워크 프로토콜기술을 기반으로 하는 Hardwired logic이다.

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

  CSMA/CD (Carrier Sense Multiple Access with Collision Detect) 을 지원하며 프로토콜은 48-bit source/destination MAC address를 기반으로 하고 있다.

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

  이 프로토콜은 Ethernet을 통해 PPP서비스를 이용한다. Ethernet 프레임 전송 시에 Payload (PPP 프레임) 를 encapsulate하고, 수신 시에 PPP 프레임을 de-capsulate한다. PPPoE는 PPPoE서버와 PAP/CHAP인증을 이용해서 PPP 통신을 한다.

- **ARP(Address Resolution Protocol)**

  ARP는 IP address를 이용한 MAC address resolution protocol이다. 이 프로토콜은 서로 MAC address를 결정하기 위해 ARP-reply와 ARP-request를 교환한다.

- **IP (Internet Protocol)**

  IP 계층 (layer) 에서 동작하며 데이터 통신을 지원한다. IP fragmentation 은 지원하지

않기 때문에 fragmented packet은 수신할 수 없다. TCP나 UDP를 제외한 모든 프로토콜 number를 지원한다. TCP나 UDP의 경우 hardwired embedded TCPIP stack을 지원한다.

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

ICMP는 information, unreachable destination 을 지원하는 프로토콜이다. Ping request ICMP packet이 수신되면, Ping reply ICMP packet을 송신한다.

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

IGMP Join/Leave와 같은 IGMP 메시지를 처리하는 프로토콜이다. IGMP는 오직 UDP multicast mode에서만 동작한다. Version 1 과 2 인 IGMP logic만 지원한다. 새로운 버전의 IGMP을 사용하려면, 직접 IGMP를 IP 계층에 구현해야만 한다.

- **UDP(User Datagram Protocol)**

UDP계층에서 데이터 통신을 지원하는 프로토콜이다. 사용자는 unicast, multicast, broadcast와 같은 사용자 데이터 그램을 지원한다.

- **TCP(Transmission Control Protocol)**

이 프로토콜은 TCP 계층에서 동작하며 데이터 통신을 지원한다. TCP 서버와 클라이언트 mode를 지원한다.

1.4 Pin Description

1.4.1 Pin Layout

Package type: LQFP 100

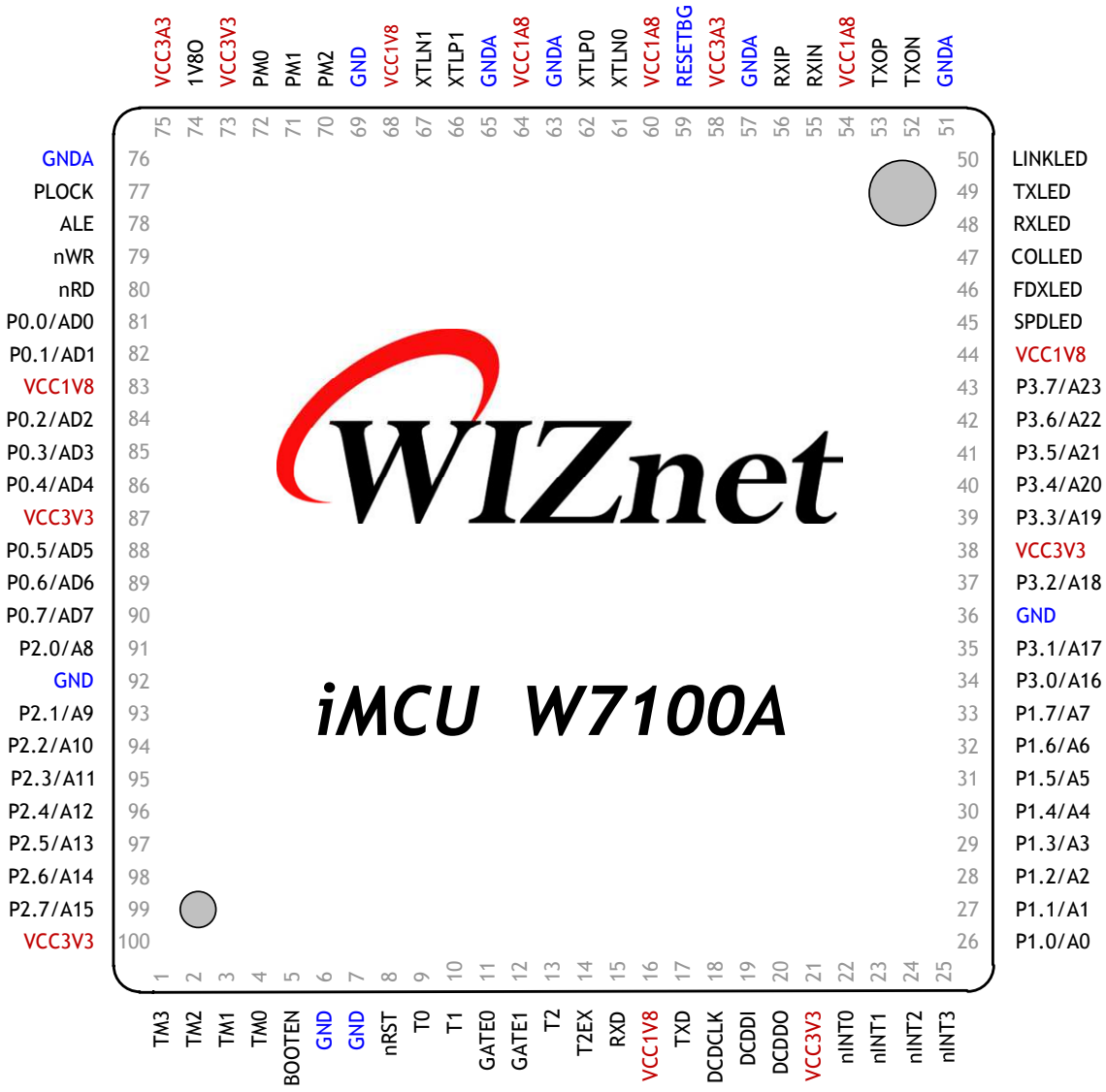


Figure 1.7 W7100A Pin Layout

Package type: QFN 64

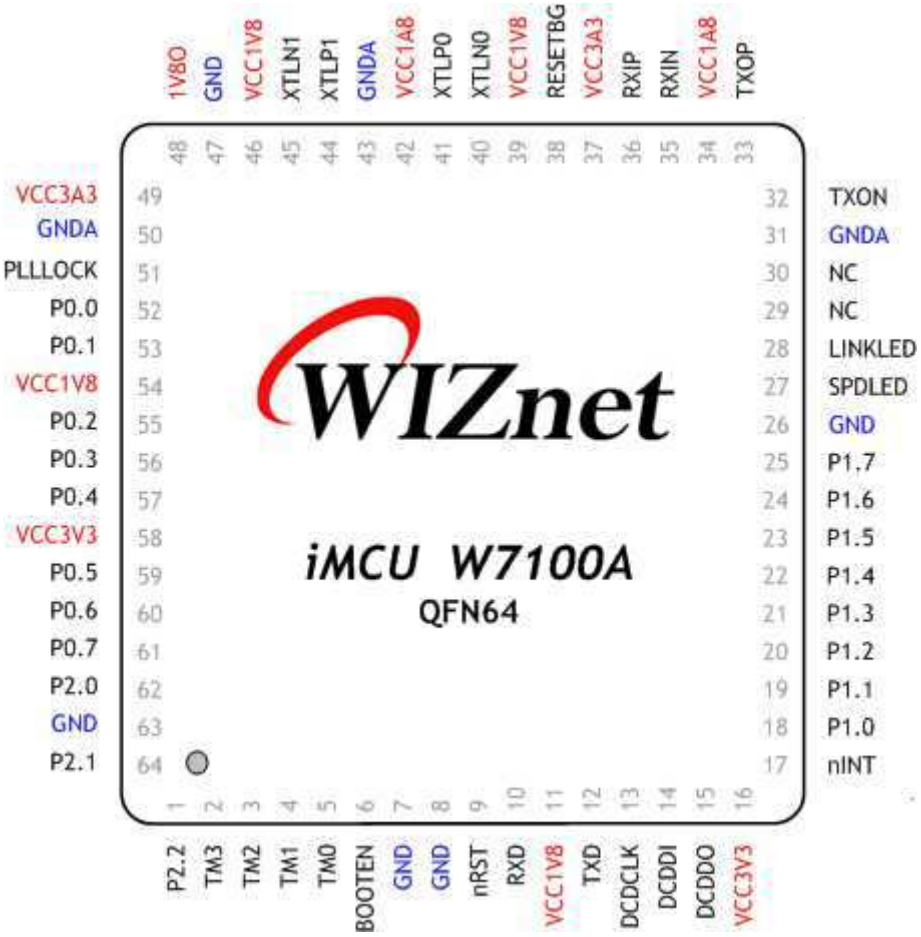


Figure 1.8 W7100A QFN 64 Pin Layout

1.4.2 Pin Description

아래 테이블에 Pin 기능들을 설명하였다. Tri – state pin이나 내부 신호는 없다.

Type	Description
I	입력
O	8mA 출력 driving 전류
IO	입/출력 (Bidirectional)
Pu	4.7KΩ저항 내부 pulled-up
Pd	85KΩ저항 내부 pulled-down

1.4.2.1 Configuration

<table>
<tr><th rowspan="2">Pin name</th><th colspan="2">Pin number</th><th rowspan="2">I/O</th><th rowspan="2">Pu/Pd</th><th rowspan="2">Description</th></tr>
<tr><th>100pin</th><th>64pin</th></tr>
<tr><td>nRST</td><td>8</td><td>9</td><td>I</td><td>-</td><td>Global asynchronous reset, Active low</td></tr>
<tr><td>TM3-0</td><td>1,2,
3,4</td><td>2,3,
4,5</td><td>I</td><td>Pd</td><td>Must be connected to GND; value ‘0000’</td></tr>
<tr><td>PM2 - 0</td><td>70,
71,
72</td><td>-</td><td>I</td><td>Pd</td><td>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, 모든 기능과 Auto-negotiation기능 활성화</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>100 BASE-TX FDX/HDX 기능과 Auto-negotiation기능 활성화</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>10 BASE-T FDX/HDX 기능과 Auto-negotiation 기능 활성화</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>100 BASE-TX FDX 직접 선택</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>100 BASE-TX HDX 직접 선택</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>10 BASE-T FDX 직접 선택</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>10 BASE-T HDX 직접 선택</td></tr>
</table>
FDX : Full-Duplex, HDX : Half-Duplex</td></tr>
<tr><td>BOOTEN</td><td>5</td><td>6</td><td>I</td><td>Pd</td><td>Boot Enable/Disable
0 - 사용자 Application 실행
Code FLASH 의 시작주소 0x0000로 jump
1- Enable Boot
Boot ROM의 Boot code실행</td></tr>
<tr><td>PLOCK</td><td>77</td><td>51</td><td>O</td><td>-</td><td>PLL Lock line, 내부 PLL의 locked 상태를 알려줌</td></tr>
</table>

1.4.2.2 Timer

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
Timer0, 1 Interface					
T0	9	-	I	-	Timer0 외부 clock 입력
T1	10	-	I	-	Timer1 외부 clock 입력
GATE0	11	-	I	-	Timer0 gate 컨트롤
GATE1	12	-	I	-	Timer1 gate 컨트롤

Timer2 Interface					
T2	13	-	I	-	Timer2 외부 clock 입력
T2EX	14	-	I	-	Timer2 Capture/Reload trigger

1.4.2.3 UART

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
RXD	15	10	I	-	Serial 수신 입력
TXD	17	12	O	-	Serial 송신 출력

1.4.2.4 DoCD™ Compatible Debugger

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
DCDCLK	18	13	O	-	DoCD clock
DCDDI	19	14	I	-	DoCD 데이터 입력
DCDDO	20	15	O		DoCD 데이터 출력

1.4.2.5 Interrupt / Clock

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
nINT0	22	17	I	-	외부 interrupt0
nINT1	23	-	I	-	외부 interrupt1
nINT2	24	-	I	-	외부 interrupt2
nINT3	25	-	I	-	외부 interrupt3
XTLN0	61	40	O	-	WIZnet Core용 clock output, 25MHz Crystal 혹은 Ceramic resonator 병렬연결, Oscillator를 사용할 경우 floating 함
XTLP0	62	41	I	-	WIZnet Core용 clock input, 25MHz Crystal 혹은 Ceramic resonator와 병렬연결, Oscillator를 사용할 경우 1.8V OSC의 output과 연결함
XTLN1	67	45	O	-	MCU Core용 clock output, 11.0592MHz Crystal 혹은 Ceramic resonator와 병렬연결, Oscillator를 사용할 경우 floating 함
XTLP1	66	44	I	-	MCU Core용 clock input, 11.0592MHz Crystal 혹은 Ceramic resonator와 병렬연결, Oscillator를 사용할 경우 1.8V OSC의 output과 연결함

1.4.2.6 GPIO

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
P0.0	81	52	IO	-	Port0 입/출력, 외부 메모리 Data0, Addr0
P0.1	82	53	IO	-	Port0 입/출력, 외부 메모리 Data1, Addr1
P0.2	84	55	IO	-	Port0 입/출력, 외부 메모리 Data2, Addr2
P0.3	85	56	IO	-	Port0 입/출력, 외부 메모리 Data3, Addr3
P0.4	86	57	IO	-	Port0 입/출력, 외부 메모리 Data4, Addr4
P0.5	87	59	IO	-	Port0 입/출력, 외부 메모리 Data5, Addr5
P0.6	89	60	IO	-	Port0 입/출력, 외부 메모리 Data6, Addr6
P0.7	90	61	IO	-	Port0 입/출력, 외부 메모리 Data7, Addr7
P1.0	26	18	IO	-	Port1 입/출력, 외부 메모리 Addr0
P1.1	27	19	IO	-	Port1 입/출력, 외부 메모리 Addr1
P1.2	28	20	IO	-	Port1 입/출력, 외부 메모리 Addr2
P1.3	29	21	IO	-	Port1 입/출력, 외부 메모리 Addr3
P1.4	30	22	IO	-	Port1 입/출력, 외부 메모리 Addr4
P1.5	31	23	IO	-	Port1 입/출력, 외부 메모리 Addr5
P1.6	32	24	IO	-	Port1 입/출력, 외부 메모리 Addr6
P1.7	33	25	IO	-	Port1 입/출력, 외부 메모리 Addr7
P2.0	91	62	IO	-	Port2 입/출력, 외부 메모리 Addr8
P2.1	93	64	IO	-	Port2 입/출력, 외부 메모리 Addr9
P2.2	94	1	IO	-	Port2 입/출력, 외부 메모리 Addr10
P2.3	95	-	IO	-	Port2 입/출력, 외부 메모리 Addr11
P2.4	96	-	IO	-	Port2 입/출력, 외부 메모리 Addr12
P2.5	97	-	IO	-	Port2 입/출력, 외부 메모리 Addr13
P2.6	98	-	IO	-	Port2 입/출력, 외부 메모리 Addr14
P2.7	99	-	IO	-	Port2 입/출력, 외부 메모리 Addr15
P3.0	34	-	IO	-	Port3 입/출력, 외부 메모리 Addr16
P3.1	35	-	IO	-	Port3 입/출력, 외부 메모리 Addr17
P3.2	37	-	IO	-	Port3 입/출력, 외부 메모리 Addr18
P3.3	39	-	IO	-	Port3 입/출력, 외부 메모리 Addr19
P3.4	40	-	IO	-	Port3 입/출력, 외부 메모리 Addr20
P3.5	41	-	IO	-	Port3 입/출력, 외부 메모리 Addr21
P3.6	42	-	IO	-	Port3 입/출력, 외부 메모리 Addr22
P3.7	43	-	IO	-	Port3 입/출력, 외부 메모리 Addr23

주의: GPIO의 입/출력 driving 전압은 Px_PU/Px_PD SFR을 설정하여 제어한다.

참고: 외부 메모리를 사용하는 경우, GPIO0~3은 외부 메모리 주소와 데이터 전송에 이용됨.

자세한 사항은 '2.3 External Data Memory Access' 참고.

1.4.2.7 External Memory Interface

Pin name	Num	I/O Type	Description
ALE	78	O	Data memory address bus [7:0] latch enable
nWR	79	OL	External data memory write
nRD	80	OL	External data memory read

참고 : 외부 메모리를 Standard 8051 Interface로 사용 시, ALE pin을 제어하여 P0[7:0]을 Data와 Address로 번갈아 사용

1.4.2.8 Media Interface

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
TXON	52	32	O	-	TXON/TXOP 신호 쌍, 전송 시에 differential 데이터는 TXON/TXOP 신호 쌍을 통해 전송됨
TXOP	53	33	O	-	
RXIN	55	35	I	-	RXIN/RXIP 신호 쌍, 수신 시에 differential 데이터는 RXIN/RXIP 신호 쌍을 통해 수신됨
RXIP	56	36	I	-	
RESETBG	59	38	I	-	PHY Off-chip 저항, 12.3kΩ±1% 저항을 통해 그라운드에 연결됨, 'Reference schematic'을 참고

최상의 성능을 위해서는,

1. RXIP/RXIN 신호 쌍 (RX) 을 가능한 같은 길이로 만듦.
2. TXOP/TXON 신호 쌍 (TX) 을 가능한 같은 길이로 만듦.
3. RXIP와 RXIN 신호를 가능한 가깝게 위치시킴.
4. TXIP와 TXIN 신호를 가능한 가깝게 위치시킴.
5. RX와 TX 신호 쌍은 noisy 신호와 멀리 위치시킴.
6. TX/RX 신호 쌍의 간격을 일정하게 유지함.

자세한 사항은 'W5100 Layout Guide.pdf'를 참고.

1.4.2.9 Network Indicator LED

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
SPDLED	45	27	O	-	Link speed LED Low: 100Mbps High: 10Mbps
FDXLED	46	-	O	-	Full duplex LED Low: Full-duplex High: Half-duplex

COLLED	47	-	O	-	Collision LED Low: Collision detected (only half-duplex)
RXLED	48	-	O	-	Receive activity LED Low: Receive signal detected on RXIP/RXIN
TXLED	49	-	O	-	Transmit activity LED Low: Transmit signal detected on TXOP/TXON
LINKLED	50	28	O	-	Link LED Low: Link (10/100M) is detected

1.4.2.10 Power Supply Signal

Pin name	Pin number		I/O	Pu/Pd	Description
	100pin	64pin			
VCC3A3	58, 75	37, 49	Power	-	Analog 3.3V power supply 안정적인 전원 공급을 위해 VCC3A3 과 GNDA사이에 10uF tantalum capacitor를 연결
VCC3V3	21,38, 73,87, 100	16, 58	Power	-	Digital 3.3V power supply 각 VCC와 GND쌍에는 0.1uF decoupling capacitor를 연결, 1uH ferrite bead는 VCC3V3과 VCC3A3을 구분하는데 사용
VCC1A8	54,60, 64	34, 42	Power	-	Analog 1.8V power supply
VCC1V8	16,44, 68, 83	39,46, 54, 11	Power	-	Digital 1.8V power supply 각 VCC와 GND쌍에는 0.1uF decoupling capacitor를 연결
GNDA	51,57, 63,65, 76	31, 43, 50	Power	-	Analog ground PCB 레이아웃에 analog ground 면적은 가능한 크게 디자인 하는 것이 좋음
GND	6, 7, 36, 69, 92	7, 8, 26, 47, 63	Power	-	Digital ground PCB 레이아웃에 digital ground 면적은 가능한 크게 디자인 하는 것이 좋음
1V8O	74	48	Power	-	1.8V regulated output voltage Core operation 전원과 내부 regulator에 의해 1.8V/150mA 전원을 만듦 (VCC1A8, VCC1V8) 출력 주파수를 안정시키기 위해 1V8O과 GND사이에 3.3uF tantalum capacitor를 연결하고 고주파 noise decoupling을 위해 0.1uF capacitor를 연결함

					VCC1V8에 연결된 1V80에 1uH inductor 를 연결하고 VCC1A8에 연결함 <Notice> 1V80은 W7100A 전원으로만 사용하고 다른 device와는 연결하지 않음

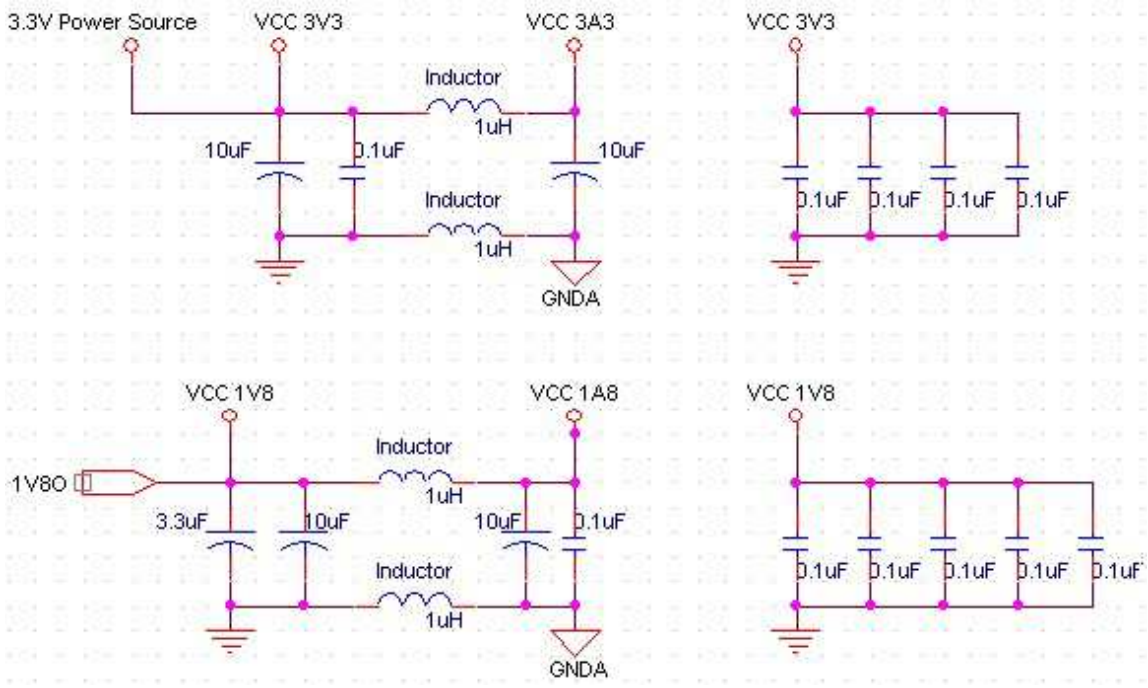


Figure 1.9 Power Design

1.5 64pin package description

1.5.1 Difference between 100 and 64pin package

Difference	64 pin	100 pin
Deleted pin	T0, T1, GATE0, GATE1, T2, T2EX, nINT1, nINT2, nINT3, FDXLED, COLLED, RXLED, TXLED, PM2, PM1, PM0, EXTALE, EXTDATAWR, EXTDATARD, GPIO3[0:7], GPIO2[3:7]	-
External memory	X	O
PHY mode setting	only use SFR	use SFR and PM pins
GPIO	max 19pin	max 32pin

*Note: 64pin package의 경우 PHY모드를 SFR을 이용해서 설정해야만 한다. 그러므로 칩 초기화 시 PHYCONF SFR의 MODE_EN bit를 설정해서 MODE2~0 bit를 이용해서 PHY mode를 설정할 수 있도록 한다. 그 다음 MODE2~0의 값을 설정하고 PHY_RSTn bit를 이용해서 PHY를 reset해야만 칩이 초기화되어 정상동작 한다. **그러므로 64pin package칩을 사용할 때는 반드시 아래 코드가 초기화 루틴에 포함되어 있어야만 한다.**

PHYCONF SFR에 대한 자세한 내용은 section 2.5.10 'New & Extended SFR' 을 참조하기 바란다.

```
PHYCONF |= 0x08;    // MODE_EN bit enable
PHYCONF &= 0xF8;    // MODE2 ~ 0 value is 0 (normal mode)
PHYCONF |= 0x20;    // Set the PHY_RSTn bit (reset bit)
Delay();                  // Delay for reset timing (refer to the section 10 'Reset Timing')
PHYCONF &= ~(0x20); // Clear the PHY_RSTn bit
```

2 Memory

W7100A의 메모리는 'Code Memory'와 'Data Memory'의 두 가지 타입으로 나뉜다. Code, Data memory별로 각각 memory lock을 설정할 수 있다. Lock기능을 설정하면 외부에서 메모리를 read하는 것을 방지할 수 있다. Memory lock이 설정되면 W7100A Debugger를 사용할 수 없기 때문에 주의해야 한다. Lock기능 사용에 대한 보다 자세한 내용은 "WizISP Program Guide"문서를 참조하기 바란다. W7100A의 Memory 구성을 Figure 2.1에 간략히 나타내었다.

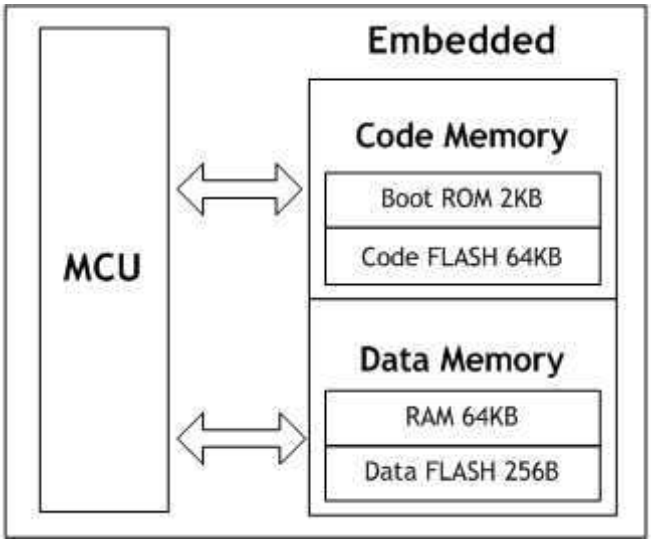


Figure 2.1 Code / Data Memory Connections

*Note:

전원 공급 장치의 전압이 충분치 않은 동안의 시스템 오작동을 방지하기 위하여 외부 전원 공급 장치의 저전압 검출기(Low Voltage Detector) 사용을 권장하며, 이를 통해 비휘발성 메모리(Non-volatile Memory)와 SRAM, CPU의 손상을 예방할 수 있다. WIZnet 웹사이트의 iMCU7100EVB Schematic을 참조하기 바란다.

(Example ICs : MAX811, MIC811, DS1811)

2.1 Code Memory

'Code Memory'는 0x0000부터 0x07FF까지의 주소를 갖는 Boot ROM과 0x0000부터 0xFFFF까지의 주소를 갖는 Code FLASH로 구성된다. 시스템이 reset된 다음 Boot ROM의 code가 실행되는데 이 때 BOOTEN핀에 따라 동작이 달라진다. Figure 2.2는 시스템부팅 후에 BOOTEN핀에 따른 BootROM코드의 flow를 나타내고 있다. BOOTEN = 1인 경우 ISP 프로세스가 실행되고, BOOTEN = 0인 경우 APP Entry와 사용자 Application코드가 실행된다. ISP 프로세스는 WizISP program을 이용해서 W7100A에 사용자 code를 write할 때 사용한다. 그리고 APP Entry는 사용자 Application code가 실행될 수 있도록 memory map switching이 일어날 수 있도록 한다.

초기 상태에서 W7100A는 아래 Figure 2.3에서와 같이 Boot ROM, APP Entry 그리고 FLASH영역의 세가지 메모리 영역을 갖는다. 하지만 그림에서 알 수 있듯이, Boot ROM과 APP Entry는 FLASH영역과 서로 겹치는 부분이 있다. 이 둘은 서로 0x0000 ~ 0x07FF와 0xFFF7 ~ 0xFFFF에서 FLASH영역과 겹친다. 그래서 W7100A는 초기화 시 Boot ROM과 APP Entry영역은 code메모리로 FLASH영역은 data메모리로 설정된다.

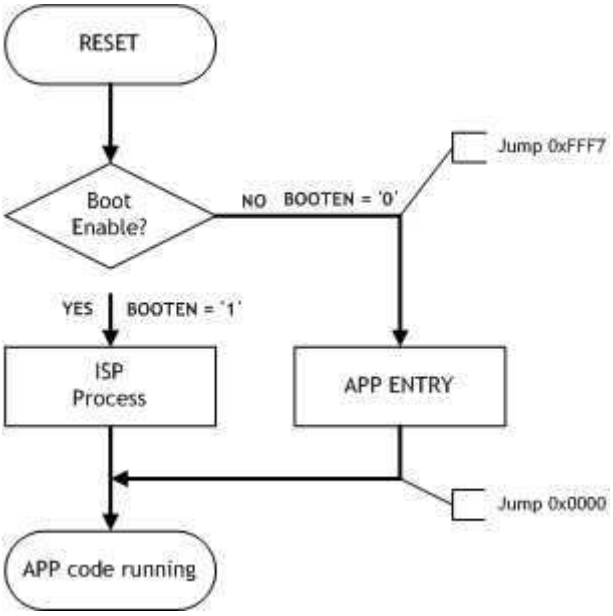


Figure 2.2. Boot Sequence Flowchart

FLASH영역을 data memory로 설정했기 때문에 사용자 application code를 FLASH에 write할 수 있다. 하지만 이 상태에서는 FLASH를 code메모리로 사용할 수 없다. FLASH를 코드메모리로 사용하기 위해서는 memory map switching이 필요하다. 이 때문에 사용자는 부팅 시 BOOTEN = 0으로 선택하여 APP모드를 선택한다. 그러면 Boot ROM 코드는 APP Entry로 즉시 jump한다. 그 후 APP Entry는 Boot ROM을 un-map하고 FLASH를 map시켜서 Code FLASH로 사용할 수 있도록 한다. code memory map switching이 끝나면 APP Entry는 Code FLASH의 시작 주소인 0x0000으로 jump한다. 이 과정은 Figure 2.3에 나타내었다.

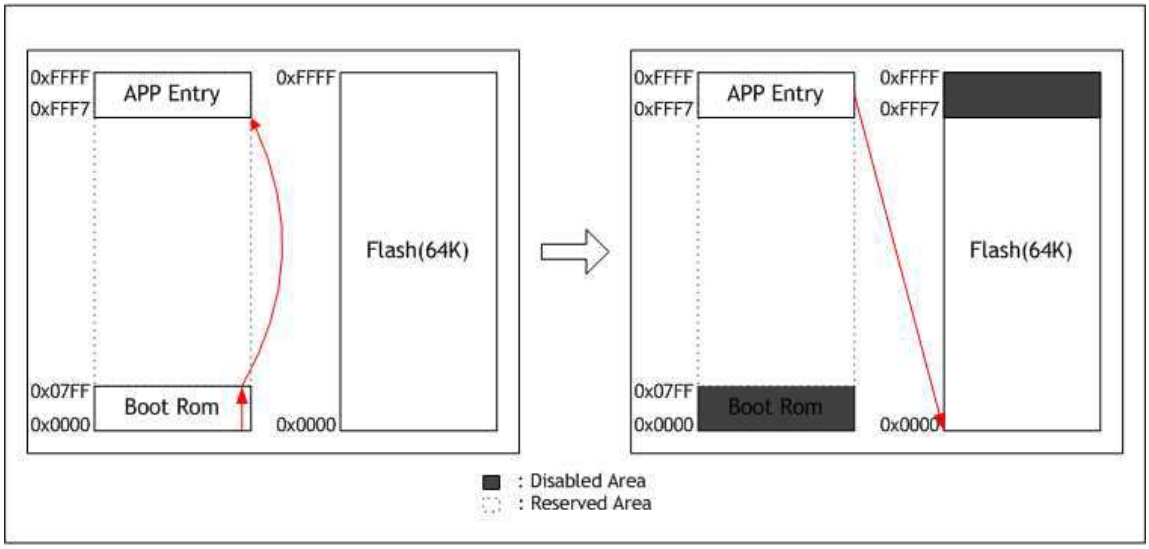


Figure 2.3 APP Entry Process

하지만 여전히 FLASH와 APP Entry는 서로 겹치는 address영역을 가지므로, FLASH 64KB를 전부 사용하기 위해서는 APP Entry를 un-map해야 한다. 이를 위해 startup code에서 WCONF(0xFF)의 RB비트를 '0'으로 설정해줘야 한다. 그러면 Figure 2.4와 같이 APP Entry는 un-map된다.

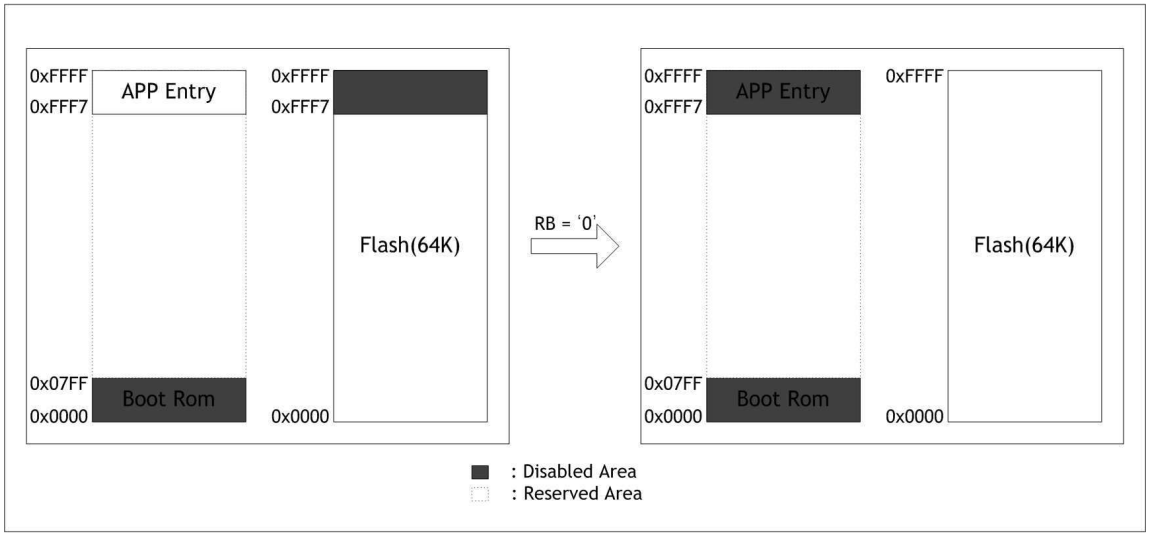


Figure 2.4 Changing the code memory Status at **RB** = '0'

WCONF (0xFF)

7	6	5	4	3	2	1	0	Reset
RB	ISPEN	EM2	EM1	EM0	RESERVED	FB	BE	0x00

Code FLASH가 0xFFF7이상의 주소를 가지게 된다면, 반드시 아래 코드를 startup code에 삽입해야 한다. 이 방법을 이용하면, W7100A는 system reset후에 즉시 APP Entry address를 disable할 것이다.

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

BOOTEN핀을 '0'으로 설정하고 Startup code에서 WCONF register의 RB비트를 clear하면 내부 Code FLASH memory 64KB를 모두 code memory로 사용할 수 있다.

2.1.1 Code Memory Wait States

Code memory wait state는 내부 WTST (0x92) register에 의해 설정되고 wait state의 종류도 WTST register의 값에 의해 결정된다. 자세한 내용은 section 2.5.10 'New & Extended SFR'을 참고하기 바란다.

2.2 Data Memory

W7100A는 내부에 64KB RAM, 64KB TCPIPCore, 255Byte Data FLASH를 포함하고 있다. Data FLASH는 사용자의 IP, MAC, subnet mask, port번호 등을 저장하는데 사용할 수 있다. 또 한 W7100A는 외부메모리를 최대 16Mbyte까지 확장할 수 있다. 이 memory들은 MOVX 명령으로만 access할 수 있다. Figure 2.5는 W7100A의 'Data Memory map'을 나타낸다.

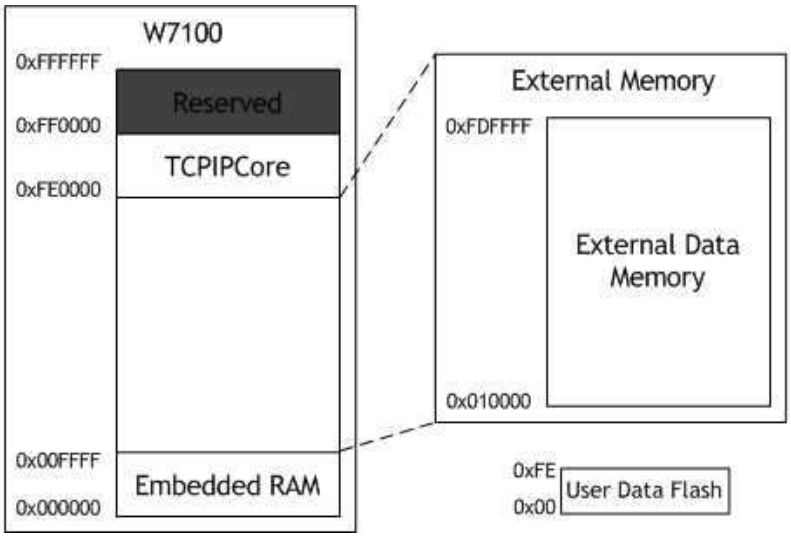


Figure 2.5 "Data Memory" Map

2.2.1 Data Memory Wait States

Data memory wait state는 CKCON (0x8E) register에 의해 설정된다. CKCON register에 설정된 값에 따라 wait state 수가 결정된다. section 2.5.10 'New & Extended SFR'을 참조하기 바란다.

2.3 External Data Memory Access

외부 메모리를 사용할 때, 외부 address, data 핀의 access mode는 크게 2가지 방법이 있다. 첫 번째는 standard 8051에서 사용하는 방법으로 address line에 latch를 사용하는 방법이고, 두 번째는 address line에 direct로 모든 라인을 연결하는 방법이다. 물론 address와 data 핀은 모두 GPIO(General Purpose I/O)의 용도로 사용이 가능하다. 외부메모리 accessing speed에 대한 정보는 section 10 'Electrical specification'을 참조하기 바란다.

아래 테이블에 EM[2:0]설정에 따른 external data memory access mode들을 정리하였다.

Table 2.1 External memory access mode

Mode	EM[2:0]	P0[7:0]	P1[7:0]	P2[7:0]	P3[7:0]
Standard 1	001	Addr[7:0]/Data[7:0]	GPIO	Addr[15:8]	GPIO
Standard 2	011	Addr[7:0]/Data[7:0]	GPIO	Addr[15:8]	Addr[23:16]
Direct 1	101	Data[7:0]	Addr[7:0]	Addr[15:8]	GPIO
Direct 2	111	Data[7:0]	Addr[7:0]	Addr[15:8]	Addr[23:16]

2.3.1 Standard 8051 Interface

이 방법은 일반적인 8051의 외부 인터페이스와 동일하다. Port0의 address와 data신호를 구분하기 위해 ALE (Address Latch Enable) pin을 이용하여 latch enable 신호를 제어한다. ALE신호가 유지되는 시간은 ALECON register를 통해 제어할 수 있으며, 보다 자세한 내용은 section 2.5.10 'New & Extended SFR'을 참조하기 바란다.

SFR register인 WCONF(0xFF)에 위치한 EM[2:0](External Memory Mode)의 setting에 따라 접근할 수 있는 address의 범위가 두 가지로 나뉜다. 먼저 EM[2:0]을 "001"로 setting하게 되면 port0을 address/data bus로 사용하고, port2를 상위 address(addr[15:8])로 사용하므로 총 16-bit address를 사용하게 된다. Address/data line으로 사용되지 않는 port1, port3은 GPIO로 사용된다.

> Note :
> Standard 8051 interface를 이용한 외부 메모리 인터페이스는 WR / RD 신호 enable 전에 ALE 신호가 발생되어야 하나, 현재 WR/RD 신호가 enable 된 이후에 ALE 신호가 발생하는 Erratum이 있다. 때문에 WR/RD를 ALE와 OR하여 외부 메모리의 nWR, nOE로 연결하면 문제가 해결된다. 더 자세한 내용은 'W7100A Errata sheet - Erratum3'을 참조하라

구성은 아래의 그림과 같다.

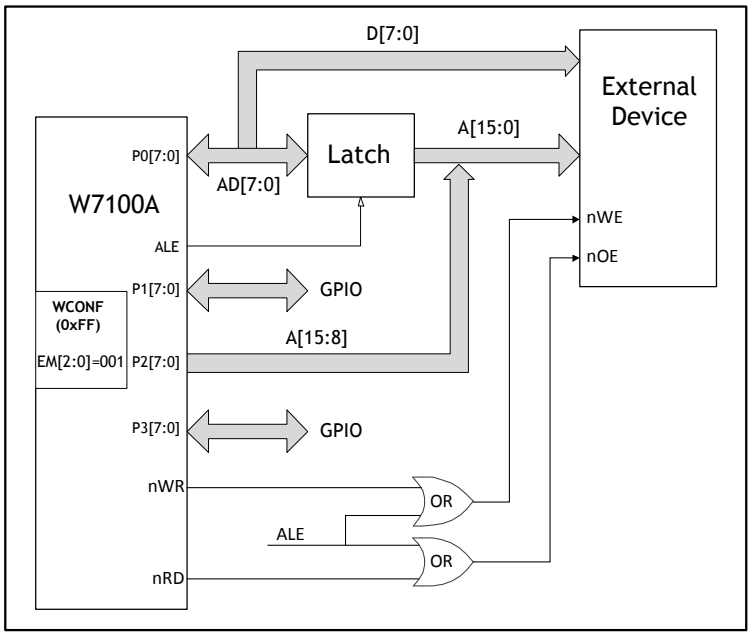


Figure 2.6 Standard 8051 External Pin Access Mode (EM[2:0] = "001")

다음으로 EM[2:0]을 “011”로 설정하게 되면 port0은 address/data bus로 사용하게 되고, port2는 상위 address(addr[15:0])로 사용된다. 그리고 port3을 최상위 address(addr[23:16])로 사용할 수 있어 address의 범위가 24-bit로 늘어난다. 마지막으로 port1은 GPIO로 사용이 가능하다. 자세한 구성은 아래의 블록도와 같다.

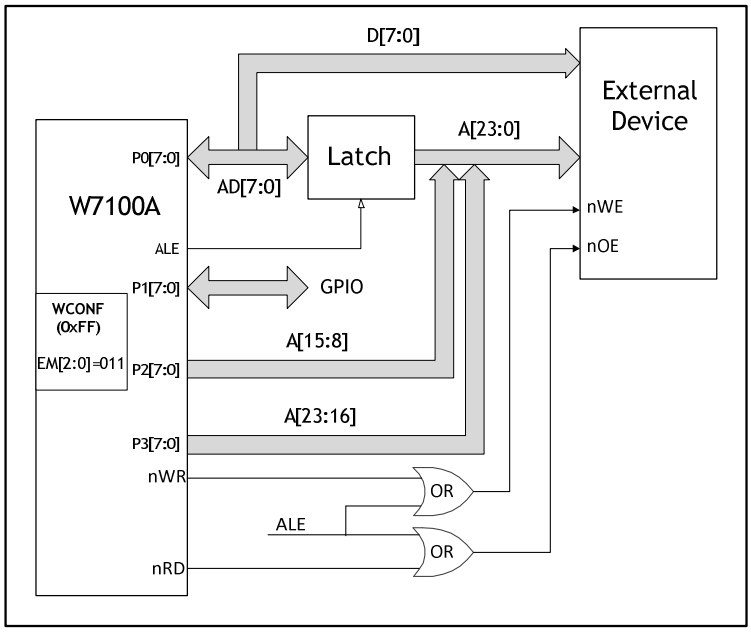


Figure 2.7 Standard 8051 External Pin Access Mode (EM[2:0] = “011”)

2.3.2 Direct Interface

이 방법은 standard 8051의 외부 연결 방법과는 다르게 address와 data 라인을 직접 연결하여 사용하는 것이다. 먼저 EM[2:0]을 “101”로 setting하게 되면, port0을 data line (data[7:0])으로 사용하게 되며, port1을 하위 address(addr[7:0])으로 사용하고, port2를 상위 address(addr[15:8])로 사용하게 된다. 그리고 남아있는 port3은 GPIO로 사용이 가능하다. 이 방법을 사용하게 되면 latch 없이 address/data 라인의 연결이 가능하다. 자세한 구성은 아래의 블록도와 같다.

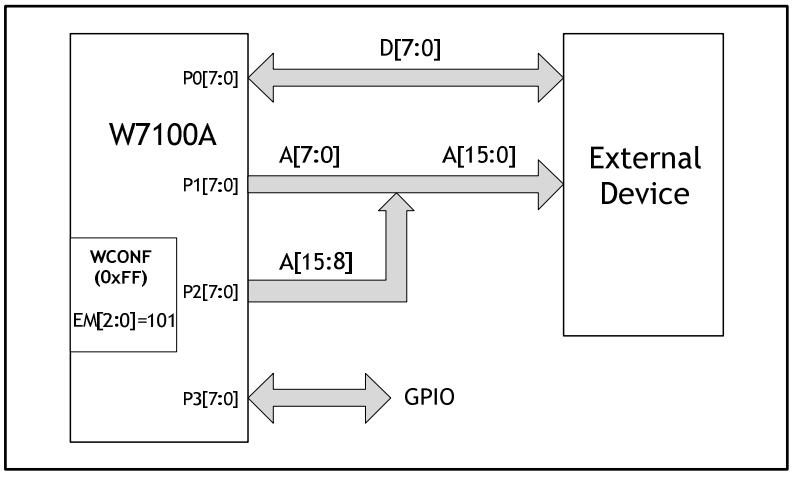


Figure 2.8 Direct 8051 External Pin Access Mode (EM[2:0] = “101”)

다음으로 EM[2:0]을 “111”로 setting하게 되면, port0, port1, port2의 구성은 위의 구성과 같고, port3을 최상위 address(addr[23:16])로 사용할 수 있다. 자세한 구성은 아래의 블록도와 같다.

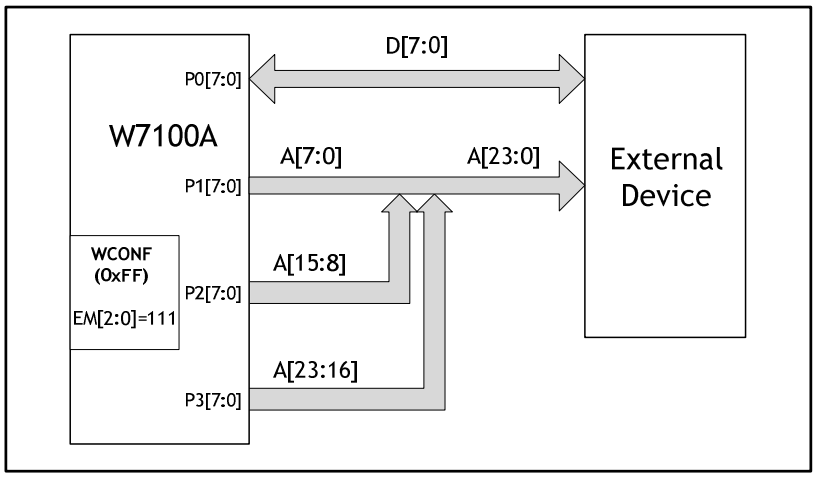


Figure 2.9 Direct 8051 External Pin Access Mode (EM[2:0] = “111”)

2.4 Internal Data Memory and SFR

아래 그림은 내부 memory와 Special Function Register (SFR) map을 보여준다.

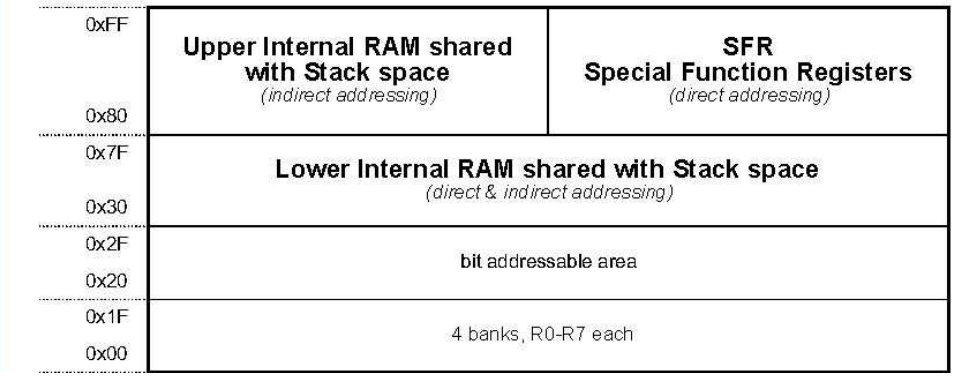
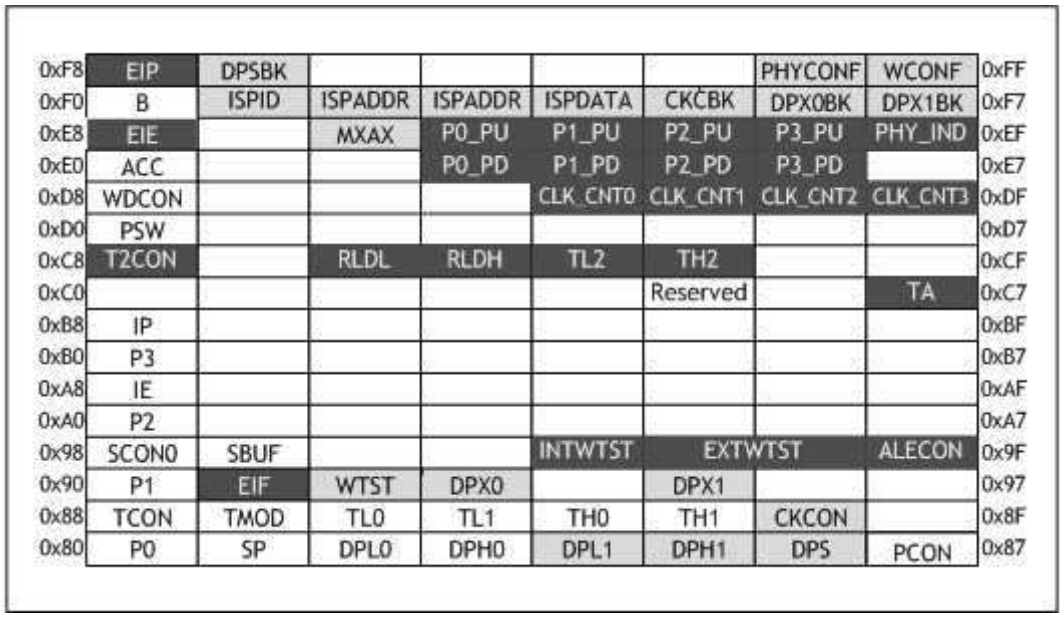


Figure 2.10 Internal Memory Map

W7100A의 내부 RAM은 0x00 ~ 0x1F까지 8개의 register들로 구성된 4개의 register bank들을 포함하고 있고, 0x20부터 128bits (16bytes)의 bit-addressable 부분과 208bytes의 scratchpad 부분으로 구성되어 있다. Indirect addressing mode로 0x80부터 0xFF까지 addressing하면, 최상위부터 128 Bytes 는 내부 memory로 accessing된다. 하지만 direct addressing mode로 0x80부터 0xFF까지 addressing하면 이 부분은 SFR memory로 accessing된다.



0xF8	EIP	DPSBK					PHYCONF	WCONF	0xFF
0xF0	B	ISPID	ISPADDR	ISPADDR	ISPDATA	CKCBK	DPX0BK	DPX1BK	0xF7
0xE8	EIE		MXAX	P0_PU	P1_PU	P2_PU	P3_PU	PHY_IND	0xEF
0xE0	ACC			P0_PD	P1_PD	P2_PD	P3_PD		0xE7
0xD8	WDCON				CLK_CNT0	CLK_CNT1	CLK_CNT2	CLK_CNT3	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	SCON0	SBUF			INTWTST	EXTWTST		ALECON	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 2.11 SFR Memory Map

New SFR – New additional SFR

Extended SFR – Standard 8051에서 확장된 SFR

Standard – Standard 8051 SFR

Figure 2.11의 표에서 왼쪽 행 0혹은 8로 끝나는 SFR은 bit addressable register이다.

2.5 SFR definition

이 section에서는 W7100A의 SFR과 그 기능들에 대해 설명한다. 더 자세한 사항은 section 2.5.11 'Peripheral SFR'을 참고하기 바란다.

2.5.1 Program Code Memory Write Enable Bit

PCON register 내부에 Program Write Enable (PWE) bit는 MOVX 명령이 수행되는 동안 Program Write signal을 제어한다. PWE bit를 '1'로 설정하면 'MOVX @DPTR, A' 명령은 accumulator register의 데이터를 DPTR (active DPH:DPL) register를 통해 설정된 code memory address에 write한다.

'MOVX @Rx, A' 명령은 accumulator register의 데이터를 P2 (bits 15:8) register와 RX(bits 7:0) register를 통해 설정된 code memory address에 write한다.

PCON (0x87)

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

Figure 2.12 PWE bit of PCON Register

Note: 1. PCON.2 ~ PCON.0 bit는 reserved이며 반드시 0으로 설정해야 한다.

2.5.2 Program Code Memory Wait States Register

Wait state register는 code memory access time을 설정하는데 쓰인다.

WTST (0x92)

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

Figure 2.13 Code memory Wait States Register

Note: 1. 위 bit들은 오직 program fetch나 MOVC명령에 영향을 준다.
Code memory write는 MOVX명령에 의해 수행되므로, CKCON register는 CODE-WR pulse width를 제어한다.
2. Read cycle은 최소 4 clock에서 최대 8 clock까지 주기를 갖는다.

Table 2.2 WTST Register Values

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

명령어 fetching을 수행하는 도중에는, code memory는 오직 MOVC명령에 의해서만 access될 수 있다. Code memory는 최소 4 wait states로 read할 수 있다. 그 Timing 다이어그램을 아래 그림에 나타내었다.

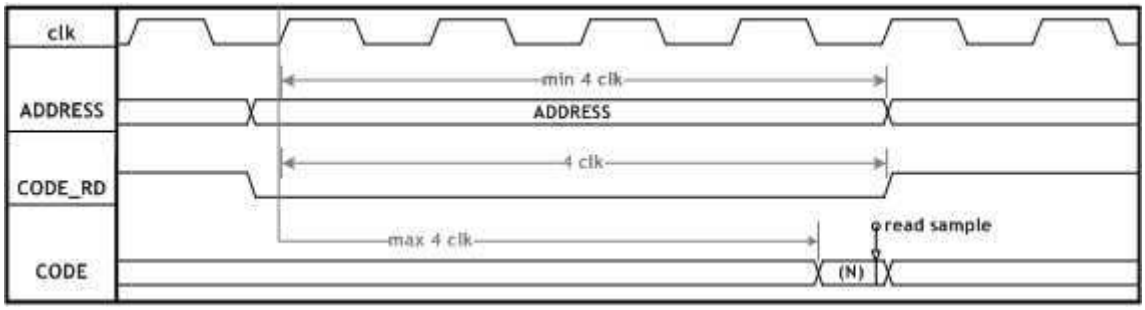


Figure 2.14 Waveform for code memory Synchronous Read Cycle with Minimal Wait States (WTST = '4')

Note: 1. clk – 시스템 clock 주파수(88.4736 MHz)
2. ADDRESS -실제 수정된 program byte 의 주소
3. CODE_RD – Code memory의 read신호
4. CODE – Data write to the actual modified program byte

Code memory는 최소 4 wait states의 MOVX명령으로 write할 수 있다. W7100A core는 wait state를 통해 고속과 저속 memory모두에서 잘 동작할 수 있다. 아래그림은 Timing diagram을 보여준다.

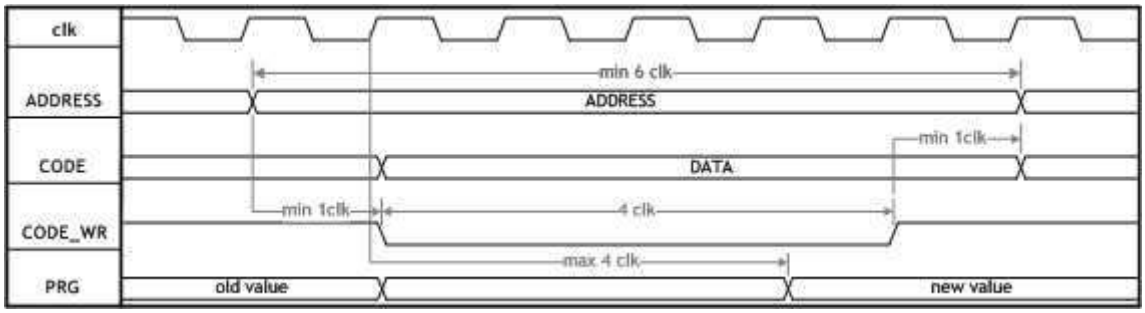


Figure 2.15 Waveform for code memory Synchronous Write Cycle with Minimal Wait States(WTST = '4')

Note: 1. clk – 시스템 clock 주파수 (88.4736 MHz)
2. ADDRESS – 실제 수정된 program byte 의 주소
3. CODE – Data write to the actual modified program byte
4. CODE_WR – Code memory의 write신호
5. PRG – Code memory 상태

2.5.3 Data Pointer Extended Registers

Data pointer extended register DPX0, DPX1, MXAX는 64KB이상의 데이터를 accessing할 때 최상위 memory address값을 갖는다. Reset후 DPX0, DPX1, MXAX는 초기값으로 0x00을 갖는다.

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 2.16 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 2.17 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 2.18 MOVX @RI Extended Register

MOVX명령이 DPTR0/DPTR1 register를 이용해서 수행되면, 최상위 address A[23:16] 는 항상 DPX0(0x93)/DPX1(0x95)에 설정된 값을 갖는다. MOVX명령이 R0혹은 R1 register를 이용해서

수행되면, 최상위 address A[23:16]는 MXAX(0xEA)의 값을 갖고 A[15:8]은 P2(0xA0)의 값을 갖는다.

2.5.4 Data Pointer Registers

Dual data pointer register들은 data block copy속도를 향상시키기 위해 만들어졌다. DPTR0와 DPTR1은 4개의 SFR영역에 위치하고 있다. Active DPTR register는 SEL bit (0x86.0)를 통해 선택할 수 있다. 만약 SEL bit가 '0'이면, DPTR0 (0x83:0x82) 가 선택된다. 반대로 SEL bit가 '1'이면 DPTR1 (0x85:0x04)가 선택된다.

DPTR0(0x83:0x82)

DPH0(0x83)								DPL0(0x82)								Reset
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	0x0000

Figure 2.19 Data Pointer Register DPTR0

DPTR1(0x85:0x84)

DPH1(0x85)								DPL1(0x84)								Reset
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	0x0000

Figure 2.20 Data Pointer 1 Register DPTR1

DPS (0x86)

7	6	5	4	3	2	1	0	Reset
ID1	ID0	TSL	-	-	-	-	SEL	0x00

Figure 2.21 Data Pointer Select Register

Note: TSL - Toggle select enable. When TSL is set, this bit toggles the SEL bit by executing the following instructions.

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 2.3 DPTR0, DPTR1 Operations

ID1	ID0	SEL = 0	SEL = 1
0	0	INC DPTR	INC DPTR1
0	1	DEC DPTR	INC DPTR1

1	0	INC DPTR	DEC DPTR1
1	1	DEC DPTR	DEC DPTR1

선택된 data pointer register는 다음 명령을 수행하는데 사용한다.

MOVX @DPTR, A

MOVX A, @DPTR

MOVC A, @A + DPTR

JMP @A + DPTR

INC DPTR

MOV DPTR, #data16

2.5.5 Clock Control Register

Clock control register CKCON (0x8E)은 data memory read/write signal pulse width정보를 가지고 있는 MD[2:0] bit를 포함하고 있다.

CKCON (0x8E)

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

Figure 2.22 Clock Control Register – STRETCH bits

Data memory read/write signal은 MOVX명령을 수행하는 동안 활성화 된다. MD[2:0] bit는 slow RAM, LCD display와 같은 I/O device들의 통신을 제어하는데 쓰인다. Reset후 MD[2:0] bit는 0x07의 초기값을 갖는데, 이 값은 느린 device들이 제대로 동작할 수 있는 값이다. 사용자는 MD[2:0] 값을 I/O device의 속도에 따라 그 값을 변경할 수 있다. 또 한 program이 실행 중이라도 상관없이 MD[2:0]의 값을 변경할 수 있다.

Table 2.4 MD[2:0] Bit Values

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

Read/write pulse width는 최소 3 clock에서 최대 8 clock까지 설정할 수 있다.

2.5.6 Internal Memory Wait States Register

Internal Memory Wait States Register INTWTST(0x9C)는 내부에 있는 64KB Ram, TCPIPCore, 255Byte Internal Flash의 access time을 설정하는데 쓰인다.

INTWTST (0x9C)

7	6	5	4	3	2	1	0	Reset
Ram WTST		TCPIPCore WTST			Flash WTST			0xFF

Figure 2.23 Internal Memory Wait States Register

- Ram WTST : 내부의 64Kbytes Ram의 access time을 설정하는데 사용되며, 2bit로 구성되어, 0~3의 값으로 설정될 수 있다.
- TCPIPCore WTST : TCPIPCore의 access time을 설정하는데 사용되며, 3bit로 구성되어, 0~7의 값으로 설정될 수 있다.
- Flash WTST : 내부의 64KBytes / 255Bytes Flash의 access time을 설정하는데 사용되며, 3bit로 구성되어, 0~7의 값으로 설정될 수 있다.

Internal ram WTST는 아래의 표와 같이 access time이 설정된다.

Table 2.5 Ram WTST Bit Values

WTST	Pulse Width[clock]
3	5
2	4
1	3
0	2

TCPIPCore, Internal flash WTST는 아래의 표와 같이 access time이 설정된다.

Table 2.6 TCPIPCore / Flash WTST Bit Values

WTST	Pulse Width[clock]
7	10
6	9
5	8
4	7
3	6
2	5
1	4
0	3

2.5.7 Address Latch Enable Register

ALECON이 '0'으로 설정되면, ALE가 1 clock후에 바로 '1'에서 '0'으로 천이된다. ALECON이 'n'으로 설정되면 ALE 신호는 1+n clock을 유지한 다음 '0'으로 천이된다.

ALE maintain duration = ALECON + 1 clock

ALECON의 초기값은 0xFF이다. 따라서 사용자는 외부 장치의 속도에 맞춰 ALECON의 값을 설정 후 사용하면 된다.

ALECON (0x9F)

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

Figure 2.24 Address Latch Enable Control register

2.5.8 External Memory Wait States Register

EXTWTST(0x9D, 0x9E)는 외부 메모리 Access속도를 제어하기 위해 사용한다. 총 16bit를 이용해서 0 ~ 65535까지 값을 설정할 수 있다.

EXTWTST0 (0x9D)

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

Figure 2.25 First Byte of Internal Memory Wait States Register

EXTWTST1 (0x9E)

7	6	5	4	3	2	1	0	Reset
EW.15	EW.14	EW.13	EW.12	EW.11	EW.10	EW.9	EW.8	0xFF

Figure 2.26 Second Byte of Internal Memory Wait States Register

2.5.9 Stack Pointer

W7100A는 내부 RAM공간에 8-bit stack pointer SP(0x81)를 가지고 있다.

SP (0x81)

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

Figure 2.27 Stack Pointer Register

이 pointer는 data가 저장되기 전 PUSH나 CALL명령에 의해 증가하고, 반대로 데이터가 pop되기 전 POP, RET, RETI명령에 의해 감소한다. 즉, stack pointer는 항상 마지막 유효 stack byte를 가리키고 있다.

2.5.10 New & Extended SFR

PHY_IND(0xEF) : PHY의 현재 상태를 알려주는 SFR이다.

PHY_IND (0xEF)

7	6	5	4	3	2	1	0	Reset
					FDX	SPD	LINK	0x00

Figure 2.28 PHY Status Register

Note: FDX : 0 - Full duplex / 1 - Half duplex

SPD : 0 - 100Mbps / 1 - 10Mbps

LINK : 0 - The link is down / 1 - The link is up

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.

PHYCONF (0xFE): W7100A PHY operation mode, reset, power down configuration register

PHYCONF (0xFE)

7	6	5	4	3	2	1	0	Reset
-	-	PHY_RSTn	PHY_PWDN	MODE_EN	MODE2	MODE1	MODE0	0x00

Figure 2.29 Internal PHY Configuration Register

Note:

PHY_RSTn: W7100A의 내부 Ethernet PHY를 reset한다. 이 bit를 1로 set하고 reset timing을 참고하여 다시 0으로 clear한다. Reset timing에 대한 보다 자세한 내용은 section 10 'Electrical Specification' 을 참조하기 바란다.

PHY_PWDN: W7100A power down 모드를 설정한다. 1일 때 power down 모드로 동작하고 0일 때 normal모드로 동작한다.

MODE_EN: 1이면 MODE2 ~ 0 bit를 사용하여 W7100A의 동작 모드를 설정한다. **64pin package에서는 반드시 이 bit를 설정한 다음 MODE2 ~ 0bit를 설정하여 동작모드를 설정해야만 한다.**

MODE2 ~ 0: PHY모드 설정 bit, 설정 값들에 대한 정보는 1.4.2 'Pin Description'의 PM2 ~ 0핀 설정 값과 동일하다.

\- : Reserved, must be set to '0'

ex> usage of mode selection using MODE2 ~ 0

```
PHYCONF |= 0x08;   // MODE_EN bit enable
PHYCONF &= 0xF8;   // MODE2 ~ 0 value is 0 (normal mode; auto configuration mode)
PHYCONF |= 0x20;   // Set the PHY_RSTn bit (reset bit)
```

```
Delay();          //Delay for reset timing(refer to the section 10 'Electrical Specification')
PHYCONF &= ~(0x20); // Clear the PHY_RSTn bit
```

WCONF(0xFF): W7100A configuration register

WCONF (0xFF)

7	6	5	4	3	2	1	0	Reset
RB	ISPEN	EM2	EM1	EM0	Reserved	FB	BE	0x00

Figure 2.30 W7100A Configuration Register

Note: RB : 1 - ISP 동작이 끝나면(APP Entry(0xFFF7 ~ 0xFFFF) RD/WR Enable) 재부팅함
0 - ISP동작이 끝나도 재부팅 하지 않음
ISPEN : 0 - ISP enable, 1 - ISP disable
EM[2:0] : 외부 메모리 모드 설정, section 2.3 'External Data Memory Access' 참조
FB : FLASH Busy Flag for ISP. Read only.
BE : Boot Enable (1 - Boot Running / 0 - Apps Running). Read only.

CLKCNT0(0xDC): W7100A core clock count register bit0 ~ 7.

CLK_CNT0 (0xDC)

7	6	5	4	3	2	1	0	Reset
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	0x00

Figure 2.31 Core clock count register

Note: CLK_CNT는 32bit 값으로 매 core clock마다 증가한다. 이 SFR은 core clock을 count 하는데 사용하거나 시간을 재는데 사용할 수 있다.
ex> 1 second = about 88000000 clock count (icore clock is about 88MHz)

CLKCNT0(0xDD): W7100A core clock count register bit8 ~ 15.

CLK_CNT1 (0xDD)

7	6	5	4	3	2	1	0	Reset
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	0x00

Figure 2.32 Core clock count register

CLKCNT0(0xDE): W7100A core clock count register bit16 ~ 23.

CLK_CNT2 (0xDE)

7	6	5	4	3	2	1	0	Reset
Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	0x00

Figure 2.33 Core clock count register

CLKCNT0(0xDF): W7100A core clock count register bit24 ~ 31.

CLK_CNT3 (0xDF)

7	6	5	4	3	2	1	0	Reset
Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	0x00

Figure 2.34 Core clock count register

2.5.11 Peripheral Registers

P0, P1, P2, P3 : Port register. 자세한 사항은 section 4 'I/O Ports'를 참고하기 바란다.

TCON(0x88) : Timer 0, 1 configuration register. 자세한 사항은 section 5.1 'Timer 0, 1'을 참고하기 바란다.

TMOD(0x89) : Timer 0, 1 control mode register. 자세한 사항은 section 5.1 'Timer 0, 1'을 참고하기 바란다.

TH0(0x8C), TL0(0x8A) : Timer 0의 Counter register. 자세한 사항은 section 5.1 'Timer 0, 1'을 참고하기 바란다.

TH1(0x8D), TL1(0x8B) : Timer 1의 Counter register. 자세한 사항은 section 5.1 'Timer 0, 1'을 참고하기 바란다.

SCON(0x98) : UART Configuration Register. 자세한 사항은 section 6 'UART'를 참고하기 바란다.

SBUF(0x99) : UART Buffer Register. 자세한 사항은 section 6 'UART'를 참고하기 바란다.

IE(0xA8) : UART Bits in Interrupt Enable Register. 자세한 사항은 section 6 'UART'를 참고하기 바란다.

IP(0xB8) : UART Bits in Interrupt Priority Register. 자세한 사항은 section 6 'UART'를 참고하기 바란다.

TA(0xC7) : Timed Access Register. 자세한 사항은 section 7 'Watchdog Timer'를 참고하기 바란다.

T2CON(0xC8) : Timer 2 Configuration Register. 자세한 사항은 section 5.2 'Timer 2'를 참고하기 바란다.

RLDH(0xCB), RLDL(0xCA) : Capture Registers of Timer 2. 자세한 사항은 section 5.2 'Timer 2'를 참고하기 바란다.

TH2(0xCD), TL2(0xCC) : Counter Register of Timer 2. 자세한 사항은 section 5.2 'Timer 2'를 참고하기 바란다.

PSW(0xD0) : Program Status Word Register. 자세한 사항은 section 1.3.1. 'ALU'를 참고하기 바란다.

WDCON(0xD8) : Watchdog Control Register. 자세한 사항은 section 7 'Watchdog Timer'를 참고하기 바란다.

3 Interrupt

Interrupt pin 기능들을 아래 테이블에 정리하였다. 모든 pin들은 단방향성 (unidirectional) 이고, tri-state 신호는 없다.

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)

W7100A core는 두 level의 interrupt priority control을 갖고 있다. 각각의 외부 interrupt는 IP (0xB8)와 EIP(0xF8) register를 설정하거나 clear함으로써 high혹은 low level priority group 으로 설정된다. 외부 interrupt pin들은 falling edge signal에 의해 activate된다. Interrupt request들은 system clock의 rising edge에서 sampling한다.

Table 3.2 W7100A 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
TI0 & RI0	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

각 interrupt 벡터는 IE (0xA8) 과 EIE (0xE8) register에서 해당 bit를 설정함으로써 각각 enable혹은 disable될 수 있다. IE register는 global interrupt system disable(0)/enable(1) bit

EA를 포함하고 있다.

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

Interrupt를 enable하는 위의 모든 bit들은 software로 clear혹은 set할 수 있고, hardware로도 같은 결과를 얻을 수 있다. 즉, interrupt들은 software를 통해 생성될 수 있고 취소될 수도 있다 (단, IE0과 IE1 flag는 제외). 만약 외부 interrupt0, interrupt1이 level-activated로 program되어있다면, IE0과 IE1은 각각 외부 source pin nINT0/FA6과 nINT1/FA7을 통해 제어할 수 있다.

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 Timer 0, 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 flag는 processor가 interrupt routine을 수행할 때 자동으로 clear된다.

IE1 – INT1 interrupt flag는 processor가 interrupt routine을 수행할 때 자동으로 clear된다.

TF0 - Timer 0 interrupt (overflow) flag는 processor가 interrupt routine을 수행할 때 자동으로 clear된다.

TF1 – Timer 1 interrupt (overflow) flag는 processor가 interrupt routine을 수행할 때 자동으로 clear된다.

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 external INT2 Interrupt

EINT3 - Enable external 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 - TCPIPCore 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, Watchdog interrupt enable bit (EIE. 4). WDIF bit는 반드시 interrupt service routine을 빠져 나오기 전에 software로 clear해 주어야 한다. WDIF를 enable하기 위해 software를 사용하면, Watchdog interrupt가 발생한다. 즉, Enabled software-set WDIF에 의해 Watchdog interrupt가 발생할 수 있다. Timed access register procedure를 이용해서 이 bit를 수정할 수 있다.

4 I/O Ports

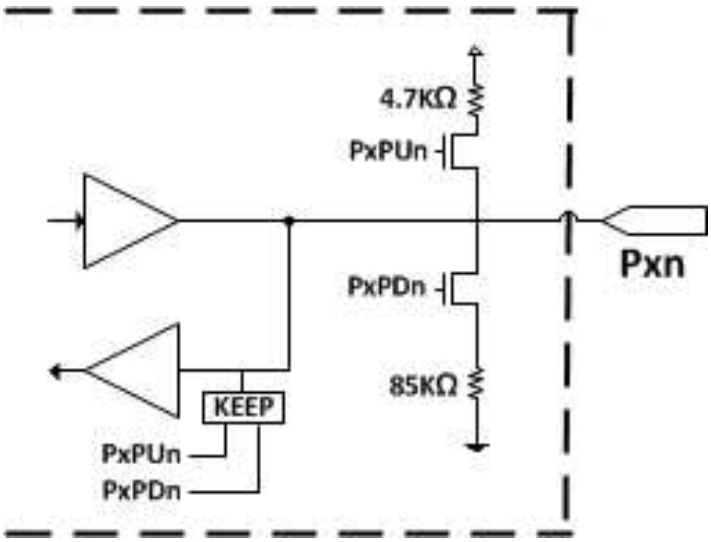


Figure 4.1 Port0 Pull-down register

W7100A의 GPIO핀은 SFR설정에 따라 Pull-up, Pull-down 그리고 Keep 세가지 상태로 설정할 수 있다. Keep상태는 Pull-up 레지스터와 Pull-down 레지스터를 동시에 set할 경우 설정되며 입출력에 변화가 없을 경우 이전 상태 값을 유지하는 특성이 있다.

I/O port pin기능들을 아래 테이블에 나타내었다.

Table 4.1 I/O Ports Pin Description

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

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 4.2 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 4.3 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 4.4 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 4.5 Port3 Register

I/O port에서 Read와 write access는 해당 SFR: P0 (0x80), P1 (0x90), P2 (0xA0), P3 (0xB0)을 통해 이루어진다. 몇몇 port-reading 명령어는 다른 port pin read 동작이 진행되는 동안 data register로부터 read한다. 'Read-Modify-Write' 명령어들은 다음 테이블처럼 직접 data register들로 전달된다.

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

모든 다른 명령어들은 port pin들을 통해 각각 따로 (exclusively) read 한다. 모든 port들은 GPIO (General Purpose Input Output) 로 사용할 수 있다. 아래 Figure 4.5에 W7100A의 GPIO를 나타내었다. GPIO의 output driving voltage는 Px_PD, Px_PU SFR의 설정 값에 따라 0V혹은 3.3V의 값을 갖거나 이전 값을 유지한다.

Px_PU	Px_PD	Status
0	0	-
0	1	Pull-down
1	0	Pull-up
1	1	Keep

P0_PD(0xE3) : Port0 8핀의 Pull-down을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-down이 설정된다.

P0_PD (0xE3)

7	6	5	4	3	2	1	0	Reset
Port0[7]	Port0[6]	Port0[5]	Port0[4]	Port0[3]	Port0[2]	Port0[1]	Port0[0]	0x00

Figure 4.6 Port0 Pull-down register

P1_PD(0xE4) : Port1 8핀의 Pull-down을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-down이 설정된다.

P1_PD (0xE4)

7	6	5	4	3	2	1	0	Reset
Port1[7]	Port1[6]	Port1[5]	Port1[4]	Port1[3]	Port1[2]	Port1[1]	Port1[0]	0x00

Figure 4.7 Port1 Pull-down register

P2_PD(0xE5) : Port2 8핀의 Pull-down을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-down이 설정된다.

P2_PD (0xE5)

7	6	5	4	3	2	1	0	Reset
Port2[7]	Port2[6]	Port2[5]	Port2[4]	Port2[3]	Port2[2]	Port2[1]	Port2[0]	0x00

Figure 4.8 Port2 Pull-down register

P3_PD(0xE6) : Port3 8핀의 Pull-down을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-down이 설정된다.

P3_PD (0xE6)

7	6	5	4	3	2	1	0	Reset
Port3[7]	Port3[6]	Port3[5]	Port3[4]	Port3[3]	Port3[2]	Port3[1]	Port3[0]	0x00

Figure 4.9 Port3 Pull-down register

P0_PU(0xEB) : Port0 8핀의 Pull-up을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-up이 설정된다.

P0_PU (0xEB)

7	6	5	4	3	2	1	0	Reset
Port0[7]	Port0[6]	Port0[5]	Port0[4]	Port0[3]	Port0[2]	Port0[1]	Port0[0]	0x00

Figure 4.10 Port0 Pull-up register

P1_PU(0xEC) : Port1 8핀의 Pull-up을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-up이 설정된다.

P1_PU (0xEC)

7	6	5	4	3	2	1	0	Reset
Port1[7]	Port1[6]	Port1[5]	Port1[4]	Port1[3]	Port1[2]	Port1[1]	Port1[0]	0x00

Figure 4.11 Port1 Pull-up register

P2_PU(0xED) : Port2 8핀의 Pull-up을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-up이 설정된다.

P2_PU (0xED)

7	6	5	4	3	2	1	0	Reset
Port2[7]	Port2[6]	Port2[5]	Port2[4]	Port2[3]	Port2[2]	Port2[1]	Port2[0]	0x00

Figure 4.12 Port2 Pull-up register

P3_PU(0xEE) : Port3 8핀의 Pull-up을 설정하는 SFR이다. 해당 비트가 '1'로 set되면 Pull-up이 설정된다.

P3_PU (0xEE)

7	6	5	4	3	2	1	0	Reset
Port3[7]	Port3[6]	Port3[5]	Port3[4]	Port3[3]	Port3[2]	Port3[1]	Port3[0]	0x00

Figure 4.13 Port3 Pull-up register

5 Timers

W7100A는 두 개의 16-bit timer/counter, Timer0와 Timer1을 가지고 있다. ‘timer mode’에서, timer register들은 매 12clock period마다 증가한다. ‘counter mode’에서, timer register들은 해당 입력 pin (T0, T1) 의 falling transition마다 증가하고 입력 pin들을 매 clock마다 샘플링 한다.

5.1 Timers 0, 1

5.1.1 Overview

Timer0, 1의 pin기능들을 아래 테이블에 정리하였다. 모든 pin들은 단 방향 (unidirectional) 이고 tri-state 출력 pin이나 내부신호는 없다.

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

Timer0와 Timer1은 표준 8051 timer들과 완벽히 호환된다. 각 timer는 2개의 8-bit register들 TH0 (0x8C), TL0 (0x8A), TH1 (0X8D), TL1 (0X8B) 로 구성되어있다. Timer들은 아래 테이블과 같이 4가지 모드로 동작한다.

Table 5.2 Timers 0, 1 Mode

M1	M0	Mode	Function Description
0	0	0	THx 는 32분주 prescaler (TLx 의 하위 5bit사용) 와 함께 8-bit timer/counter로 동작
0	1	1	16-bit timer/counter. THx, TLx를 모두 사용
1	0	2	TLx는 8-bit timer/counter로 동작, THx에 의해 TLx는 auto-reload 됨
1	1	3	두 개의 8-bit timer/counter로 동작, TL0는 Timer0 제어 bit에 의해 제어되고 TH0는 Timer1 제어 bit에 의해 제어됨

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 Timer 0, 1 Control Mode Register

Note: GATE - Gating control

1: GATE pin과 TRx pin 모두 ‘1’인 상태에서만 Timer x가 동작함

0: TRx pin이 '1'인 상태에서만 Timer x가 동작함

CT - Counter or timer select bit

1: Counter mode, Timer x 는 Tx pin의 신호를 counting함

0: Timer mode, 내부 clock으로 동작함

M1, M0 – Mode 선택 bits

TCON (0x88)

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

Figure 5.2 Timer 0, 1 Configuration Register

Note: TR0 - Timer 0 동작제어 bit

1: Enabled

0: Disabled

TR1 - Timer 1 동작제어 bit

1: Enabled

0: Disabled

외부 입력 pin들, GATE0와 GATE1는 pulse의 폭을 측정하기 위한 용도로 사용할 수 있다.

5.1.2 Interrupts

Timer0, 1 interrupt에 대한 bit들을 아래에 나타내었다. IE register를 이용해서 interrupt를 toggle할 수 있다. 그리고 interrupt들의 우선순위는 IP register를 이용해서 제어할 수 있다.

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 Timer 0, 1 Configuration Register

Note: TF0 – Timer0 interrupt (overflow) flag는 processor가 interrupt routine을 처리할 때 자동으로 clear된다.

TF1 – Timer1 interrupt (overflow) flag는 processor가 interrupt routine을 처리할 때 자동으로 clear된다.

Interrupt를 발생시키는 모든 bit들은 hardware와 마찬가지로 software를 이용해서 set혹은 clear할 수 있다. 즉, interrupt들은 software에 의해 발생할 수도 있고 취소될 수도 있다.

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

Timer0는 13bit register (8bit: Timer, 5bit: prescaler) 를 통해 제어할 수 있다. 모든 count (valid bits) 가 1에서 0으로 바뀌면, Timer0 interrupt flag TF0는 set된다. Timer는 TCON.4 = 1 이고 TMOD.3 = 0 혹은 GATE0 = 1 인 상태에서 시작된다. TMOD.3 = 1로 설정 함으로써 외부 입력 GATE0는 Timer0가 pulse width 측정을 할 수 있도록 한다. Mode0에서 13bit register는 TH0의 8bit와 TL0의 하위 5bit를 사용한다. TL0의 상위 3bit는 사용하지 않는다. 아래 Figure 5.6에 Timer0의 mode0구조를 나타내었다.

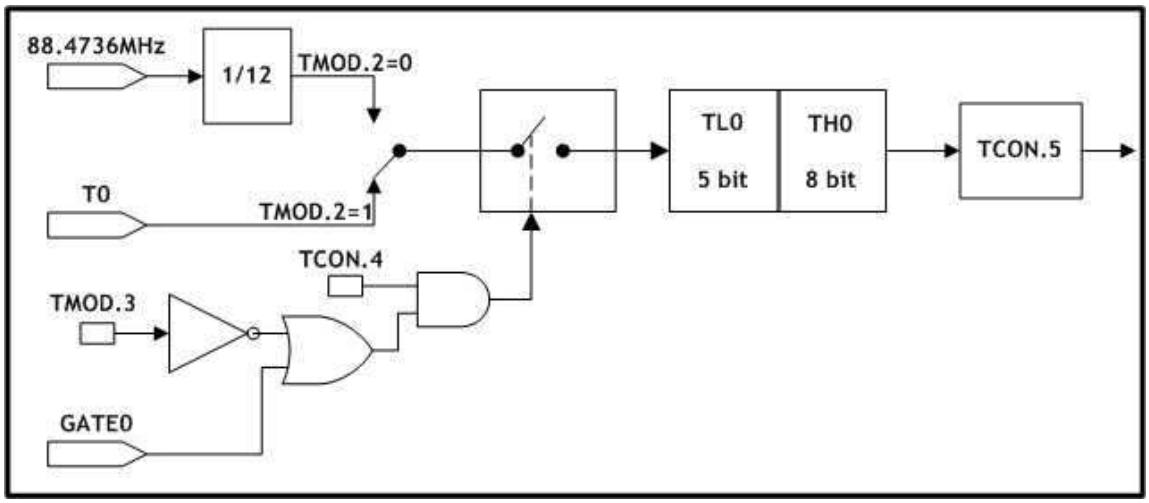


Figure 5.6 Timer Counter0, Mode0: 13-Bit Timer/Counter

5.1.4 Timer0 – Mode1

Mode1은 timer register가 16bit로 동작된다는 점을 제외하면 Mode0와 같다. 그 구조는 아래 Figure 5.7에 나타내었다.

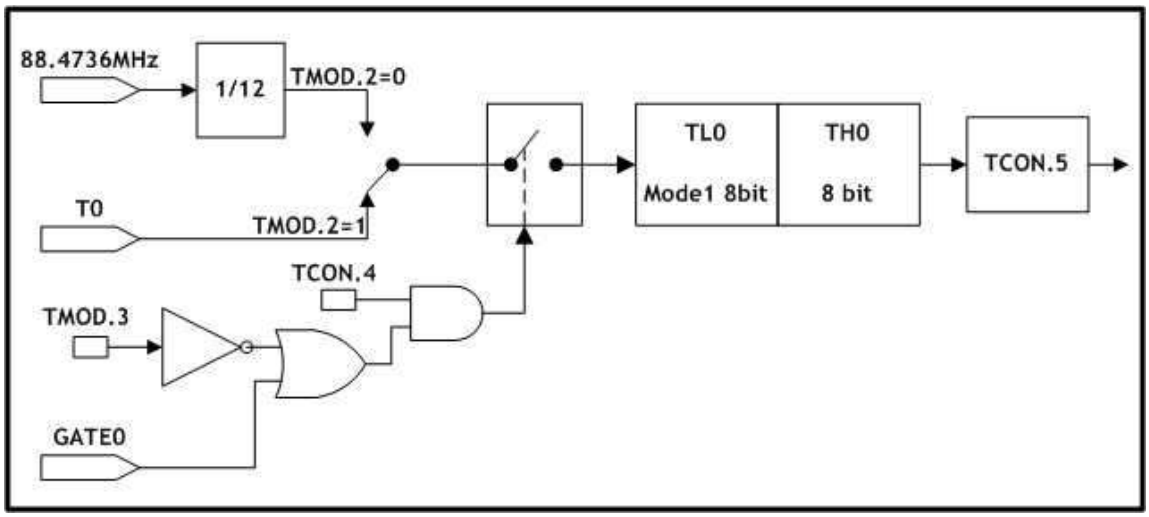


Figure 5.7 Timer/Counter0, Mode1: 16-Bit Timer/Counter

5.1.5 Timer0 – Mode2

Mode2는 TL0의 8bit를 timer/counter register로 사용한다. Overflow가 발생하면 TL0는 TH0에 저장된 값으로 자동으로 reload된다. TH0의 값은 reload된 후에도 변하지 않는다.

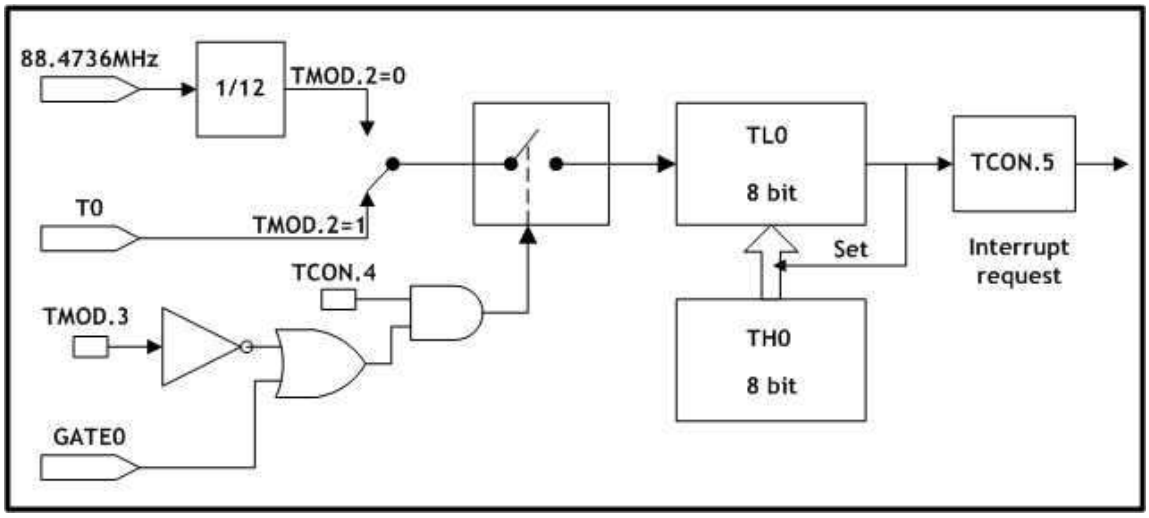


Figure 5.8 Timer/Counter0, Mode2: 8-Bit Timer/Counter with Auto-Reload

5.1.6 Timer0 – Mode3

이 모드에서는 TL0와 TH0가 두 개의 counter로 나뉜다. 다음 Figure는 Timer0의 Mode3 동작을 보여준다. TL0는 Timer0 control bit (C/T, GATE, TR0, GATE0, TF0)를 이용해 제어한다. 그리고 TH0는 Timer1의 TR1에 의해 제어되고 interrupt 발생은 TF1 (interrupt flag)를 이용

한다. Mode3은 추가로 8-bit timer/counter가 필요한 경우 사용할 수 있다. Timer0가 Mode3로 사용되고 있을 때, Timer1은 스스로 Mode3로 turn on/off할 수도 있고 그렇지 않으면, serial channel의 baud-rate generator로 사용될 수도 있다. 그리고 Timer1의 인터럽트를 사용하지 않아도 되는 application에 사용될 수도 있다.

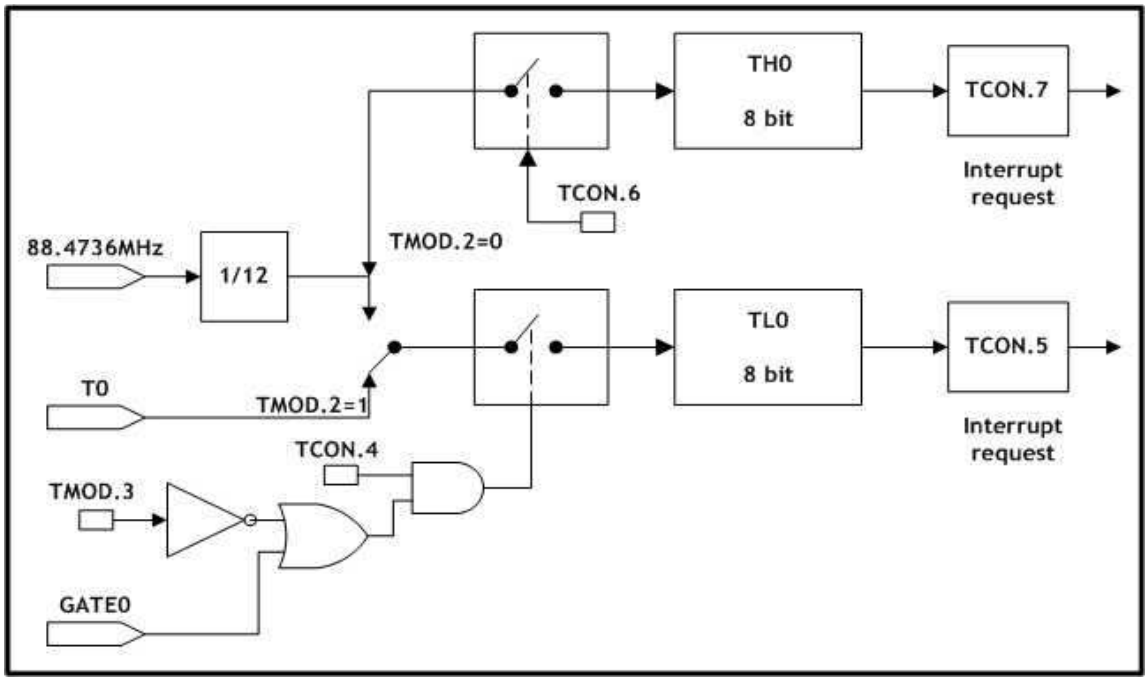


Figure 5.9 Timer/Counter0, Mode3: Two 8-Bit Timers/Counters

5.1.7 Timer1 – Mode0

Timer1은 13bit register (8bit: Timer, 5bit: prescaler) 를 통해 제어할 수 있다. 모든 count (valid bits) 가 1에서 0으로 바뀌면, Timer1 interrupt flag TF1은 set된다. Timer는 TCON.6 = 1 이고 TMOD.6 = 0 혹은 GATE1 = 1 인 상태에서 시작된다. TMOD.7 = 1로 설정 함으로써 외부 입력 GATE1은 Timer1이 pulse width 측정을 할 수 있도록 한다. Mode0에서 13bit register는 TH1의 8bit와 TL1의 하위 5bit를 사용한다. TL1의 상위 3bit는 사용하지 않는다. 아래 Figure 5.10에 Timer1의 mode0구조를 나타내었다.

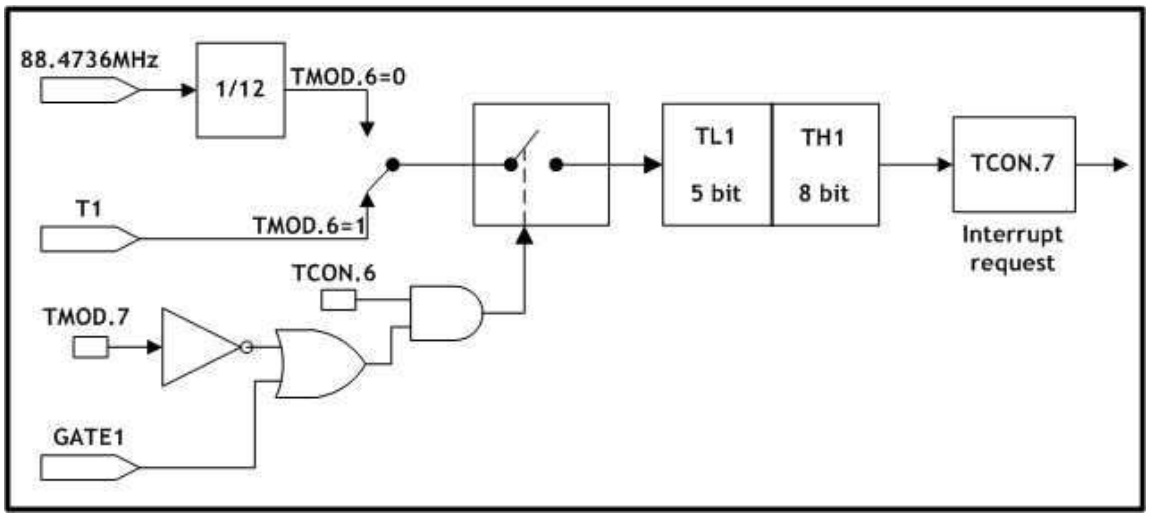


Figure 5.10 Timer/Counter1, Mode0: 13-Bit Timer/Counter

5.1.8 Timer1 – Mode1

Mode1은 timer register가 16bit로 동작된다는 점을 제외하면 Mode0와 같다. 그 구조는 아래 Figure 5.11에 나타내었다.

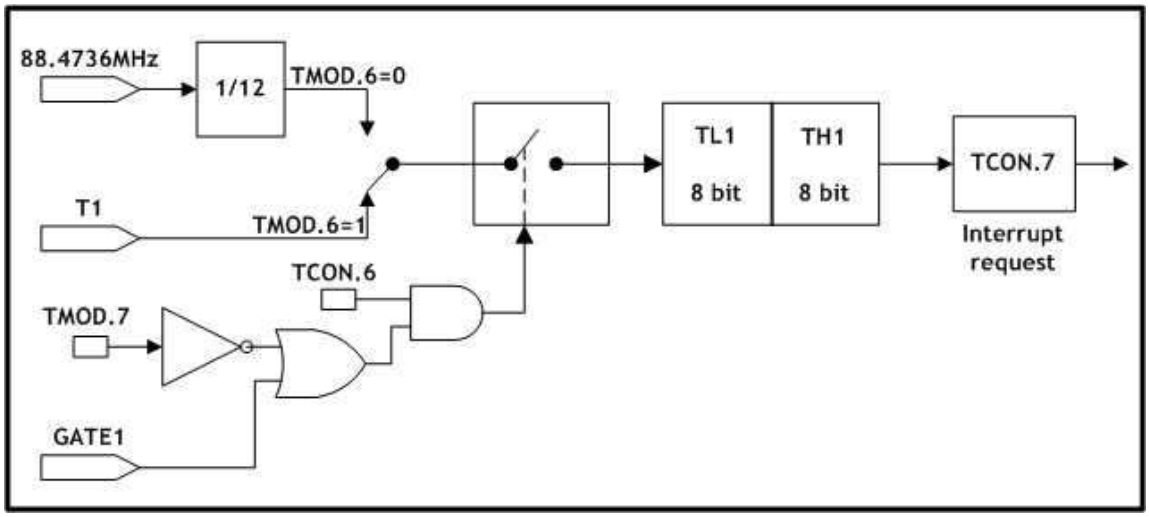


Figure 5.11 Timer/Counter1, Mode1: 16-Bit Timers/Counters

5.1.9 Timer1 – Mode2

Mode2는 TL1의 8bit를 이용해 제어한다. Overflow가 발생하면 TF1을 set하고 TL1의 8bit는 TH1에 저장된 8bit로 자동으로 reload된다. Reload후 TH1 값은 변하지 않는다.

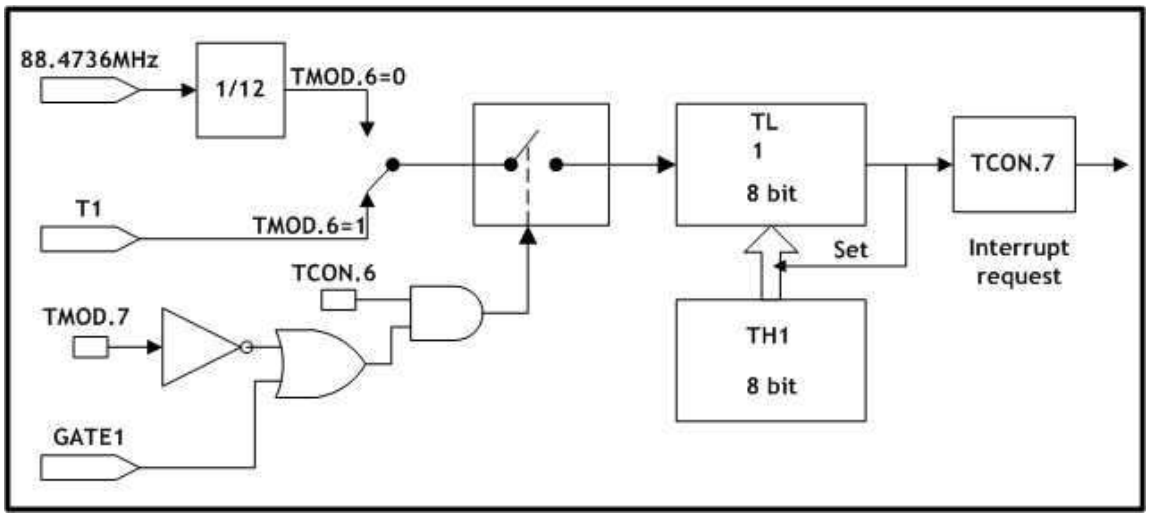


Figure 5.12 Timer/Counter1, Mode2: 8-Bit Timer/Counter with Auto-Reload

5.1.10 Timer1 – Mode3

Timer1이 Mode3으로 동작할 경우에는, 이미 Timer0에서 Mode3으로 Timer1을 사용하고 있기 때문에 TR1을 0으로 설정한 것과 같다. 자세한 동작은 5.1.6 'Timer0-Mode3'을 참고하기 바란다.

5.2 Timer2

5.2.1 Overview

Timer2의 pin기능들을 아래 테이블에 정리하였다. 모든 pin들은 단 방향 (unidirectional) 이고 tri-state 출력 pin이나 내부신호는 없다.

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

W7100A의 Timer2는 표준 8051 Timer2와 완벽하게 호환된다. Timer2는 TH2/TL2 (0xCD/0xCC) counter registers, RLDH/RLDL (0xCB, 0xCA) capture registers, T2CON (0xC8) control register 총 5개의 SFR에 의해 제어된다. Timer2는 T2CON register의 bit들을 설정하여 아래 테이블과 같이 3가지 mode로 동작한다.

Table 5.5 Timer2 Modes

RCLK,TCLK	CPRL2	TR2	Function Description
0	0	1	16-bit auto-reload mode. TF2 bit는 Timer2 overflow가 발생하면 set되고, TH2와 TL2 register는 RLDH와 RLDL에 저장된 값으로 reload된다.

0	1	1	16-bit capture mode. Timer2 overflow가 발생하면 TF2 bit가 set되고 EXEN2=1, T2EX pin이 falling edge일 때, TH2와 TL2register 값은 RLDH와 RLDL에 저장된다.
1	X	1	UART interface를 위한 Baud rate generator mode
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 – EXEN2=1, T2EX pin이 falling edge일 때 EXF2는 set된다. 반드시 software를 통해 clear해야 한다.

RCLK - Receive clock enable

0: UART receiver는 Timer1 overflow pulses에 의해 clock됨

1: UART receiver는 Timer2 overflow pulses에 의해 clock됨

TCLK - Transmit clock enable

0: UART transmitter는 Timer1 overflow pulses에 의해 clock됨

1: UART transmitter는 Timer2 overflow pulses에 의해 clock됨

EXEN2 - Enable T2EX pin functionality

0: Ignore T2EX events

1: T2EX pin이 falling edge일 때 capture혹은 reload를 허용함

TR2 - Start/Stop Timer2

0: Stop

1: Start

CT2 - Timer/Counter select

0: Internally clocked timer

1: External event counter, Clock source는 T2 pin

CPRL2 - Capture/Reload select

0: Timer2 overflow발생시 혹은 T2EX pin이 falling edge이고 EXEN2=1인 경우 자동으로 reload 발생. RCLK혹은 TCLK가 set되면, 이 bit는 무시되고 Timer2 overflow발생시에 자동으로 reload됨

1: EXEN2=1이고 T2EX pin이 falling edge일 때 capture활성화

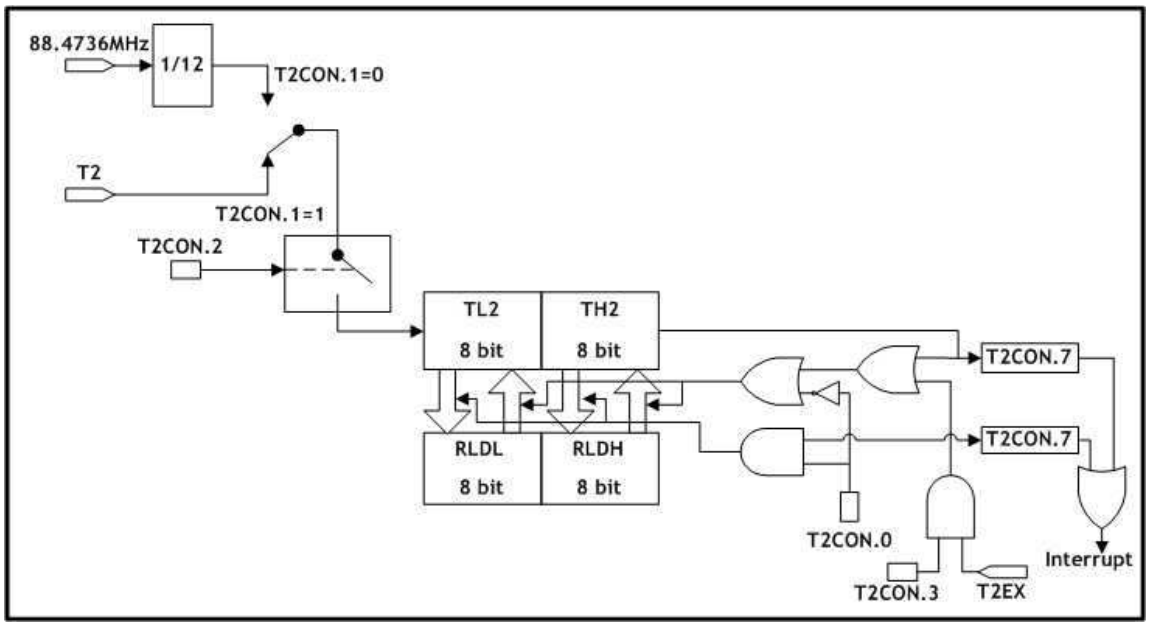


Figure 5.14 Timer/Counter2, 16-Bit Timer/Counter with Auto-Reload

5.2.2 Interrupts

Timer2의 interrupt bit들은 아래와 같다. IE register를 설정하여 interrupt를 toggle할 수 있고, interrupt 우선순위는 IP register를 통해 설정할 수 있다.

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)

7	6	5	4	3	2	1	0	Reset
-	-	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)

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

Figure 5.17 Timer2 Configuration Register – TF2

Note: TF2 – Timer2 interrupt (overflow) flag, 반드시 software로 clear해줘야 함.

이 flag는 RCLK 혹은 TCLK가 set된 경우 set되지 않는다.

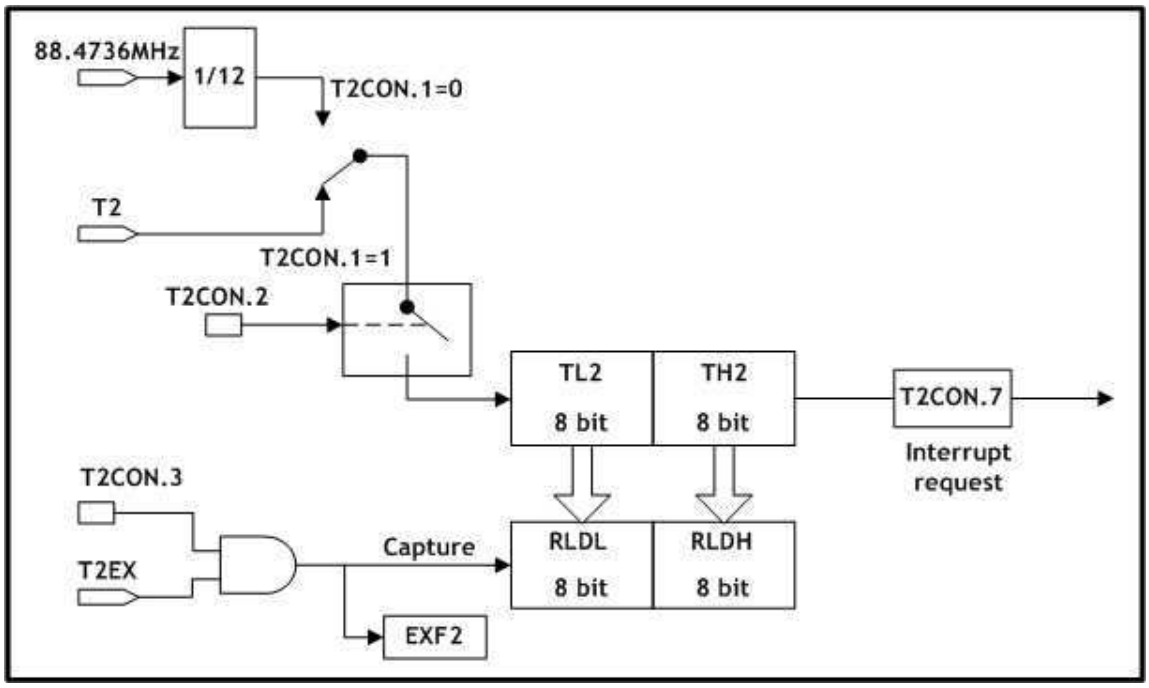


Figure 5.18 Timer/Counter2, 16-Bit Timer/Counter with Capture Mode

Interrupt를 발생과 관련된 모든 bit들은 hardware와 마찬가지로 software에 의해 clear되거나 set될 수 있다. 즉, interrupt들은 software에 의해 발생되거나 취소될 수도 있다.

Table 5.6 Timer2 Interrupt

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

EXEN2 bit가 set되어 있고 T2EX pin이 falling edge일 때 Timer2 interrupt가 발생한다. 0x2B vector를 이용해서 EXF2는 이 interrupt에 의해 set된다. 하지만 TF2 flag는 변하지 않고 유지된다.

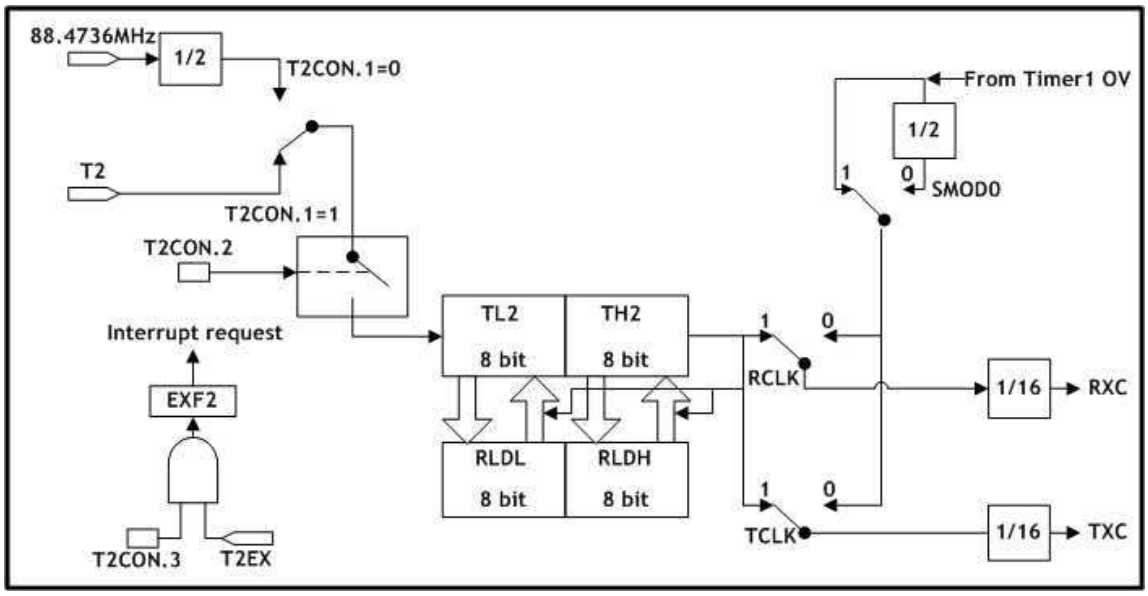


Figure 5.19 Timer2 for Baud Rate Generator Mode

6 UART

W7100A의 UART는 수신과 송신이 동시에 가능한 full duplex mode로 동작한다. W7100A는 double-buffer를 가지고 있기 때문에 receiver는 다음 프레임의 수신시작 전에 수신인터럽트에 응답하지 않아서 그 데이터를 읽지 못하더라도, overrun 에러는 발생하지 않는다. Read동작 중에, SBUF는 receive register로 부터 데이터를 읽는다. 반대로 송신동작 중에 SBUF는 transmit register에 데이터를 load한다. W7100A의 UART는 총 4가지 mode를 가지고 있는데 그 중 하나는 동기식이고 나머지 3가지는 비 동기 방식이다. Mode2와 3은 multiprocessor communication을 위한 특별한 기능을 가지고 있다. 이 기능은 SCON register의 SM2 bit를 setting하여 사용할 수 있다. Master processor는 첫 번째 address byte에 slave processor를 식별할 수 있는 정보를 담아서 송신한다. Address byte는 9번째 bit가 1이고 data byte는 9번째 bit가 0으로 각각 다르다. SM2 = 1인 경우 data byte는 slave에게 아무런 interrupt도 발생시키지 않겠지만 address byte는 모든 slave들에 interrupt를 발생시킬 것이다. 이 경우 interrupt가 발생한 slave는 SM2 bit를 clear하고 이어서 수신될 data byte의 수신을 준비한다. Interrupt가 발생하지 않은 slave는 SM2 bit를 set 한 채로 수신되는 data byte를 무시한다.

UART의 pin기능들을 아래 테이블에 나타내었다.

Table 6.1 UART Pin Description

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

W7100A의 UART는 표준 8051의 UART와 완전히 호환된다. 관련 register들로서 SBUF (0x99), SCON (0x98), PCON (0x87), IE (0xA8), IP (0xB8)가 있다. UART data buffer (SBUF)는 transmit, receive 2개의 register로 구성되어있다. SBUF transmit register에 데이터를 쓰면 send 과정이 시작되고 SBUF receive register에 데이터가 write되면 receive과정이 시작된다.

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 – Mode2 와 3에서 9번 째 송신 data bit. 이 bit는 MCU 동작에 따라 달라진다 (parity check, multiprocessor communication, etc.).

RB8 – Mode2 와 3에서 수신 데이터의 9번 째 bit. Mode1에서 SM2가 0이면 RB08은 stop bit이고, Mode0에서 이 bit는 사용하지 않는다.

UART의 4가지 mode들을 아래 테이블에 나타내었다.

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

UART의 baud rate계산을 아래 테이블에 나타내었다.

Table 6.3 UART Baud Rates

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

아래와 같이 SMOD0 bit는 PCON register에 있다.

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 2-0는 반드시 0이어야 함

6.1 Interrupts

UART interrupt와 관련된 bit들을 아래에 나타내었다. Interrupt는 IE register에 의해 toggle 되고 interrupt들의 우선순위는 IP register를 이용해 설정한다.

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 - RI0 & TI0 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, serial 전송이 끝나면 자동으로 set됨, 반드시 software로 clear해야 함

RI - Receive interrupt flag, serial 수신이 끝나면 자동으로 set됨, 반드시 software로 clear해야 함

Interrupt 생성에 관한 bit들은 hardware와 마찬가지로 software에 의해 clear되거나 set될 수 있다. 즉, interrupt들은 software에 의해 생성되거나 취소될 수 있다.

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 출력은 shift clock이다. Mode0에서 baud rate는 CLK clock 주파수의 1/12로 고정되어 있다. 처음에는 LSB하위 bit를 시작으로 총 8bit들이 전송된다. 수신은 SCON의 RI = 0, REN = 1로 설정하여 초기화 한다.

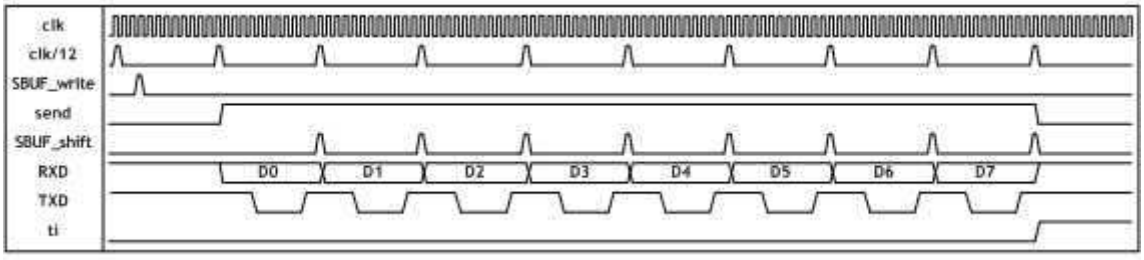


Figure 6.7 Timing Diagram for UART Transmission Mode0 (clk = 88.4736 MHz)

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

시작 bit (항상 0), 데이터 8bit (LSB를 시작으로), 정지 bit (항상1) 총 10bit를 전송한다. 데이터 수신 중에는 시작 bit로 송신단과 동기를 맞춘다. 그 다음 8bit는 SBUF에 저장되고 정지 bit는 SFR SCON (0x98)의 RB08 flag를 trigger한다. Baud rate는 Timer1혹은 2의 설정에 따라 달라질 수 있다. Timer2를 이용할 경우 T2CON (0xC8) register의 TCLK와 RCLK bit를 set해야 한다.

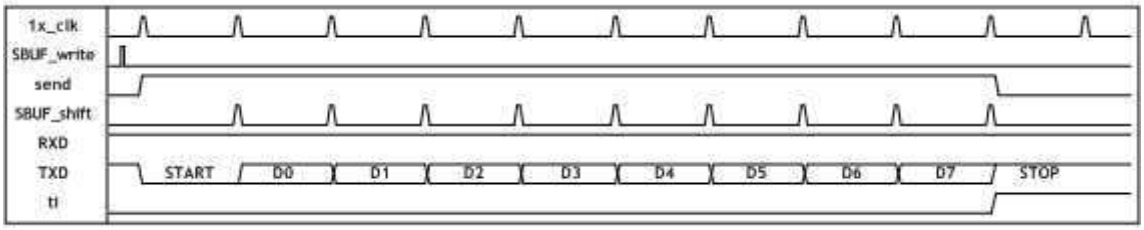


Figure 6.8 Timing Diagram for UART Transmission Mode1

6.4 Mode 2, 9-Bit UART, Fixed Baud Rate

Mode2는 Baud rate가 CLK clock 주파수의 1/32혹은 1/64인 점을 제외하고는 mode1과 거의 유사하다. Mode2는 시작 bit (항상 0), 데이터 8 bit (LSB를 시작으로), programmable bit (9^{th} bit), 정지 bit (항상 1)의 11bit 가 전송되거나 수신된다. Programmable 9^{th} bit 는 parity check bit에 사용된다. Mode2에서는 전송 중에 9^{th} bit가 SCON의 TB08 bit에 저장된다. 수신 중에는 9^{th} bit가 SCON의 RB08 bit에 저장된다.

Figure 6.9 Timing Diagram for UART Transmission Mode2

6.5 Mode 3, 9-Bit UART, Variable Baud Rate, Timer1 or 2 Clock Source

Mode3는 baud rate가 가변적이라는 점을 제외하고는 mode2와 같다. REN = 1인 경우 데이터를 수신할 수 있고 baud rate는 Timer1혹은 Timer2의 설정에 따라 달라진다. Timer2의 clock을 사용하는 경우에는 T2CON (0xC8) register의 TCLK와 RCLK bit를 설정해야 한다.

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 / Mode 2		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) * (Clock Frequency / 12(256 – TH1))

Using Timer2 – Baud Rate = Clock Frequency / (32 * (65536 – (RLDH, RLDL)))

7 Watchdog Timer

7.1 Overview

Watchdog Timer는 아래 Figure 7.1과 같이 시스템 main clock과 연결된 divider들에 의해 동작한다. Divider의 출력은 CKCON (0xC8) register의 WD[1:0] bit들로 선택할 수 있고 이것을 이용해서 timeout 주기를 설정할 수 있다. Timeout이 발생하면, interrupt flag가 set되고 설정에 따라 시스템 reset이 발생할 수 있다. Interrupt enable bit와 global interrupt가 enable 되어있다면, interrupt는 정상적으로 동작할 것이다. Reset과 interrupt는 완전히 별개의 기능이다. 따라서 timeout이 발생하면 각각 이것을 인지하고 응용에 따라 모두 동작하거나 하나만 동작하도록 할 수 있다.

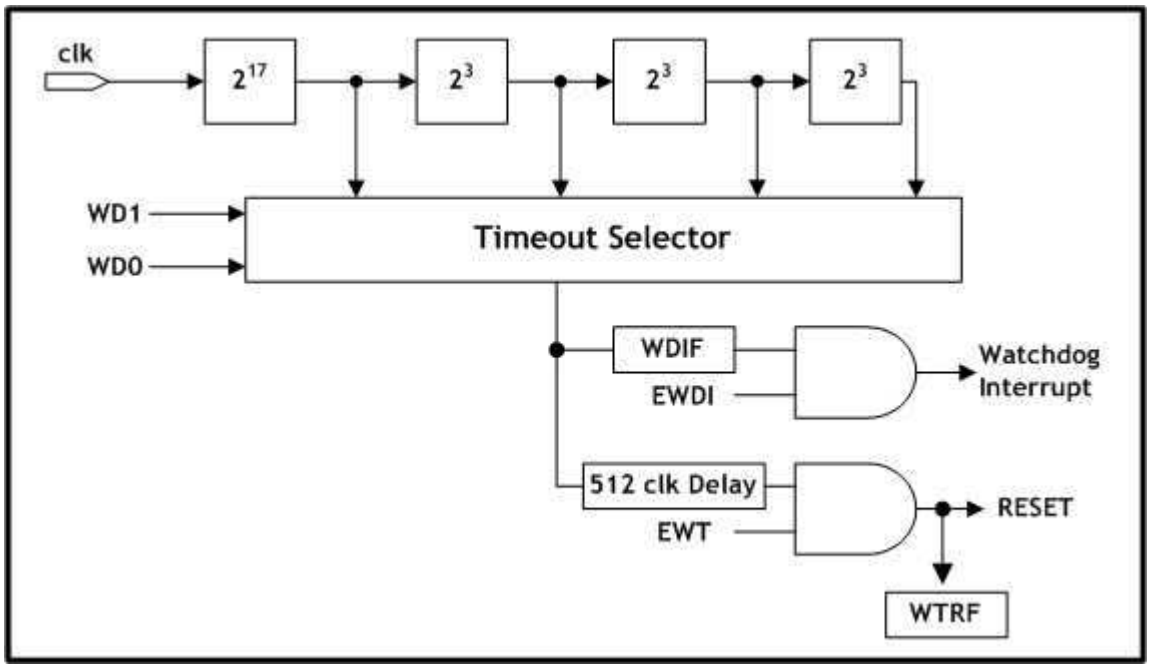


Figure 7.1 Watchdog Timer Structure

7.2 Interrupts

Watchdog interrupt와 관련한 bit들은 아래와 같다. Interrupt는 IE (0xA8)와 EIE (0xE8) register를 이용해서, 켜고 끌 수 있으며 interrupt 우선순위는 EIP (0xF8) register를 통해 설정할 수 있다. IE는 EA bit를 설정함으로써 global interrupt 시스템을 제어할 수 있다.

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. Watchdog interrupt는 WDIF, EWT bit와 EIE SFR의 EWDI bit와 연계되어 있다. Software로 WDIF를 설정해도 interrupt가 enable된 경우Watchdog interrupt가 발생할 것이다. WDIF bit는 **Timed Access Register**를 이용해서 반드시 interrupt service루틴에서 software로 clear해야 한다. 'Timed Access'과정은 section 7.8 'Timed Access'를 참조하기 바란다.

모든 interrupt관련 bit들은 hardware에 의한 것과 동일하게 software에 의해 set되거나 clear될 수 있다. 즉, interrupt들은 software에 의해 생성되거나 취소 될 수 있다.

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

Watchdog Timer reset의 동작과정은 다음과 같다. Timeout interval을 초기화 한 다음 system을 reset하면 Watchdog은 RWT bit를 통해 reset된다. EWT (Enable Watchdog Timer reset = WDCON.1) bit를 설정하여 reset mode를 설정한다. 사용자는 timer가 정해진 timeout 시간에 도달하기 전 software를 통해 Watchdog Timer를 reset할 수 있다. Timeout전에 RWT (Reset Watchdog Timer = WDCON.0) bit를 set하여 reset을 했다면 timer는 다시 시작될 것이다. 하지만 timeout까지 RWT bit를 set하지 않았다면, Watchdog은 MCU를 reset할 것이다. Software에 의해 RWT bit를 set하면, RWT bit는 자동으로 clear된다. Reset이 발생하면, WTRF (Watchdog Timer reset Flag = WDCON.2) bit는 자동으로 set 될 것이다. 하지만, 이 bit는

software에 의해 반드시 직접 clear해야 한다.

7.4 Simple Timer

Watchdog Timer는 reset mode를 사용하지 않고 (EWT = 0) interrupt를 초기화하면 (EWDI = 0) 보통 timer로써 사용할 수 있다. Timer는 WD[1:0]에 미리 설정된 값까지 count를 하고 timeout이 발생하면 Watchdog interrupt flag를 set할 것이다. 이 때 RWT bit를 설정함으로써 timer는 계속해서 동작할 수 있다. WDIF bit는 software를 통해서나 reset을 통해 clear할 수 있다. Watchdog interrupt는 긴 timer를 요구하는 응용에 사용할 수 있다. Interrupt는 EWDI (Enable WatchDog timer Interrupt = EIE.4) bit를 설정하여 enable하고, timeout이 발생 했을 때 Watchdog Timer는 WDIF bit를 set할 것이다. EA bit가 enable되어있다면 interrupt가 발생할 것이다.

WDIF는 Watchdog reset이 발생하기 512 clock전에 set된다는 점을 명심해야 한다. Watchdog interrupt는 반드시 software에 의해 clear되어야 한다. Watchdog interrupt를 적절히 이용한다면, 예기치 못한 error발생을 감시할 수 있는 시스템 monitor로써 사용할 수 있다.

7.5 System Monitor

WDCON SFR의 EWT bit를 설정하면 Watchdog timeout이 발생했을 때 W7100A는 reset된다. 이 기능을 이용하여 Watchdog Timer를 시스템 monitor로 사용할 수 있다. 예를 들어 사용자가 원치 않는 잘못된 code가 실행되고 있다고 가정하자. 사용자가 의도한 code에는 RWT bit를 주기적으로 clear하는 routine이 있지만, 잘못된 code에는 없기 때문에 Watchdog timeout이 발생하고 W7100A는 reset될 것이다. 이 방법을 이용해서 System이 잘못 동작 할 때 이것을 바로 잡을 수 있다.

7.6 Watchdog Related Registers

Watchdog Timer는 동작 중에 몇몇 SFR bit들을 사용한다. 이들 bit들은 reset source, interrupt source, software polled timer혹은 세 가지를 조합한 경우처럼 공유할 수 있다. Reset과 interrupt는 status flag를 가지고 있다. Watchdog또한 timer를 재 시작하는 bit를 가지고 있다.

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

RWT bit를 설정하여 Watchdog Timer가 reset되는 동안에도 Watchdog Timer 동작은 disable 되지 않는다. 다음 섹션에서 Watchdog을 제어하는 bit들에 대해서 설명할 것이다.

7.7 Watchdog Control

Watchdog 제어 bit들을 아래에 나타내었다. 이 register에 access (write) 하려면 '7.8 Timed Access Registers' 과정을 거쳐야 한다.

WDCON (0xD8)

7	6	5	4	3	2	1	0	Reset
-	-	-	-	WDIF	WTRF	EWT	RWT	0x00

Figure 7.6 Watchdog Control Register

Note: WTRF – Watchdog Timer reset flag. 이 flag가 set되면 Watchdog Timer reset이 발생한다. 하지만 software를 통해 이 flag를 enable하면 Watchdog Timer reset은 trigger되지 않는다. Reset과정 동안 이 flag는 software에 의해 clear되어야 한다. 만약 EWT bit가 clear되면 Watchdog Timer는 더 이상 동작하지 않는다.

EWT – Enable the Watchdog Timer reset. 이 bit는 Watchdog Timer가 microcontroller를 reset하도록 제어한다. Watchdog Timer가 Watchdog interrupt를 발생하는 기능에는 아무런 영향을 주지 않는다. 이 bit를 제어하려면 Timed Access 과정을 이용해야 한다.

0: Watchdog Timer timeout이 발생해도 microcontroller를 reset하지 않음

1: Watchdog Timer timeout이 발생하면 microcontroller를 reset함

RWT – Watchdog Timer를 reset함. RWT는 Watchdog Timer count를 reset한다. Watchdog Timer가 expire되기 전에 Timed Access과정을 반드시 거쳐야 한다. RWT가 enable상태면 Reset혹은 interrupt가 발생한다.

Unimplemented bit - Read as 0 or 1

아래 테이블은 Watchdog control bit들의 기능을 정리한 것이다.

Table 7.3 Watchdog Bits and Actions

EWT	EWDI	WDIF	Result
X	X	0	Watchdog을 사용하지 않음
0	0	1	Watchdog timeout이 발생해도 interrupt는 발생하지 않음
0	1	1	Watchdog interrupt발생
1	0	1	Watchdog timeout이 발생해도 interrupt는 발생하지 않음 Timeout이 발생하고 512 clock안에 RWT bit를 set하지 않으면 Watchdog Timer reset이 발생함

1	1	1	Watchdog interrupt 발생함, Timeout이 발생하고 512 clock안에 RWT bit를 set하지 않으면 Watchdog Timer reset이 발생함

7.7.1 Clock Control

Watchdog timeout주기는 WD[1:0] bit를 이용해서 설정한다. WD[1:0] bit는 아래와 같이 CKCON register내부에 있다.

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

Watchdog은 CLK pin과 직접 clock이 연결되어있다. Watchdog timeout주기는 WD[1:0] bit에 따라 아래와 같이 4가지 중 선택할 수 있다. *W7100A clock frequency = 88.4736MHz

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

Watchdog reset이 enable되면 interrupt와 상관없이 timeout후 512 clock이 지나면 reset이 발생한다. 그러므로 실제 Watchdog timeout은 선택된 Watchdog interval에 512 clock을 더한 값이 된다.

7.8 Timed Access Registers

WDCON은 timed access register이기 때문에 비정상적인 쓰기동작(accidental write)을 방지하기 위해 반드시 아래와 같은 과정을 거쳐서 write한다. TA는 SFR address의 0xC7에 위치한다.

```
MOV TA, #0xAA
MOV TA, #0x55
;Any direct addressing instruction writing timed access register
```

TA SFR을 위와 같은 순서로 write하고 WDCON SFR의 값을 변경한다. WDCON의 값을 변경할 때는 매번 이와 같은 과정을 거쳐야 한다.

Table 7.5 Timed Access Registers

Register name	Description
WDCON(0xD8)	Watchdog configuration

8 TCPIPCore

8.1 Memory Map

TCPIPCore는 아래 그림처럼 Common register와 SOCKET register, TX memory, RX memory의 조합으로 구성되어있다.

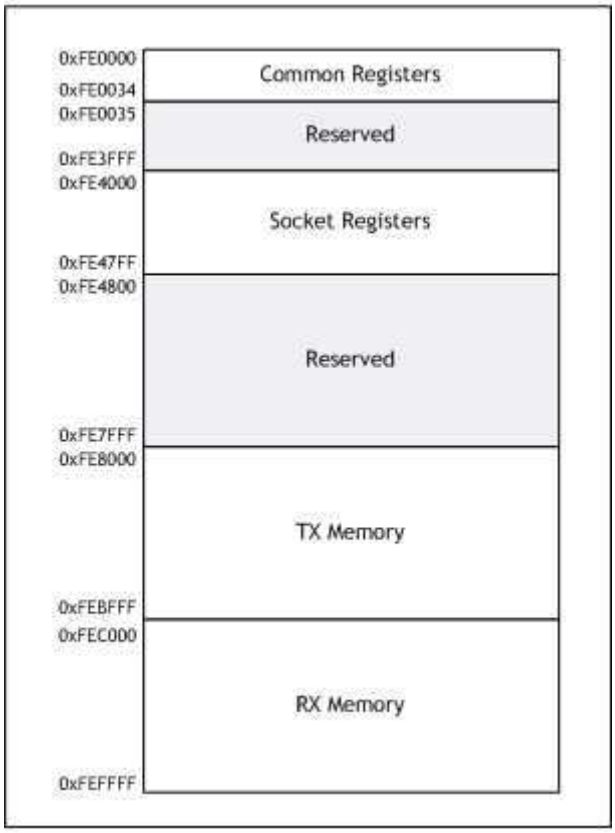


Figure 8.1 TCPIPCore Memory Map

8.2 Registers list

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	
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	W7100A Version Register
0xFE0020 ~ 0xFE0027		Reserved
0xFE0028	PTIMER	PPP Link Control Protocol Request Timer Register
0xFE0029	PMAGIC	PPP LCP Magic Number Register
0xFE002A ~ 0xFE002F		Reserved
0xFE0030	INTLEVEL0	INTLEVEL (Interrupt Low Level Timer Register)
0xFE0031	INTLEVEL1	
0xFE0032		Reserved
0xFE0033		

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_RD
0xFE4023	S0_TX_RD1	(SOCKET 0 Transmit Memory Read Pointer Register)
0xFE4024	S0_TX_WR0	S0_TX_WR
0xFE4025	S0_TX_WR1	(SOCKET 0 Transmit Memory Write Pointer Register)
0xFE4026	S0_RX_RSR0	S0_RX_RSR
0xFE4027	S0_RX_RSR1	(SOCKET 0 Received Data Size Register)
0xFE4028	S0_RX_RD0	S0_RX_RD
0xFE4029	S0_RX_RD1	(SOCKET 0 Receive Memory Read Pointer Register)
0xFE402A	S0_RX_WR0	S0_RX_WR
0xFE402B	S0_RX_WR1	(SOCKET 0 Receive Memory Write Pointer Register)
0xFE402C	S0_IMR	SOCKET 0 Interrupt Mask Register
0xFE402D	S0_FRAG0	S0_FRAG
0xFE402E	S0_FRAG1	(SOCKET 0 Fragment Field Value in IP Header Register)
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 (SOCKET 1 Transmit Memory Read Pointer Register)
0xFE4123	S1_TX_RD1	
0xFE4124	S1_TX_WR0	S1_TX_WR (SOCKET 1 Transmit Memory Write Pointer Register)
0xFE4125	S1_TX_WR1	
0xFE4126	S1_RX_RSR0	S1_RX_RSR (SOCKET 1 Received Data Size Register)
0xFE4127	S1_RX_RSR1	
0xFE4128	S1_RX_RD0	S1_RX_RD (SOCKET 1 Receive Memory Read Pointer Register)
0xFE4129	S1_RX_RD1	
0xFE412A	S1_RX_WR0	S1_RX_WR (SOCKET 1 Receive Memory Write Pointer Register)
0xFE412B	S1_RX_WR1	
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	
0xFE4309	S3_DHAR3	
0xFE430A	S3_DHAR4	S3_DHAR (SOCKET 3 Destination Hardware Address Register)
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	
0xFE450E	S5_DIPR2	S5_DIPR (SOCKET 5 Destination IP Address Register)
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 (SOCKET 7 Transmit Memory Read Pointer Register)
0xFE4723	S7_TX_RD1	
0xFE4724	S7_TX_WR0	S7_TX_WR (SOCKET 7 Transmit Memory Write Pointer Register)
0xFE4725	S7_TX_WR1	
0xFE4726	S7_RX_RSR0	S7_RX_RSR (SOCKET 7 Received Data Size Register)
0xFE4727	S7_RX_RSR1	
0xFE4728	S7_RX_RD0	S7_RX_RD (SOCKET 7 Receive Memory Read Pointer Register)
0xFE4729	S7_RX_RD1	
0xFE472A	S7_RX_WR0	S7_RX_WR (SOCKET 7 Receive Memory Write Pointer Register)
0xFE472B	S7_RX_WR1	

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 Common Register

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

MR은 S/W reset, ping block mode, PPPoE mode에 사용된다.

7	6	5	4	3	2	1	0
RST			PB	PPPoE			

Bit	Symbol	Description
7	RST	**S/W Reset** 이 bit가 '1'인 경우 내부 register는 초기화 되고 reset후에 자동으로 clear 됨
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 사용자가 router와 같은 장비 없이 ADSL을 사용하고자 한다면, 이 bit를 '1'로 설정하여 ADSL 서버에 연결한다. 자세한 사항은 'How to connect ADSL' 문서를 참고하기 바란다.
2	Reserved	Reserved
1	Reserved	Reserved
0	Reserved	Reserved

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

GAR은 default gateway address를 설정할 때 사용한다.

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]

SUBR은 subnet mask address를 설정할 때 사용한다.

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]

SHAR은 Source Hardware address를 설정할 때 사용한다.

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]

SIPR은 Source IP address를 설정할 때 사용한다.

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]

IR은 interrupt 발생여부를 판단하기 위해 W7100A의 MCU에서 access한다. IR bit가 set되면 이 INT5(nINT5: TCPIPcore interrupt) 신호는 low 상태 asserted되고 IR의 모든 bit들을 clear하지 않는 이상 high상태로 변하지 않는다. IR은 MCU에 interrupt신호를 발생시킨다.

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

Bit	Symbol	Description
7	CONFLICT	**IP Conflict** ARP 요청에 Source IP address와 같은 IP address응답이 있다면, 이 bit는 ‘1’로 set된다. 이 bit는 ‘1’을 write함으로써 ‘0’으로 clear할 수 있다.
6	UNREACH	**Destination unreachable** W7100A는 UDP 데이터전송 중에 destination IP address가 존재하지 않는 경우 ICMP (Destination Unreachable) packet을 수신할 것이다. 이 경우 UNREACH bit는 ‘1’로 set된다. 이 bit는 ‘1’을 write함으로써 ‘0’으로 clear

		할 수 있다.
5	PPPoE	**PPPoE Connection Close** PPPoE mode에서 이 bit가 '1'인 경우는 PPPoE 연결이 closed임을 나타낸다. 이 bit는 '1'을 write함으로써 '1'으로 clear할 수 있다.
4	Reserved	Reserved
3	Reserved	Reserved
2	Reserved	Reserved
1	Reserved	Reserved
0	Reserved	Reserved

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

IMR (Interrupt Mask Register)는 interrupt masking하는데 사용한다. 각 interrupt mask bit는 Interrupt register2 (IR2)의 bit와 같다. Interrupt mask bit가 set되어있다면, IR2의 해당 bit가 set되었을 때 interrupt가 발생 할 것이다. 만약 IMR이 '0'으로 set되어 있다면, IR2의 해당 bit가 set되더라도 interrupt는 발생하지 않을 것이다.

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-period Register) [R/W] [0xFE0017 - 0xFE0018] [0x07D0]

RTR은 timeout 주기를 설정한다. 이 register에서 1값이 갖는 의미는 100us와 같다. Default timeout은 2000 (0x07D0) 즉, 200ms이다.

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

0xFE0017	0xFE0018
0x0F	0xA0

만약 peer로 부터 응답이 없거나 정해진 timeout시간 보다 delay가 길어질 경우 재 전송이 발생한다.

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

RCR은 재 전송 횟수를 설정한다. 만약 재 전송횟수가 RCR에 저장된 횟수 이상으로 발생할 경우, Timeout Interrupt가 발생한다. (SOCKET*n* Interrupt Register (Sn_IR)의 TIMEOUT bit는 '1'로 설정된다.) TCP 통신인 경우, Sn_IR(TIMEOUT)= '1'과 동시에 Sn_SR의 값이 'SOCK_CLOSED'로 변경된다. 하지만 TCP 통신이 아닌 경우, Sn_IR(TIMEOUT) = '1'로 설정되며 Sn_SR의 값은 변경되지 않는다.

W7100A에서는 RTR과 RCR로 Data 재전송의 시간과 횟수를 설정할 수 있다. W7100A의 Timeout은 ARP retransmission timeout과 TCP retransmission timeout 2가지에 의해 발생할 수 있다. 먼저 ARP("RFC 826" 참조, http://www.ietf.org/rfc.html) retransmission timeout을 살펴보면, W7100A는 IP, UDP, TCP를 이용한 통신시 상대방의 IP address로 MAC address를 알기 위해 자동으로 ARP-request를 전송한다. 이때 상대방의 ARP-response 수신을 기다리는데, RTR의 설정 대기 시간 동안 ARP-response의 수신이 없으면, Timeout이 발생하고 ARP-request를 Retransmission한다. 이와 같은 작업은 'RCR + 1'만큼 반복하게 된다.

'RCR + 1'개의 ARP-request retransmission이 일어나고, 그에 대한 ARP-response가 없다면, Final timeout이 발생하게 되고, Sn_IR(TIMEOUT) = '1' 된다.

ARP-request의 Final timeout(ARP timeout) 값은 다음과 같다.

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

TCP packet retransmission timeout을 살펴보면, W7100A는 TCP packet (SYN, FIN, RST, DATA packet)을 전송하고 그에 대한 Acknowledgment(ACK)을 RTR과 RCR에 의해 설정된 대기 시간 동안 기다리게 된다. 이때 상대방으로부터 ACK가 없으면 Timeout이 발생하고 이전에 보냈던 TCP packet을 Retransmission한다. 이와 같은 작업은 'RCR + 1'만큼 반복하게 된다.

'RCR + 1'개의 TCP packet retransmission이 일어나고, 그에 대한 ACK 수신이 없다면, Final timeout이 발생하게 되고, Sn_IR(TIMEOUT) = '1'과 동시에 Sn_SSR이 'SOCK_CLOSED"로 변경된다. TCP packet retransmission의 Final timeout(TCP timeout) 값은 다음과 같다.

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

N : Retransmission count, 0 <= N <= M

M : RTR X $2^{(M+1)}$ > 65535 and 0 <= M <= RCR를 만족하는 최소값

RTR_{MAX} : RTR X 2^M

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

ARP_{TO} = 2000 X 0.1ms X 9 = 1800ms = 1.8s (ARP_{TO} = ARP timeout, TCP_{TO} = TCP timeout)

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

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

= 318000 X 0.1ms = 31.8s

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

PATR은 PPPoE 연결을 위한 인증 type을 알려준다. W7100A는 PAP와 CHAP의 2가지 type의 인증 방법을 지원한다.

Value	Authentication Type
0xC023	PAP
0xC223	CHAP

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

PPPALGO는 PPPoE연결의 인증 알고리즘을 알려준다. 자세한 정보는 PPPoE application note를 참고하기 바란다.

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

PTIMER은 LCP echo request를 보내는 지속시간을 나타낸다. 1의 값은 25ms를 의미한다.

Ex) in case that PTIMER is 200,

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

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

PMAGIC은 LCP negotiation도중에 Magic number option을 설정하는데 사용된다. W5100의 Application note 'How to connect ADSL'을 참조하기 바란다.

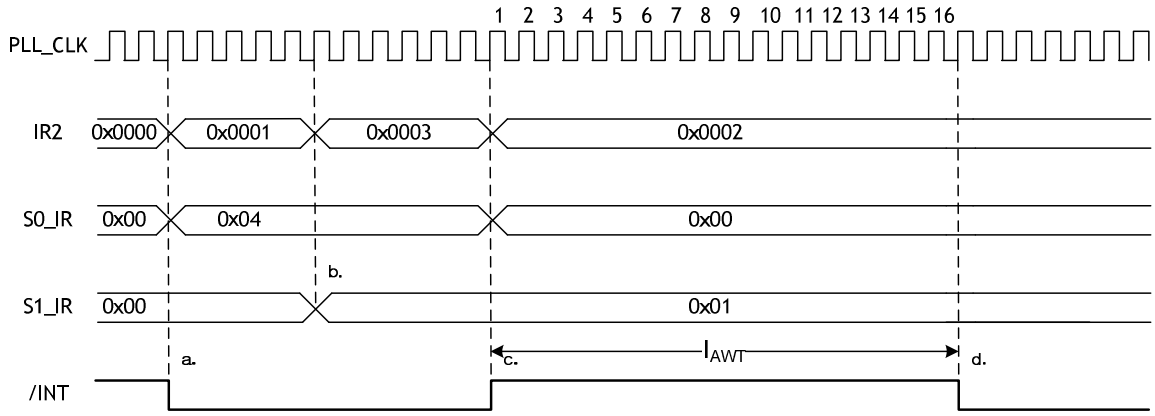
VERSIONR (W7100A Chip Version Register) [R] [0xFE001F] [0x02]

VERSIONR은 W7100A chip version을 나타내는 register이다.

INTLEVEL (Interrupt Low Level Timer Register)[R/W][0xFE0030 - 0xFE0031][0x0000]

INTLEVEL register는 Interrupt Assert wait time(I_{AWT})을 설정한다. 다음 interrupt가 발생했을 때 설정한 시간(I_{AWT})만큼 기다린 다음 칩 내부의 INT5 신호를 Low로 assert한다. TCP/IP Core interrupt를 사용하고자 한다면, 이 register값은 반드시 최소 0x2B00 이상으로 설정하여야 한다. 그렇지 않으면 MCU에서 TCP/IP Core interrupt가 무시될 수도 있다.

$$I_{AWT} = (INTLEVEL0 + 1) * PLL\_CLK \text{ (when } INTLEVEL0 > 0)$$



a. 소켓0 에서 interrupt가 발생했다면 (S0_IR(3) = '1' 해당 IR2 bit도 '1'로 set되고 INT5신호는 Low로 assert된다.

b. 연속해서 소켓1 에서 interrupt가 발생하면 (S1_IR1(0) = '1') 해당 IR2 bit가 '1'로 set된다.

c. MCU는 S0_IR을 clear(S0_IR1 = 0x00) 하고 해당 IR2 bit 또한 clear한다. 칩 내부의 INT5신호는 low에서(activated) High로(deactivated) 변한다.

d. 여기서 S0_IR이 clear되었지만, socket1 interrupt 때문에 IR2의 값은 0x00이 아니다. 따라서 칩 내부의 INT5신호는 Low로 assert되어야 한다. 이 때 INTLEVEL register의 값이 0x000F라면 칩 내부의 INT5신호는 I_{AWT}(16 PLL_CLK) time 후에 Low로 assert될 것이다.

IR2 (W7100A SOCKET Interrupt Register) [R/W] [0xFE0034] [0x00]

IR2는 W7100A SOCKET interrupt가 발생할 경우 이것을 알려주는 register이다. Interrupt가 발생했을 때, IR2의 해당 bit가 set된다. 이 경우 IR2의 모든 bit들이 '0'으로 clear될 때까지 INT5(nINT5: TCPIPCore interrupt) 신호는 low상태가 된다. Sn_IR bit들을 이용해서 IR2 register를 clear하면 INT5 신호는 high 상태가 된다.

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** SOCKET 7에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S7_IR에 반영된다. 이 bit는 사용자에 의해 S7_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.
6	S6_INT	**Occurrence of SOCKET 6 Interrupt** SOCKET 6에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S6_IR에 반영된다. 이 bit는 사용자에 의해 S6_IR가

		0x00으로 clear되면 이에 따라 자동으로 clear된다.
5	S5_INT	**Occurrence of SOCKET 5 Interrupt** SOCKET 5에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S5_IR에 반영된다. 이 bit는 사용자에 의해 S5_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.
4	S4_INT	**Occurrence of SOCKET 4 Interrupt** SOCKET 4에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S4_IR에 반영된다. 이 bit는 사용자에 의해 S4_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.
3	S3_INT	**Occurrence of SOCKET 3 Interrupt** SOCKET 3에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S3_IR에 반영된다. 이 bit는 사용자에 의해 S3_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.
2	S2_INT	**Occurrence of SOCKET 2 Interrupt** SOCKET 2에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S2_IR에 반영된다. 이 bit는 사용자에 의해 S2_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.
1	S1_INT	**Occurrence of SOCKET 1 Interrupt** SOCKET 1에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S1_IR에 반영된다. 이 bit는 사용자에 의해 S1_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.
0	S0_INT	**Occurrence of SOCKET 0 Interrupt** SOCKET 0에서 interrupt가 발생한 경우, 이 bit는 '1'로 되고 이 interrupt 정보는 S0_IR에 반영된다. 이 bit는 사용자에 의해 S0_IR가 0x00으로 clear되면 이에 따라 자동으로 clear된다.

8.3.2 SOCKET Registers

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

Sn_MR은 SOCKET n의 option이나 protocol type등을 설정한다.

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

Bit	Symbol	Description
7	MULTI	**Multicasting** 0 : disable Multicasting 1 : enable Multicasting 이 기능은 UDP의 경우에만 적용됨 (P3-P0 : '0010')

<table>
<tr><td></td><td></td><td>Multicasting을 사용하기 위해 OPEN 명령 이전에 SOCKET n destination IP와 port register에 각각 multicast group address와 port number를 write함</td></tr>
<tr><td>6</td><td>MF</td><td>MAC Filter
0: disable MAC filtering
1: enable MAC filtering
자신의 MAC주소와 broadcasting MAC을 제외한 나머지 MAC주소는 filtering 하여 수신하지 않는다.</td></tr>
<tr><td>5</td><td>ND/MC</td><td>Use No Delayed ACK
0 : Disable No Delayed ACK option
1 : Enable No Delayed ACK option,
이 기능은 TCP의 경우에만 적용됨 (P3-P0: ‘0001’)
만약 이 bit가 ‘1’로 set되어있다면 peer로부터 데이터 packet을 수신한 다음 곧바로 ACK packet이 전송될 것이다. 만약 이 bit가 ‘0’이라면 ACK packet은 내부 timeout 메커니즘에 따라 전송됨
Multicast
0 : using IGMP version 2
1 : using IGMP version 1
이 bit는 MULTI bit가 enable상태이고 UDP모드일 때 유효함 (P3-P0: ‘0010’)
추가적으로 multicast는 IGMP message에 Join/Leave/Report와 같은 version number를 Multicast group으로 보냄</td></tr>
<tr><td>4</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>3</td><td>P3</td><td rowspan="4">Protocol
해당 SOCKET의 TCP, UDP, IPRAW등의 protocol을 설정함
<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>
S0_MR_MACRAW와 S0_MR_PPPoE는 SOCKET 0에만 쓸 수 있음
S0_MR_PPPoE는 임시로 PPPoE server connection/Termination에 사용되지만, Connection이 잡히면, 다른 protocol로 사용될 수 있음</td></tr>
<tr><td>2</td><td>P2</td></tr>
<tr><td>1</td><td>P1</td></tr>
<tr><td>0</td><td>P0</td></tr>
</table>

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

Sn_CR은 OPEN, CLOSE, CONNECT, LISTEN, SEND, RECEIVE와 같은 SOCKET *n*의 명령을 설정하는데 사용한다. W7100A이 명령을 인식하고 난 다음 Sn_CR은 W7100A에 의해 자동으로

clear 된다. Sn_CR이 0x00으로 clear되었더라도, 해당 명령은 여전히 처리 중 일 수 있다. Sn_CR의 명령처리가 완료되었는지는 Sn_IR이나 Sn_SR을 확인하면 된다.

<table>
<tr><th>Value</th><th>Symbol</th><th>Description</th></tr>
<tr><td>0x01</td><td>OPEN</td><td>SOCKET n은 초기화 되고 Sn_MR (P3:P0)로 선택한 protocol에 따라 open된다. 아래 테이블은 Sn_MR에 따른 Sn_SR값을 보여준다.
<table>
<tr><th>Sn_MR(P3:P0)</th><th>Sn_SR</th></tr>
<tr><td>Sn_MR_CLOSE(0x00)</td><td>-</td></tr>
<tr><td>Sn_MR_TCP(0x01)</td><td>SOCK_INIT(0x13)</td></tr>
<tr><td>Sn_MR_UDP(0x02)</td><td>SOCK_UDP(0x22)</td></tr>
<tr><td>Sn_MR_IPRAW(0x03)</td><td>SOCK_IPRAW(0x32)</td></tr>
<tr><td>S0_MR_MACRAW(0x04)</td><td>SOCK_MACRAW(0x42)</td></tr>
<tr><td>S0_MR_PPPoE(0x05)</td><td>SOCK_PPPoE(0x5F)</td></tr>
</table></td></tr>
<tr><td>0x02</td><td>LISTEN</td><td>LISTEN은 TCP mode (Sn_MR(P3:P0) = Sn_MR_TCP)에서만 유효함
이 모드에서, SOCKET n 은 ‘TCP CLIENT’로부터 connection-request (SYN packet)을 기다리는 TCP server로 설정된다. 이 경우 Sn_SR의 상태는 SOCKET_INIT에서 SOCKET_LISTEN으로 바뀐다.
Client의 connection-request가 성공적으로 established되면 Sn_SR의 상태는 SOCK_LISTEN에서 SOCK_ESTABLISHED로 변하고 Sn_IR(0)은 ‘1’로 된다. 반면에 connection failure (SYN/ACK packet전송 실패)의 경우 Sn_IR(3)은 ‘1’로 set되고 Sn_SR의 상태는 SOCK_CLOSED로 변한다.
cf> 만약 connection request동안 TCP client의 destination port가 존재하지 않을 경우, W7100A는 RST packet을 전송하고 Sn_SR의 상태는 변하지 않는다.</td></tr>
<tr><td>0x04</td><td>CONNECT</td><td>CONNECT는 TCP mode(Sn_MR(P3:P0) = Sn_MR_TCP)에서만 유효하고 SOCKET n 이 ‘TCP CLIENT’로 동작할 경우 사용된다. CONNECT는 Sn_DIPR와 Sn_DPORTR로 설정된 ‘TCP SERVER’에게 Connect-request(SYN packet)를 전송한다. Connect-request가 성공했을 경우 (SYN/ACK packet을 수신했을 경우), Sn_IR(0)=‘1’로 되고 Sn_SSR은 SOCK_ESTABLISHED로 변경된다.
Connect-request가 실패했을 경우는 다음과 같이 3가지가 있다.
- ARP-process를 통해 Destination hardware address를 얻지 못하여 ARP timeout이 발생(Sn_IR(3)=‘1’)한 경우
- SYN/ACK packet를 수신 못하고 TCP timeout이 발생(Sn_IR(3)= ‘1’)한 경우
- SYN/ACK packet 대신 RST packet을 수신했을 경우.
위와 같은 경우 Sn_SR은 SOCK_CLOSED상태로 바뀐다.</td></tr>
<tr><td>0x08</td><td>DISCON</td><td>DISCON은 TCP mode일 때만 유효하다.</td></tr>
</table>

		W7100A는 'TCP SERVER'와 'TCP CLIENT'에 상관없이, 접속중인 상대방에게 Disconnect-request(FIN packet)를 전송하거나(Active close), 상대방으로부터 Disconnect-request(FIN packet)을 수신했을 때(Passive close), W7100A는 FIN packet을 전송한다(Disconnect-process). Disconnect-request가 성공했다면(FIN/ACK packet을 수신했을 경우), Sn_SR은 SOCK_CLOSED로 변경된다. 그러나 Disconnect-request가 실패했다면, TCP timeout이 발생(Sn_IR(3)= '1')하고 Sn_SR은 SOCK_CLOSED로 변경된다. cf> DISCON 대신 CLOSE를 사용할 경우, Disconnect-process(disconnect-request 전송)없이, 단지 Sn_SR만 SOCK_CLOSED로 변경된다. 그리고 통신 중 상대방으로부터 RST packet을 수신할 경우, 무조건 Sn_SR은 SOCK_CLOSED로 변경된다.
0x10	CLOSE	SOCKET *n* 을 close한다. 이 때 Sn_SR은 SOCK_CLOSED로 변경된다.
0x20	SEND	SEND는 TX memory의 buffer에 전송되지 않은 데이터를 송신한다. 자세한 사항은 SOCKET *n* TX Free Size Register (Sn_TX_FSR0), SOCKET *n* TX Write Pointer Register (Sn_TX_WR), SOCKET *n* TX Read Pointer Register(Sn_TX_RD)를 참고하기 바란다.
0x21	SEND_MAC	SEND_MAC은 UDP mode일 때만 유효하다. 기본동작은 SEND와 같다. SEND는 자동으로 ARP-process를 통해 Destination hardware address를 얻은 후 Data를 전송하는 반면, SEND_MAC은 Host가 설정한 Sn_DHAR을 Destination hardware address로 하여 Data를 전송한다.
0x22	SEND_KEEP	SEND_KEEP은 TCP mode일 때만 유효하다. Keep alive packet을 송신하여 connection이 유효한지 확인한다. 만약 상대방이 더 이상 응답이 없어서 connection이 유효하지 않은 경우 connection을 종료한다. Timeout interrupt가 발생한다.
0x40	RECV	RECV는 RX read pointer register (Sn_RX_RD)를 이용해서 데이터를 수신한다. 자세한 사항은 9.2.1.1 SERVER mode의 Receiving Process와 SOCKET *n* RX Received Size Register (Sn_RX_RSR), SOCKET *n* RX Write Pointer Register(Sn_RX_WR), and SOCKET *n* RX Read Pointer Register(Sn_RX_RD)를 참고하기 바란다.

아래 명령어들은 오직 SOCKET 0 에서 S0_MR(P3:P0) = S0_MR_PPPoE 일 때만 유효하다.
자세한 사항은 W5100 application note 'How to use ADSL'을 참고하기 바란다.

Value	Symbol	Description
0x23	PCON	PPPoE Discovery Packet을 전송하여 ADSL 연결 시작
0x24	PDISCON	ADSL connection 종료

0x25	PCR	각 과정에서, REQ message 전송
0x26	PCN	각 과정에서, NAK message 전송
0x27	PCJ	각 과정에서, REJECT message 전송

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

Sn_IR register는 SOCKET *n*의 interrupt (establishment, termination, receiving data, timeout) type과 같은 정보를 제공한다. Interrupt가 발생하고 Sn_IMR의 해당 mask bit가 '1'인 경우 Sn_IR의 interrupt bit는 '1'로 된다.

Sn_IR bit를 clear하기 위해서는, 해당 bit에 다시 '1'을 write해야 한다. Sn_IR의 모든 bit들이 clear되면, IR(*n*)은 자동으로 clear된다. Sn_IR은 MCU에 INT5신호 (nINT5: TCPIPCore interrupt)를 발생시킨다.

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

Bit	Symbol	Description
7	PRECV	PPP Receive Interrupt, 지원하지 않는 옵션 데이터를 수신한 경우
6	PFAIL	PPP Fail Interrupt, PAP 인증이 실패한 경우
5	PNEXT	PPP Next Phase Interrupt, ADSL연결 과정에서 phase가 변할 경우
4	SENDOK	SEND OK Interrupt, SEND명령이 완료되면
3	TIMEOUT	TIMEOUT Interrupt, ARP timeout혹은 TCP timeout이 발생한 경우
2	RECV	Receive Interrupt, peer로 부터 데이터 packet이 수신된 경우
1	DISCON	Disconnect Interrupt, peer로 부터 FIN/ACK packet의 FIN이 수신된 경우
0	CON	Connect Interrupt, peer와 연결이 성립되어 SOCKET status가 established로 바뀔 때 1번 발생함

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

Sn_IMR은 Host로 알려줄 SOCKET *n*의 Interrupt를 설정한다. Sn_IMR의 Interrupt mask bit들은 Sn_IR의 Interrupt bit들과 각각 대응된다. 임의의 SOCKET interrupt가 발생하고 Sn_IMR의 그 bit가 '1'로 설정되어있을 경우 Sn_IR의 대응 Bit가 '1'로 설정된다. Sn_IMR과 Sn_IR의 임의 bit가 모두 '1'일 때 IR(n) = '1'이 된다. 이때 IMR(n) = '1'이라면 Host에 Interrupt가 발생('/INT' signal low assert)한다.

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

Sn_SR은 SOCKETn의 SOCKET 상태를 알려준다. SOCKET status는 Sn_CR의 Command나, packet 송수신중에 변경될 수 있다.

아래 테이블은 SOCKET *n* 의 여러 가지 상태를 나타낸 것이다.

Value	Symbol	Description
0x00	SOCK_CLOSED	SOCKET *n*의 resource가 release된 상태로서 DISCON, CLOSE command가 수행되거나 ARP timeout, TCP timeout이 발생했을 경우 이전 값에 관계없이 상태가 변한다.
0x13	SOCK_INIT	SOCKET *n*이 TCP mode로 open되고 TCP연결의 첫 단계로 initialize된 상태이다. 사용자는 LISTEN과 CONNECT명령을 사용할 수 있다. Sn_MR(P3:P0)이 Sn_MR_TCP이고 OPEN명령을 사용했을 때, Sn_SR의 상태는 SOCK_INIT으로 변한다.
0x14	SOCK_LISTEN	SOCKET *n*이 TCP server mode로 동작하며, ‘TCP CLIENT’로부터 connection-request(SYN packet)를 기다리는 상태다. LISTEN 명령을 사용하면, Sn_SR의 상태는 SOCK_LISTEN으로 변한다. SOCK_LISTEN상태에서 ‘TCP CLIENT’의 Connect-request (SYN packet)를 성공적으로 처리했을 경우 Sn_SR의 상태는 SOCK_ESTABLISHED로 바뀌고, 실패했을 경우 TCP timeout이 발생(Sn_IR(TIME OUT)=‘1’)하고 SOCK_CLOSED로 바뀐다.
0x17	SOCK_ESTABLISHED	TCP 연결이 성립된 상태로서 SOCK_LISTEN상태에서 ‘TCP CLIENT’의 SYN packet 처리를 성공했을 경우나 CONNECT command에 수행이 성공했을 경우 Sn_SR의 상태는 SOCK_ESTABLEISHED로 바뀐다. 이 상태에서는 DATA packet 송수신이 가능하다. 즉 SEND나 RECV command를 수행할 수 있다.

0x1C	SOCK_CLOSE_WAIT	Peer로부터 disconnect-request(FIN packet)를 수신한 상태로서 TCP connection이 완전히 disconnect된 것이 아닌 half close 상태이므로 DATA packet 송수신이 가능하다. TCP connection을 완전히 disconnect 하기 위해서는 DISCON 명령을 수행해야 한다. 하지만 단순히 SOCKET을 close하려면 CLOSE 명령을 수행한다.
0x22	SOCK_UDP	SOCKET *n*이 UDP mode로 Open된 상태, Sn_MR(P3:P0) = Sn_MR_UDP인 상태에서 OPEN 명령이 수행되었을 때 Sn_SR은 SOCK_UDP상태로 바뀐다. TCP mode SOCKET과 달리 connection-process없이 직접 DATA packet을 송수신할 수 있다.
0x32	SOCK_IPRAW	SOCKET *n*이 IPRAW mode로 Open된 상태, Sn_MR(P3:P0) = Sn_MR_IPRAW인 상태에서 OPEN 명령이 수행되었을 때 Sn_SR은 SOCK_IPRAW상태로 바뀐다. UDP mode SOCKET 처럼 connection-process없이 직접 IP packet을 송수신할 수 있다.
0x42	SOCK_MACRAW	SOCKET0이 MACRAW mode로 Open된 상태, S0_MR(P3:P0) = S0_MR_MACRAW인 상태에서 OPEN 명령이 수행되었을 때 Sn_SR은 SOCK_MACRAW상태로 바뀐다. UDP mode SOCKET처럼 connection-process없이 직접 MAC packet (Ethernet frame)을 송수신할 수 있다.
0x5F	SOCK_PPPOE	SOCKET0이 PPPoE mode로 Open된 상태, S0_MR(P3:P0) = S0_MR_PPPoE인 상태에서 OPEN 명령이 수행되었을 때 Sn_SR은 SOCK_PPPOE상태로 바뀐다.

아래 테이블에 Sn_SR의 상태가 변할 때 나타나는 일시적인 상태들을 설명하였다.

Value	Symbol	Description
0x15	SOCK_SYNSENT	SOCK_SYNSENT상태는 'TCP SERVER'에게 Connect-request (SYN packet)를 전송한 상태로서, CONNECT 명령에 의해 Sn_SR의 상태가 SOCK_INIT에서 SOCK_ESTABLISEHD로 바뀔 때 나타난다. 이 상태에서 'TCP SEVER'로부터 Connect-accept (SYN/ACK packet)를 수신할 경우 자동으로 SOCK_ ESTABLISHED상태로 바뀐다. 하지만 'TCP SEVER'로부터 TCP timeout이 발생하기 전까지 (Sn_IR(TIMEOUT)='1') SYN/ACK packet을 수신하지 못할 경우에는 SOCK_CLOSED 상태로 바뀐다.
0x16	SOCK_SYNRECV	SOCK_SYNRECV상태는 'TCP CLIENT'로부터 connect-request

		(SYN packet)를 수신한 상태로서, 이 상태에서 W7100A이 connect-request에 대한 응답으로 connect-accept (SYN/ACK packet)을 'TCP CLIENT'에게 성공적으로 전송하였을 경우에는 자동으로 SOCK_ESTABLISHED상태로 바뀐다. 하지만 전송에 실패하였을 경우 Timeout interrupt가 발생하고 (Sn_IR(TIME OUT)='1') SOCK_CLOSED상태로 바뀐다.
0x18	SOCK_FIN_WAIT	SOCKETn이 Closing되는 상태로서, Active close나 Passive close인 경우의 Disconnect-process에서 나타나는 상태다. Disconnect-process 과정이 성공적으로 완료되거나, Timeout interrupt가 발생하면 (Sn_IR(TIMEOUT)='1') SOCK_CLOSED상태로 변한다.
0x1A	SOCK_CLOSING	
0X1B	SOCK_TIME_WAIT	
0X1D	SOCK_LAST_ACK	Passive close인 경우 송신한 마지막 FIN패킷에 대한 ACK를 기다리는 상태, ACK패킷을 수신한 경우 또는 timeout이 발생한 경우 SOCK_CLOSED상태로 변한다.
0x01	SOCK_ARP	Destination hardware address를 찾기 위해 peer로 ARP-request를 전송하는 상태로서 SOCK_UDP나 SOCK_IPRAW에서 SEND 명령을 수행 할 경우 나타나는 상태다. Peer로부터 Hardware address를 성공적으로 수신한 경우 (ARP-response을 수신한 경우), SOCK_UDP, SOCK_IPRAW, SOCK_SYNSENT로 각각 상태가 변하고, 실패할 경우 timeout interrupt가 발생하고 (Sn_IR(TIMEOUT)='1'), UDP나 IPRAW mode일 경우 이전 Status인 SOCK_UDP나 SOCK_IPRAW로 되돌아 가며, TCP인 경우 SOCK_CLOSED로 상태가 바뀐다. cf> SOCK_UDP나 SOCK_IPRAW에서, 이전 SEND command에 대한 Sn_DIPR와 현재 SEND command의 Sn_DIPR이 다를 경우에만 ARP-process가 동작한다.

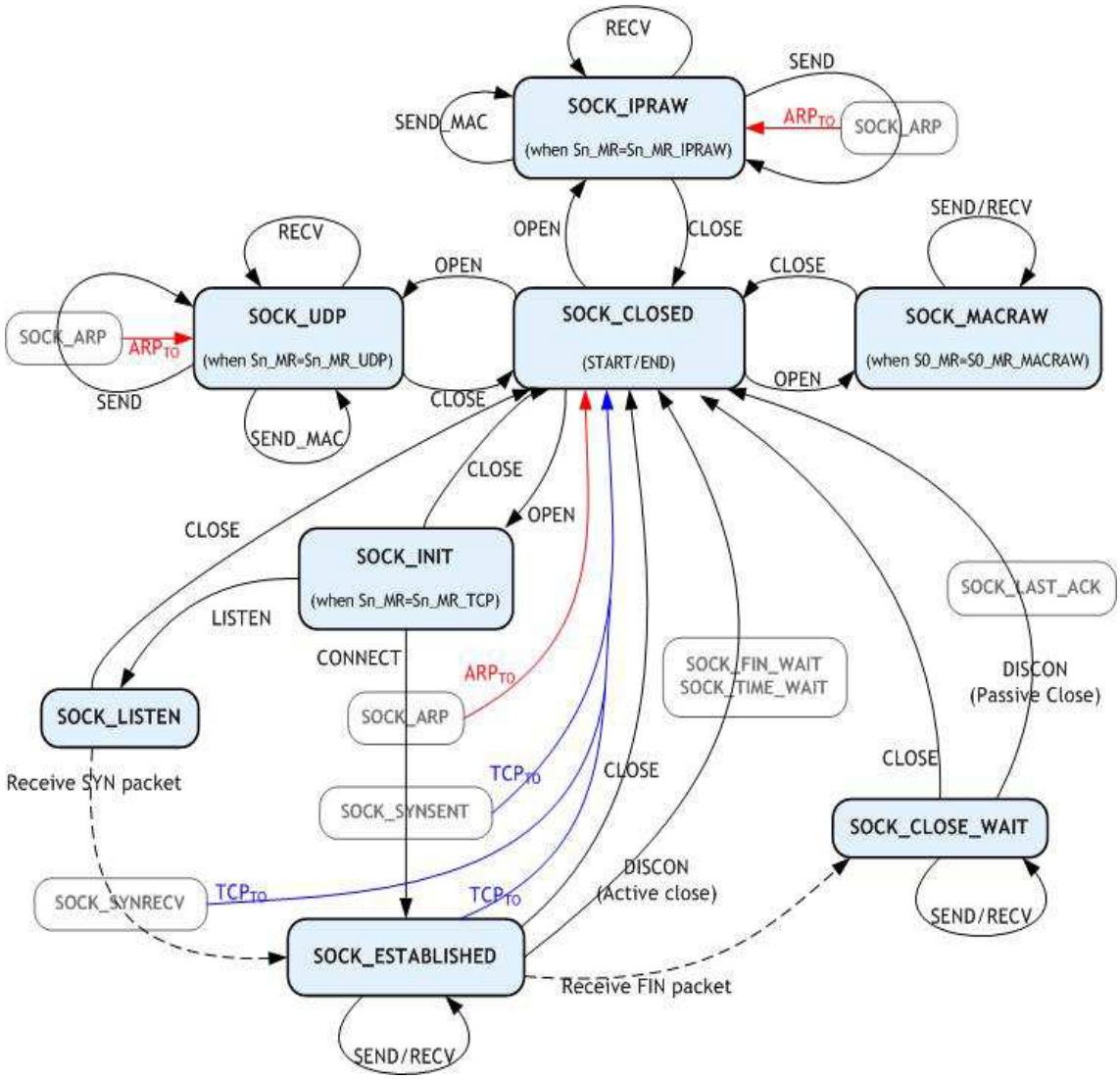


Figure 8.2 SOCKET *n* Status transition

Sn_PORT (SOCKET *n* Source Port Register)[R/W][(0xFE4004 + 0x100n) - (0xFE4005 + 0x100n)][0x0000]

Sn_PORT는 source port number를 설정한다. SOCKETn을 TCP나 UDP mode로 사용할 때만 유효하며, 그 외 mode에서는 무시된다. OPEN Command 이전에 반드시 설정해야 한다.

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.FF]

Sn_DHAR은 SOCKET *n*의 Destination hardware address를 설정한다. 또한 SOCKET0이 PPPoE mode로 사용될 경우 S0_DHAR은 이미 알고 있는 PPPoE server hardware address로 설정한다.

UDP나 IPRAW mode에서 SEND_MAC command를 사용할 경우 SOCKETn의 Destination hardware address를 설정한다. 또한 TCP, UDP, IPRAW mode에서 Sn_DHAR은 CONNECT나

SEND command에 의한 ARP-process를 통해 획득한 Destination hardware address로 설정된다. Host는 CONNECT나 SEND command 성공 이후 Sn_DHAR을 통해 Destination hardware address를 알 수 있다.

PPPoE mode에서, W7100A의 PPPoE-process를 이용할 경우 PPPoE server hardware address를 따로 설정할 필요는 없다. 하지만 W7100A의 PPPoE-process를 이용하지 못하고 MACRAW mode로 PPPoE-process를 직접 구현하여 처리한 경우라 할지라도, PPPoE packet을 송수신하기 위해서는, 직접 구현한 PPPoE-process를 통해 획득한 PPPoE server hardware address, PPPoE server IP address, PPP session ID를 설정하고 MR(PPPoE)를 '1'로 반드시 설정한다. S0_DHAR은 이미 알고 있는 PPPoE server hardware address를 설정하며, OPEN command 이전에 설정한다. S0_DHAR을 통해 설정된 PPPoE server hardware address는 OPEN command 이후 PDHAR에 반영된다. 설정된 PPPoE information은 CLOSE command 이후에도 계속 유효하다.

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]

Sn_DIPR은 SOCKETn의 Destination IP address를 설정하거나 설정되며, SOCKET0이 PPPoE mode로 사용될 경우 S0_DIPR은 이미 알고 있는 PPPoE server IP address로 설정한다.

Sn_DIPR은 TCP, UDP, IPRAW, PPPoE mode에서만 유효하고, MACRAW mode에서는 무시된다. TCP mode에서, 'TCP CLIENT'로 동작할 경우 접속하기 위한 'TCP SERVER'의 IP address로 설정하고, CONNECT command 이전에 설정한다. 'TCP SERVER'로 동작할 경우 'TCP CLIENT'와 접속 성공 이후 자동으로 'TCP CLIENT'의 IP address로 설정된다. UDP나 IPRAW mode에서는, Sn_DIPR은 UDP나 IP DATA packet 전송에 사용될 Destination IP address로 SEND나 SEND_MAC command 이전에 설정한다. PPPoE mode에서는, S0_DHAR과 같은 경우로 S0_DIPR는 이미 알고 있는 PPPoE server IP address를 설정한다.

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 + 0x100n) - (0xFE4011 + 0x100n)][0x0000]

Sn_DIPR은 SOCKETn의 Destination port number를 설정한다. 만약 SOCKET0이 PPPoE mode로 사용되는 경우 S0_DPORTR은 이미 알고 있는 PPP Session ID로 설정한다.

Sn_DIPR은 TCP, UDP, PPPoE mode에서만 유효하고, 그 외의 mode에서는 무시된다. TCP mode에서, 'TCP CLIENT'로 동작할 경우 접속하기 위한 'TCP SERVER'의 Listen port number로

설정하고, CONNECT command 이전에 설정한다. UDP mode에서, Sn_DPORTR은 UDP DATA packet 전송에 사용될 Port number로 SEND나 SEND_MAC command 이전에 설정한다. PPPoE mode에서는, S0_PDHAR과 같은 경우로 S0_DPORTR는 이미 알고 있는 PPP Session ID를 설정한다. S0_DPORTR을 통해 설정된 PPP Session ID는 OPEN command 이후 PSIDR에 반영된다.

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 + 0x100n) - (0xFE4013 + 0x100n)][0x0000]

Sn_MSSR은 SOCKETn의 MTU(Maximum Transfer Unit)를 설정하거나, 설정된 MTU를 알려준다. TCP나 UDP mode만 지원하며, PPPoE를 사용할 경우(MR(PPPoE)='1') PPPoE의 MTU내에서 TCP나 UDP mode의 MTU가 결정된다. IPRAW나 MACRAW는 자동으로 MTU를 처리하지 않고 Default MTU가 적용되므로, Host는 Default MTU보다 큰 Data를 전송할 때 Data를 Default MTU 단위로 직접(Manually) 나누어 전송해야 한다.

Reset시 초기값은 0이지만, 소켓 초기화 과정에서 사용자가 설정한 값과 default MTU값 중 작은 값으로 MSSR값이 결정된다. 사용자가 설정한 값이 없다면 default 값으로 설정된다.

TCP나 UDP mode에서는 Host가 전송할 Data가 설정된 MTU보다 클 경우, W7100A는 설정된 MTU 단위로 Data를 자동으로 나누어 전송한다. MTU는 TCP mode에서 MSS라 불리며, MSS는 TCP connection 과정을 통해 Host-Written-Value(Host 설정 값)와 상대방의 MSS 값 중 작은 값으로 자동으로 설정된다.

UDP mode에서는 TCP mode와 같은 Connection-process가 없고 Host-Written-Value를 그대로 사용한다. MTU가 서로 다른 상대방과 통신 할 경우, W7100A는 ICMP(Fragment MTU) packet을 수신할 수 있다. 이 경우 IR(FMTU)='1'가 되고 Host는 FMTUR을 통해 Fragment MTU를 알 수 있다. IR(FMTU)='1'일 경우 그 상대방과는 UDP 통신이 불가능하므로, 해당 SOCKET을 close하고 알아낸 FMTU를 Sn_MSSR로 설정한 후 OPEN command로 open하여 다시 통신을 시도한다.

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 + 0x100n][0x00]

Sn_PROTO는 1 byte register로 IP layer에서 IP header의 Protocol number field를 설정한다. IPRAW mode에서만 유효하며, 그 외 mode는 무시된다. Sn_PROTOR은 OPEN command 이전에 설정한다. IPRAW mode로 Open된 SOCKETn은 Sn_PROTOR에 설정된 Protocol number의 Data 만을 송수신한다. Sn_PROTOR은 0x00 ~ 0xFF 의 범위 내에서 설정 가능하나, W7100A는 TCP(0x06), UDP(0x11) protocol number는 지원하지 않는다. Protocol number는 IANA (Internet Assigned Numbers Authority)에서 정의하고 있으며, 더 자세한 사항은 IANA의 online document (http://www.iana.org/assignments/protocol-numbers)를 참조하기 바란다.

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

Sn_TOS (SOCKET *n* TOS Register)[R/W][0xFE4015 + 0x100n][0x00]

Sn_TOS는 IP layer에서 IP header의 TOS(Type of service) field를 설정한다. Sn_TOS는 OPEN command 이전에 설정해야 한다. 자세한 사항은 http://www.iana.org/assignments/ip-parameters 참조하기 바란다.

Sn_TTL (SOCKET *n* TTL Register)[R/W][0xFE4016 + 0x100n][0x80]

Sn_TTL은 IP layer에서 IP header의 TTL(Time to live) field를 설정한다. Sn_TTL은 OPEN command 이전에 설정해야 한다. 자세한 사항은 http://www.iana.org/assignments/ip-parameters 참조하기 바란다.

Sn_RXMEM_SIZE (SOCKET *n* Receive Memory Size Register)[R/W][0xFE401E + 0x100n][0x02]

Sn_RXMEM_SIZE는 각 SOCKET의 RX memory size를 설정한다. 각 SOCKET의 RX memory 1, 2, 4, 8, 16Kbyte크기로 설정할 수 있다. Reset후에 초기값으로 2Kbyte의 값을 갖는다. $Sn\_RXMEM\_SIZE_{SUM}$(각 Sn_RXMEM_SIZE의 총 합)은 최대 16Kbyte를 넘을 수 없다.

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 (Sn_RXMEM_SIZE$_{SUM}$) is 16Kbytes.

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

Sn_TXMEM_SIZE는 각 SOCKET의 TX memory size를 설정한다. 각 SOCKET의 TX memory 1, 2, 4, 8, 16Kbyte크기로 설정할 수 있다. Reset후에 초기값으로 2Kbyte의 값을 갖는다. Sn_TXMEM_SIZE$_{SUM}$(각 Sn_TXMEM_SIZE의 총 합)은 최대 16Kbyte를 넘을 수 없다.

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 + 0x100n) - (0xFE4021 + 100n)][0x0000]

Sn_TX_FSR은 SOCKET *n*의 Internal TX memory의 Free size(전송 가능한 데이터의 Byte size)를 알려준다. 데이터 전송 전에 Sn_TX_FSR를 반드시 확인하고, 전송할 데이터의 크기가 Sn_TX_FSR보다 작거나 같으면 SEND나 SEND_MAC command로 데이터를 전송한다. TCP mode에서는 상대방으로부터 데이터 수신이 확인(DATA/ACK packet 수신)되면, Sn_TX_FSR은 상대방이 수신한 DATA packet 크기만큼 자동으로 증가하게 된다. 그 외 mode에서는 Sn_IR(SENDOK) = '1'인 경우 Sn_TX_FSR은 전송한 Data size만큼 자동으로 증가하게 된다.

Ex) In case of 2048(0x8000) in S0_TX_FSR,

0xFE4020	0xFE4021
0x08	0x00

Sn_TX_RD (SOCKET *n* TX Read Pointer Register)[R][(0xFE4022 + 0x100n) - (0xFE4023 + 0x100n)][0x0000]

Sn_TX_RD는 TX memory에서 마지막 전송이 끝날 때 address를 알려준다. SOCKET *n* Command Register의 SEND 명령으로 현재 Sn_TX_RD부터 Sn_TX_WR까지 데이터를 전송한다. 전송이 끝나면 자동으로 그 값이 갱신된다. 따라서 전송이 끝나면 Sn_TX_RD와 Sn_TX_WR은 같은 값을 가질 것이다. 이 register를 읽을 때, 사용자는 정확한 값을 얻기 위해 upper byte (0xFE4022, 0xFE4122, 0xFE4222, 0xFE4322, 0xFE4422, 0xFE4522, 0xFE4622, 0xFE4722) 들을 먼저 읽어야 하고 다음 lower byte (0xFE4023, 0xFE4123, 0xFE4223, 0xFE4323, 0xFE4423, 0xFE4523, 0xFE4623, 0xFE4723) 들을 읽어야 한다.

Sn_TX_WR (SOCKET *n* TX Write Pointer Register)[R/W][(0xFE4024 + 0x100n) - (0xFE4025 + 0x100n)][0x0000]

Sn_TX_WR은 전송할 데이터가 write되어야 할 위치정보를 알려준다. 이 register를 읽을 때, 사용자는 정확한 값을 얻기 위해 upper byte (0xFE4024, 0xFE4124, 0xFE4224, 0xFE4324, 0xFE4424, 0xFE4524, 0xFE4624, 0xFE4724) 들을 먼저 읽어야 하고 다음 lower byte (0xFE4025, 0xFE4125, 0xFE4225, 0xFE4325, 0xFE4425, 0xFE4525, 0xFE4625, 0xFE4725) 들을 읽어야 한다.

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

0xFE4024	0xFE4025
0x08	0x00

하지만 이 값은 write할 physical address가 아니므로, physical address는 다음과 같이 계산해야 한다. (W7100A Driver code 참조)

1. Sn_TXMEM_SIZE(*n*)에서 SOCKET *n* TX Base Address (SBUFBASEADDRESS(*n*))와 SOCKET *n* TX Mask Address (SMASK(*n*))를 계산한다.
2. 위에서 계산한 두 값을 bitwise-AND operation하고, SOCKET의 TX memory 범위에서 Sn_TX_WR과 SMASK(*n*)를 통해 offset address (dst_mask)를 얻는다.

3. Dst_mask와 SBUFBASEADDRESS(*n*)를 더해서 physical address (dst_ptr)를 얻는다.

이제, 전송할 데이터를 dst_ptr에 사용자가 원하는 만큼 write한다. (만약 SOCKET의 TX memory의 upper bound이상으로 데이터를 write하는 경우, TX memory의 upper bound만큼의 데이터를 먼저 write 한다. 그리고 난 다음 SBUFBASEADDRESS(*n*)에 physical address를 변경하여 나머지 데이터를 write한다. 그 후, Sn_TX_WR값을 writing 데이터 크기만큼 증가시킨다. 마지막으로 Sn_CR (SOCKET *n* Command Register)에 SEND명령을 내린다.

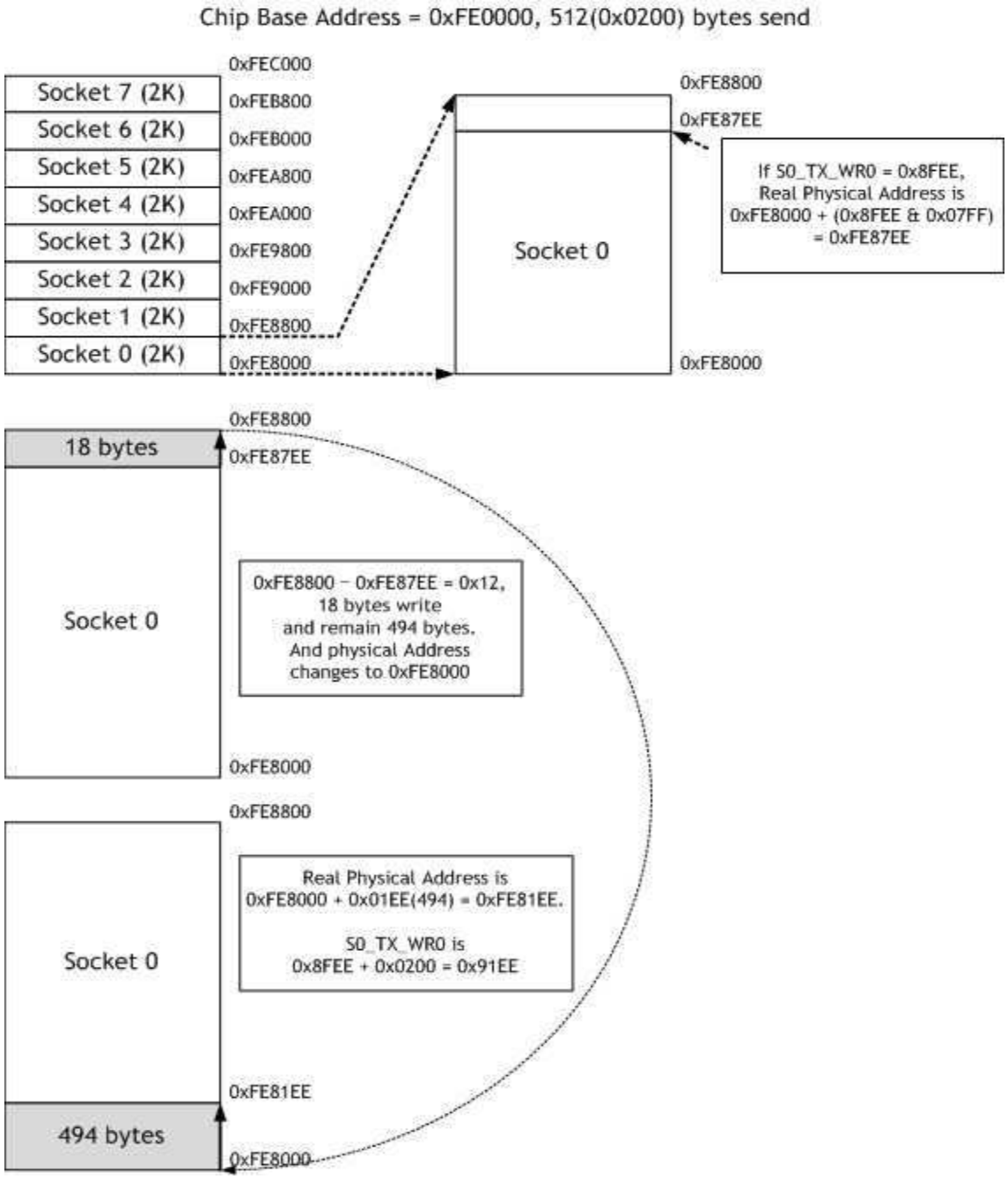


Figure 8.3 Calculate Physical Address

Sn_RX_RSR (SOCKET *n* RX Received Size Register)[R][(0xFE4026 + 0x100n) - (0xFE4027 + 0x100n)][0x0000]

Sn_RX_RSR은 SOCKETn의 Internal RX memory의 수신데이터 byte size를 알려준다. 이 값은

SOCKET *n* Command Register (Sn_CR) RECV 명령어에 의해 자동으로 변하고 remote peer로부터 데이터를 수신한다. 이 register를 read할 때, 정확한 값을 얻기 위해 사용자는 상위 byte (0xFE4026, 0xFE4126, 0xFE4226, 0xFE4326, 0xFE4426, 0xFE4526, 0xFE4626, 0xFE4726)를 먼저 read 하고 그 다음 하위 byte (0xFE4027, 0xFE4127, 0xFE4227, 0xFE4327, 0xFE4427, 0xFE4527, 0xFE4627, 0xFE4727)를 read한다.

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

0xFE4026	0xFE4027
0x08	0x00

Sn_RX_RSR 값의 총합은 RX memory size register의 값에 따라 달라질 수 있다.

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

Sn_RX_RD는 수신 데이터를 read하기 위한 pointer의 위치 정보를 제공한다. 이 register를 read할 때, 사용자는 정확한 값을 read하기 위해 상위 byte (0xFE4028, 0xFE4128, 0xFE4228, 0xFE4328, 0xFE4428, 0xFE4528, 0xFE4628, 0xFE4728)를 먼저 read한 다음 하위 byte (0xFE4029, 0xFE4129, 0xFE4229, 0xFE4329, 0xFE4429, 0xFE4529, 0xFE4629, 0xFE4729)를 read해야 한다.

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

0x0428	0x0429
0x08	0x00

하지만 이 값은 read할 physical address가 아니므로, physical address는 다음과 같이 계산해야 한다. (W7100A Driver code 참조)

1. Sn_RXMEM_SIZE(*n*)에서 SOCKET *n* RX Base Address (RBUFBASEADDRESS(*n*))와 SOCKET *n* RX Mask Address (RMASK(*n*))가 계산된다.
2. 위에서 계산한 두 값을 bitwise-AND operation하고, SOCKET의 RX memory 범위에서 Sn_RX_WR과 RMASK(*n*)를 통해 offset address (src_mask)를 얻는다.
3. src_mask와 RBUFBASEADDRESS(*n*)를 더해서 physical address (src_ptr)를 얻는다.

이제, 수신 데이터를 src_ptr부터 사용자가 원하는 만큼 read한다. (만약 SOCKET의 RX memory의 upper bound이상으로 데이터를 read하는 경우, RX memory의 upper bound만큼의 데이터를 먼저 read 한다. 그리고 난 다음 RBUFBASEADDRESS(*n*)에 physical address를 변경하여 나머지 데이터를 read한다. 그 후, Sn_RX_RD값을 read한 만큼 증가시킨다. (수신된 데이터 크기 이상으로 증가해서는 안되므로, 수신 전에 반드시 Sn_RX_RSR을 check해야 한다.) 마지막으로 Sn_CR (SOCKET *n* Command Register)에 RECV명령을 내린다.

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

Sn_RX_WR은 수신 데이터를 write하기 위한 pointer의 위치 정보를 제공한다. 이 register를

read할 때, 사용자는 정확한 값을 read하기 위해 상위 byte (0xFE402A, 0xFE412A, 0xFE422A, 0xFE432A, 0xFE442A, 0xFE452A, 0xFE462A, 0xFE472A)를 먼저 read하고 난 다음 하위 byte (0xFE402B, 0xFE412B, 0xFE422B, 0xFE432B, 0xFE442B, 0xFE452B, 0xFE462B, 0xFE472B)를 read 해야 한다.

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

0xFE402A	0xFE402B
0x08	0x00

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

Sn_FRAG는 IP layer에서 IP header의 Fragment field를 설정한다. W7100A는 IP layer의 packet fragment를 지원하지 않는다. 따라서 Sn_FRAG를 설정하더라도 IP data는 fragment되지 않으며 이를 설정하는 것은 권장하지 않는다. Sn_FRAG는 OPEN command 이전에 설정한다.

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

0xFE402D	0xFE402E
0x40	0x00

9 Functional Description

W7100A는 내부에 general 8051 core와 TCPIPCore를 포함하고 있어, 다른 어떠한 추가 장치 없이 Ethernet application에 사용이 가능하다. 이 chapter에서는 W7100A의 초기화와 각 Protocol(TCP, UDP, IPRAW, MACRAW)에 따른 통신 방법에 대하여 각 단계별로 Pseudo code와 함께 살펴본다.

9.1 Initialization

W7100A의 초기화는 8051 MCU 설정, Network 정보 설정, Internal TX/RX memory 설정의 3 단계로 이루어 진다.

- **STEP 1 : Initializes MCU**

1. Interrupt setting

   일반적인 8051과 같이 사용할 인터럽트의 enable / disable의 상태를 설정해야 한다. 자세한 내용은 section 3 ‘ Interrupt’를 참고하면 된다.

2. Memory Access timing setting

   Memory access timing은 2개의 레지스터로 설정할 수 있다. Data memory access timing을 조절할 수 있는 CKCON(0x8E) 레지스터와 Program memory access timing을 조절할 수 있는 WTST(0x92) 레지스터의 설정을 통해 memory access timing을 설정할 수 있다. 이 두 레지스터는 모두 0~7까지 설정이 가능하지만 W7100A에서는 CKCON은 1~7, WTST는 4~7의 값이 설정 가능하고, 나머지 값은 사용되지 않는다. 만약 사용자가 위의 값보다 작은 값으로 설정을 한다면 W7100A이 제대로 동작하지 않을 수 있다. 자세한 내용은 section 2.5 ‘SFRs definition'을 참고하면 된다.

   Ex) Setting: 인터럽트를 사용하지 않고, 외부 데이터 메모리와 2 clock의 access time을 갖고, 프로그램 메모리와는 7 clock의 access time을 갖게 설정

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

3. 시리얼 통신을 위한 baud rate, register, interrupt 설정

1) Serial 통신을 위해 설정해 줘야 하는 W7100A의 레지스터는 TMOD, PCON, SCON이며 구성은 아래와 같다.

① TMOD(89H): serial 통신에 사용할 timer/counter의 모드를 결정한다.

GATE	C/T	M_1	M_0	GATE	C/T	M_1	M_0

Table 9.1 Timer / Counter Mode

M_1	M_0	Mode
0	0	0
0	1	1
1	0	2
1	1	3

② PCON(87H): serial 전송의 rate를 제어하는 플래그인 SMOD bit를 결정한다.

SMOD	-	-	-	-	-	-	-

Table 9.2 Baud rate

Mode	SMOD = '0'	SMOD = '1'
1, 3	Timer/Counter 1의 overflow를 1/2	Timer/Counter 1의 overflow
2	XTAL을 1/4	XTAL을 1/2

③ SCON(98H): serial port의 제어와 serial port의 상태를 감시하기 위한 레지스터이다.

SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

Table 9.3 Mode of UART

SM_0	SM_1	Mode
0	0	0
0	1	1
1	0	2
1	1	3

SM_2 : Mode 2, 3에서 사용. 이 bit를 1로 설정하면, 수신 데이터의 9번bit가 '1'이면 데이터를 수신, '0'이면 데이터를 무시한다.

REN : 수신 enable bit('1'이면 수신 가능)

TB_8 : 모드 2, 3에서 송신 데이터의 8번 bit

RB_8 : 모드 2, 3에서 수신 데이터의 8번 bit

TI : 송신 완료 인터럽트 플래그

RI : 수신 완료 인터럽트 플래그

2) Serial 통신을 초기화할 때 인터럽트의 상태를 설정해 주어야 한다. 기본적으로 serial 통신이 인터럽트 방식을 사용하므로, serial 통신을 초기화할 때 반드시 해당되는 인터럽트들을 disable해 주어야 한다.

3) 사용할 Baud-rate에 맞는 값을 계산하여 설정해야 한다. W7100A의 Timer에 따른 Baud

rate 설정 값은 section 6.6 'Examples of Baud Rate Setting'을 참고하여 설정하면 된다.

① Timer1을 이용한 계산식

TH1 = 256 - ((K * 88.4736MHz) / (384 * baud rate))

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

② Timer2를 이용한 계산식

(RCAP2H, RCAP2L) = 65536 - (88.4736MHz / (32 * baud rate))

Ex) Using timer mode 2, SMOD = 1, Clock speed = 88.4736MHz, Baud rate = 115200.

```
ET1= 0;          // Timer1 INT disable
TMOD = 0x20;     // TIMER MODE 2
PCON |= 0x80;    // SMOD = 1
TH1 = 0xFC;      // x2 115200(SMOD = 1) at 88.4736MHz
TR1 = 1;         // Start the TIMER1
SCON = 0x50;     // Serial MODE 1, REN = 1, TI = 0, RI = 0
ES = 0;          // Serial interrupt disable
RI = 0;          // Receive interrupt disable
TI = 0;          // Transmit interrupt disable
```

4) TCPIP Core interrupt를 사용한다면, INTLEVEL register값을 최소 0x2B00 이상으로 설정해야만 W7100A의 내부 interrupt 루틴이 문제없이 동작한다.

Ex) Set the INTLEVEL register to 0x2B00

```
IINCHIP_WRITE (INTLEVEL0,  0x2B);  //write high byte of INTLEVEL TCPIPCore register
IINCHIP_WRITE (INTLEVEL0 + 1,  0x00); //write low byte of INTLEVEL TCPIPCore register
```

- **STEP 2 : Setting Network Information**

1. 통신을 위한 기본 Network 정보 설정:

   다음의 기본적인 Network 정보를 반드시 설정해 주어야 한다.

   ① SHAR(Source Hardware Address Register)

   SHAR에 의해 설정되는 Source hardware address는 모든 Device에 대해 유일한 Hardware address(Ethernet MAC address)값을 Ethernet MAC layer에서 사용하도록 정해져 있다. 이 MAC address의 할당은 IEEE에서 관장하고 있으며, Network device를 생산하는 Manufacture는 생산된 Network device에 IEEE로부터 할당 받은 MAC address를 부여하여야 한다.

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

   ② GAR(Gateway Address Register)

   ③ SUBR(Subnet Mask Register)

   ④ SIPR(Source IP Address Register)

2. Packet 전송을 실패 시 사용하게 될 재전송 time & count 설정

재전송 시간의 설정을 위해 다음과 같은 레지스터를 설정해 주어야 한다.

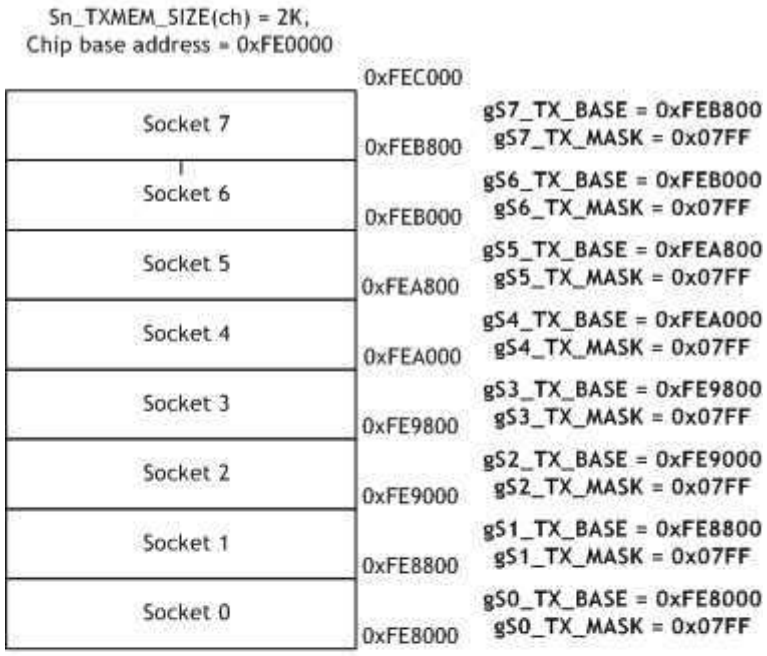
① RTR(Retry Time-value Register), RTR에서 1은 100us를 의미한다.

② RCR(Retry Count Register)

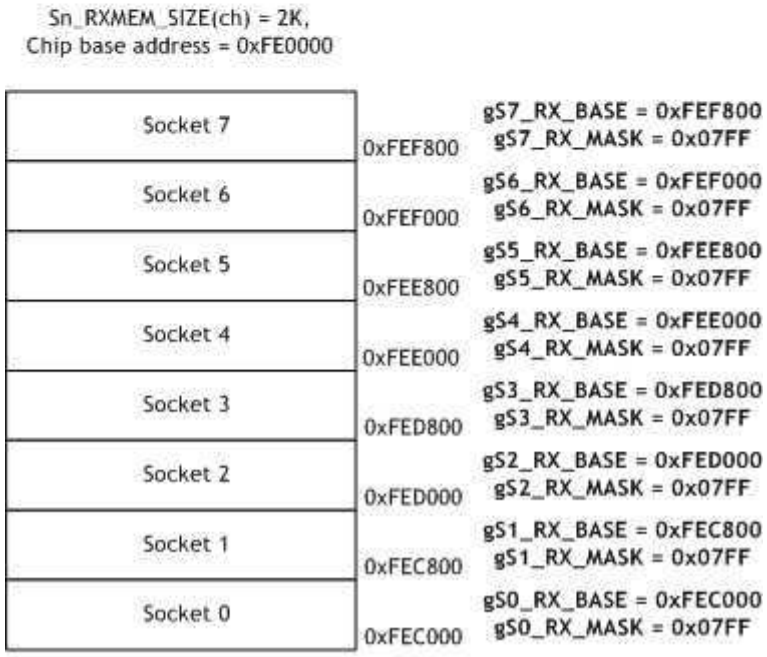
- **STEP 3 : Allocation TX/RX Memory for SOCKET *n***

W7100A의 설정 가능한 TX, RX의 최대 메모리 사이즈는 16KBytes이다. 16Kbytes의 범위 안에서는 1KB, 2KB, 4KB, 8KB, 16KB의 크기로 8개의 소켓까지 자유롭게 설정이 가능하지만, 소켓 별로 할당한 TX혹은 RX메모리의 총 합이 16Kbyte를 넘어가서는 안 된다. (TX_{max} = 16KB, RX_{max} = 16KB)

```
In case of, assign 2KB rx, tx memory per SOCKET
{// Set base address of RX memory for SOCKET 0
gS0_RX_BASE = 0xFE0000(Chip base address) + 0xFEC000(RX buffer address);
Sn_RXMEM_SIZE(ch) = (uint8 *) 2;   // Assign 2K rx memory per SOCKET
gS0_RX_MASK = 2K - 1; //0x07FF, offset address within assigned SOCKET0 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(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.
}
```



(a) TX memory allocation



(b) RX memory allocation

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

3 단계의 W7100A Initialization 과정이 성공적으로 끝났다면, W7100A는 Ethernet을 통해 Data communication이 가능하다. 이 시점부터 W7100A는 network으로부터 수신한 Ping-request packet에 대한 Ping-reply를 전송할 수 있게 된다.

9.2 Data Communication

Data 통신을 하기 위해서 W7100A initialization 과정 후 TCP, UDP, IPRAW, MACRAW mode중 사용하고자 하는 mode로 SOCKET을 open한다. W7100A는 독립적으로 동시에 사용 가능한 SOCKET을 총 8개까지 지원한다. 이 section에서는 각 mode에 따른 통신 방법에 대해서 설명한다.

9.2.1 TCP

TCP는 Connection-oriented protocol이다. TCP는 자신의 IP address와 Port number 그리고 상대방의 IP address와 Port number를 한 쌍으로 Connection SOCKET을 형성하게 되고, 형성된 Connection SOCKET을 통해 Data를 송수신한다. Connection SOCKET의 형성 방법에는 'TCP SERVER'와 'TCP CLIENT' 2가지가 있다. 이는 어디에서 connect-request(SYN packet)을 전송하느냐에 따라 구분할 수 있다.

'TCP SERVER'은 상대방의 connect-request 전송을 대기하며, 전송된 connect-request을 accept하여 Connection SOCKET을 형성한다(Passive-open).

'TCP CLIENT'는 자신이 connect-request를 상대방에게 전송하여 Connection SOCKET 형성을 먼저 요구한다(Active-open).

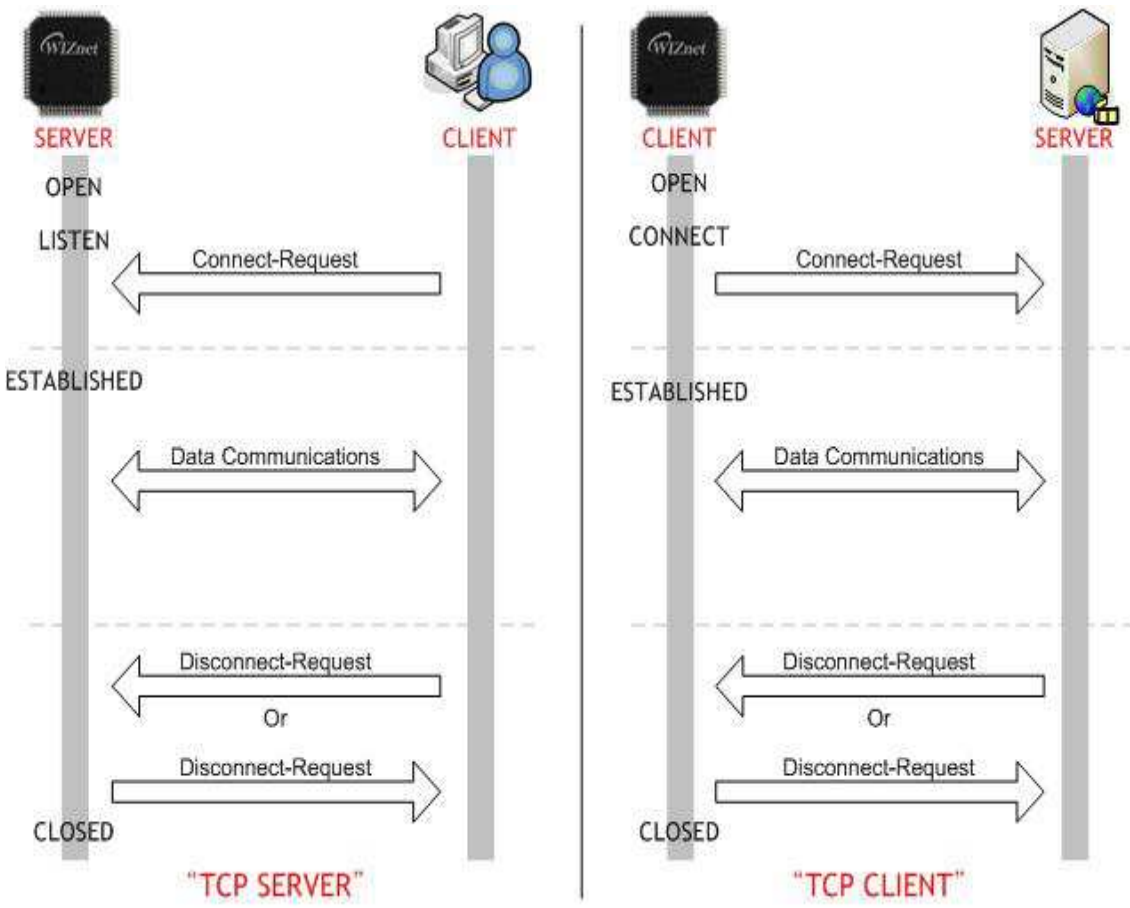


Figure 9.2 TCP SERVER & TCP CLIENT

9.2.1.1 TCP SERVER

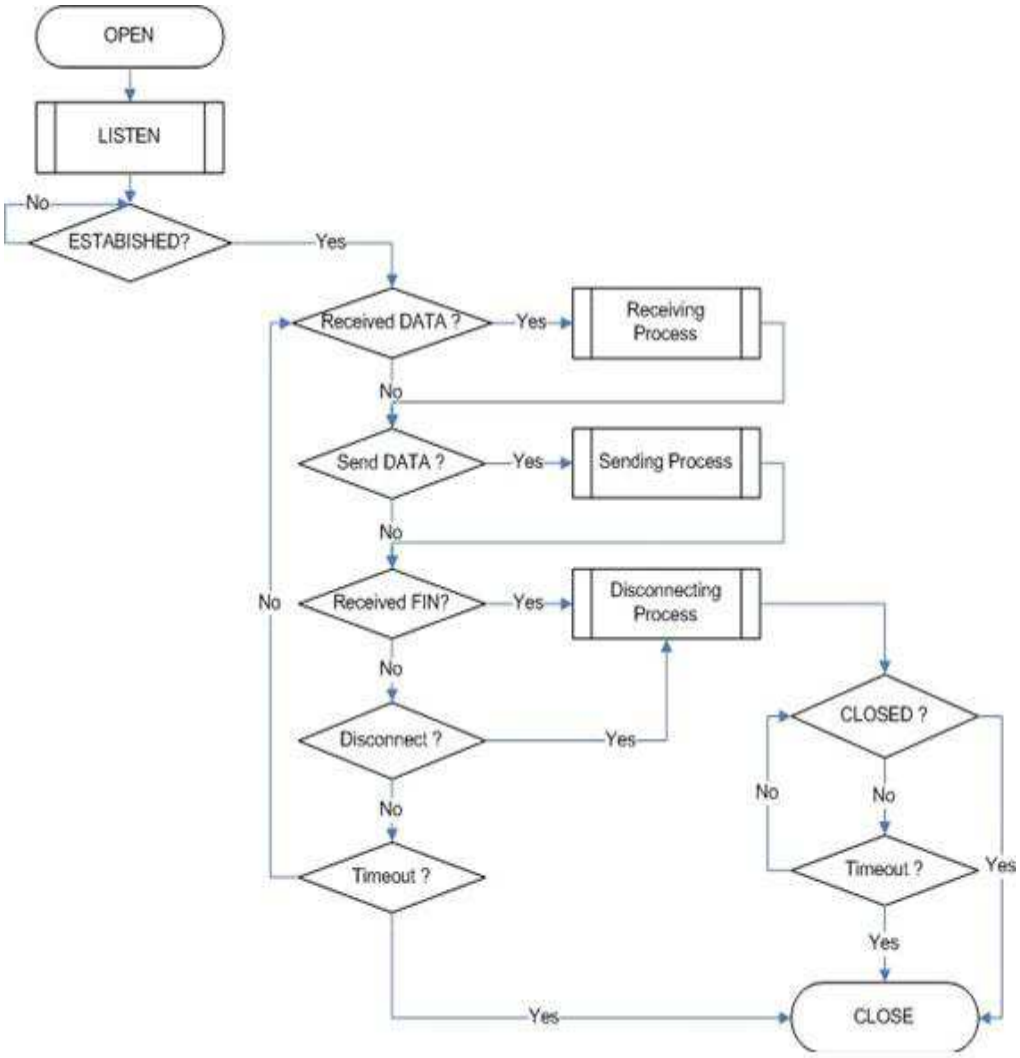


Figure 9.3 "TCP SERVER" Operation Flow

■ SOCKET Initialization

TCP Data communication을 위해 SOCKET Initialization 과정이 필요하다. 이 과정은 SOCKET을 open하는 일이다. SOCKET open 과정은 W7100A의 8개의 SOCKET 중 하나를 선택하여 선택된 SOCKET의 Protocol mode(Sn_MR(P3:P0))와 Source port number("TCP SERVER"에서는 Listen port number라고 함)인 Sn_PORT0을 설정한 후, OPEN command를 수행함으로써 이루어진다. OPEN command 이후 Sn_SR이 SOCK_INIT으로 변경되면 SOCKET initialization 과정은 완료된다. SOCKET initialization 과정은 "TCP SEVER"와 "TCP CLIENT"의 구분 없이 동일하게 적용된다. 다음은 Socket n을 TCP mode로 open하는 과정이다.

```
{
START:
  Sn_MR = 0x0001;                  // sets TCP mode
  Sn_PORT = 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

LISTEN command를 수행하여 "TCP SERVER"로 동작시킨다.

```
{  /* 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

Sn_SR이 SOCK_LISTEN일 때 상대방으로부터 SYN packet을 수신하게 되면 Sn_SR은 SOCK_SYNRECV로 바뀌고 SYN/ACK packet을 전송 한다. SYN/ACK packet에 대한 ACK packet을 수신하면 connection이 형성되고 Sn_SR은 SOCK_ESTABLISHED로 바뀐다. Socket n의 connection이 형성된 이후부터 data 송/수신이 가능해진다. Socket n의 connection 형성을 확인하는 방법은 2가지가 있다.

```
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: 수신 데이터 check

상대방으로부터의 TCP data 수신을 확인한다.

```
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_RSR != 0x00000000) goto Receiving Process stage;
}
```

First method는 매 수신 DATA packet 마다 Sn_IR(RECV)이 '1'로 설정된다. Host가 이전에 수신한 DATA packet의 Sn_IR(RECV)를 미처 처리 못하고 W7100A이 다음 DATA packet을 수신할 경우, 이전 Sn_IR(RECV)에 중복 설정되어 Host는 그 다음의 수신 DATA packet에 대한 Sn_IR(RECV)를 인지할 수가 없게 된다. 따라서 Host가 각 Sn_IR(RECV)에 대한 DATA packet을 완벽하게 처리하지 못한다면 이 방법은 권장하지 않는다.

■ ESTABLISHMENT: Receiving process

이 과정에서는 내부 RX memory에 수신된 TCP 데이터를 처리한다. TCP mode에서 상대방이 전송한 Data 크기가 Socket n의 RX memory free size보다 클 경우 W7100A는 그 data를 수신할 수 없으며, RX memory free size가 전송한 데이터 크기보다 클 때까지 connection을 유지한 체 기다린다.

Receive / Send process에서 사용되는 wizmemcpy 함수는 WIZnet에서 제공하는 W7100A Driver파일 중 wizmemcpy.c 파일에 정의되어있다. wizmemcpy는 memory copy를 빠르게 처리하기 위한 기능이다. 이 함수의 성능에 대해서는 section 13 'Performance Improvement about W7100A'를 참조하기 바란다. 또 한 wizmemcpy.c 파일에 대한 보다 자세한 내용은 "W7100A Driver Guide"를 참조하기 바란다. 만약 wizmemcpy 함수를 사용하지 않는다면 일반 memory copy 함수를 사용하면 된다.

그림 2.6에서 알 수 있듯이 TCPIPCore용 RX 메모리의 최상위 address byte는 0xFE다. 따라서 TCPIPCore RX 메모리에서 data memory로 memory copy를 진행할 때는 반드시 DPX0 (Data Pointer eXtended) 레지스터를 0xFE로 설정해 주어야 한다. 이 과정은 wizmemcpy()함수에 구현되어 있다.

```
{    /* first, get the received size */
    len = Sn_RX_RSR;      // 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

전송할 data 크기는 할당된 Socket n의 TX memory보다 클 수 없으며, 전송할 data 크기가 설정된 MSS보다 클 경우 MSS 단위로 나뉘어져 전송된다. 다음 data를 전송하기 위해선 반드시 이전의 SEND command가 완료되었는지 확인해야 한다. 이전 SEND command 완료 전에 다시 SEND command를 수행할 경우 오류가 발생할 수 있다. Data의 크기가 클수록 SEND command 완료 시간도 길어지므로, 전송 Data를 적정한 크기로 나누어 전송하는 것이 유리하다.

Send process도 receive process와 마찬가지로 DPX0 register를 “0xFE”로 설정해야 한다.

```
{/* first, get the free TX memory size */
FREESIZE:
    freesize = Sn_TX_FSR;
    if (freesize < len) goto FREESIZE;     // len is send size
    /* calculate offset address */
    dst_mask= Sn_TX_WR & 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_WR += send_size;
    /* set SEND command */
    Sn_CR = SEND;
}
```

■ ESTABLISHMENT: Check disconnect-request(FIN packet)

상대방으로부터 disconnect-request(FIN packet)를 수신했는지 확인한다. FIN packet 수신은 다음과 같이 확인할 수 있다.

```
First method :
{
    if (Sn_IR(DISCON) == '1') Sn_IR(DISCON)='1'; goto DISCONNECT 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 DISCONNECT stage;
}
```

■ ESTABLISHMENT : Check disconnect / disconnecting process

더 이상 상대방과의 data communication이 필요가 없는 경우나 상대방으로부터 FIN packet을 수신했을 경우는 connection SOCKET을 disconnect한다.

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

■ ESTABLISHMENT: Check closed

DISCON이나 CLOSE command에 의해 Socket n이 Disconnect 혹은 close 되었는지 확인한다.

```
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

Timeout은 connect-request(SYN packet)나 그것에 대한 응답(SYN/ACK packet), DATA packet이나 그것의 응답(DATA/ACK packet), disconnect-request(FIN packet)나 그것의 응답(FIN/ACK packet)등, 모든 TCP packet을 전송할 때 발생할 수 있다. RTR과 RCR에 설정된 Timeout 시간 동안 위 packet들을 전송하지 못하면 TCP final timeout(TCP timeout)이 발생하게 되고 Sn_SR은 SOCK_CLOSED로 바뀐다. TCP timeout의 확인은 다음과 같이 할 수 있다.

```
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

Disconnect-process에 의해 이미 disconnection된 SOCKET이나 TCP timeout에 의해 Close된 SOCKET을 완전히 close하거나, Host가 disconnect-process없이 필요에 의해 SOCKET을 close 할 경우 사용할 수 있다.

```
{
    Sn_IR = 0x00FF;      /* clear the remained interrupts of SOCKET n*/
    IR(n) = '1';
```

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

9.2.1.2 TCP CLIENT

TCP client는 CONNECT state를 제외한 모든 state가 TCP SEVER와 동일하다. 자세한 내용은 '9.2.1.1 TCP SERVER'를 참조하기 바란다.

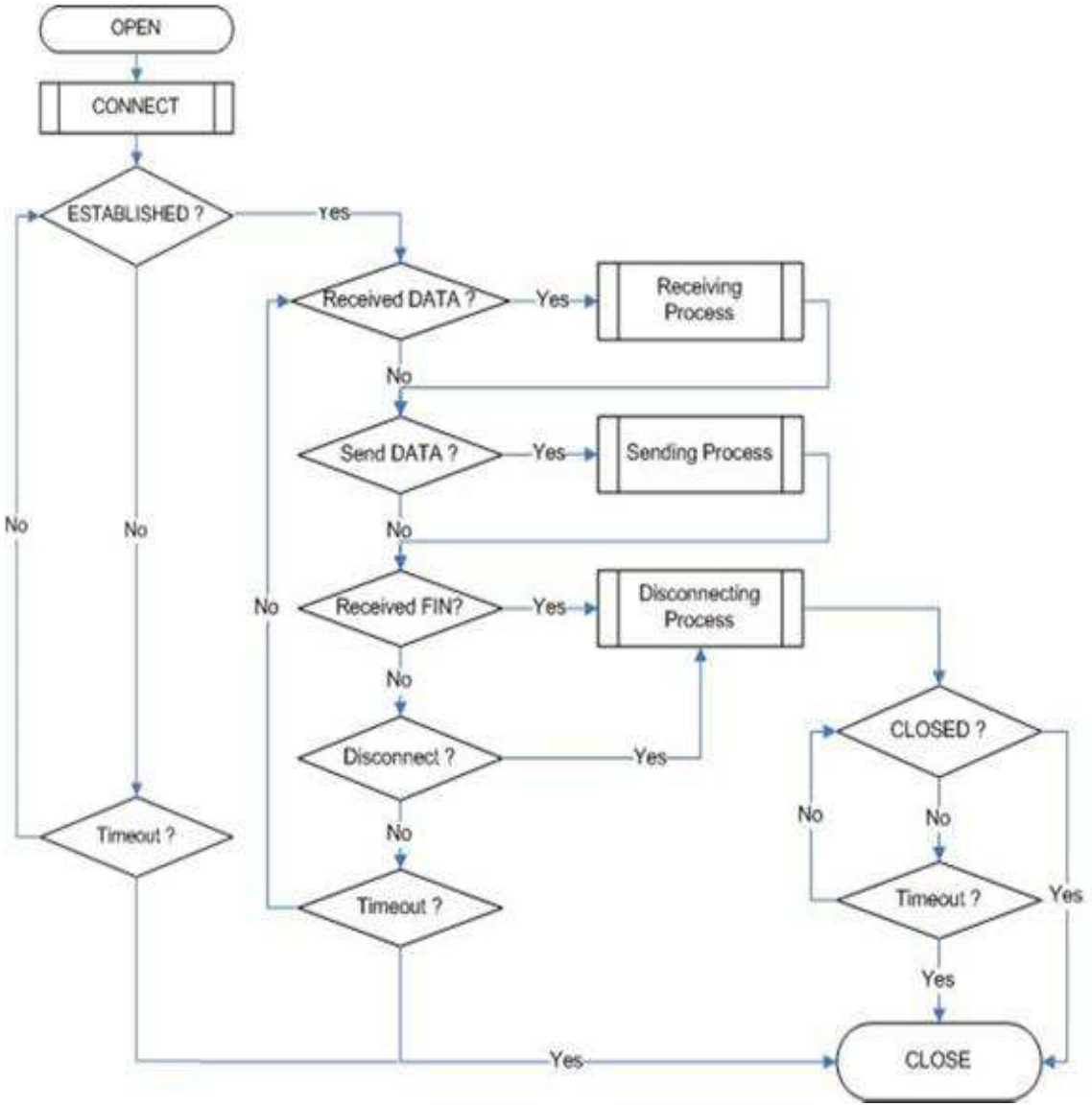


Figure 9.4 "TCP CLIENT" Operation Flow

■ CONNECT

'TCP SERVER'에게 connect-request (SYN packet)를 전송한다. 'TCP SERVER'와의 Connection SOCKET 형성 과정에서 ARP timeout, TCP timeout과 같은 Timeout이 발생할 수 있다.

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

9.2.2 UDP

UDP는 Connection-less protocol이다. UDP는 TCP와 달리 connection 을 형성하지 않고 data 를 송수신한다. TCP는 신뢰성 있는 data 통신을 보장하는 반면, UDP는 data 통신의 신뢰성을 보장하지 않는 datagram 통신을 하는 protocol이다. UDP는 connection 을 사용하지 않기 때문에 동시에 자신의 IP address와 port number를 알고 있는 많은 상대방과의 통신이 허락된다. 이와 같은 datagram 통신은 하나의 SOCKET을 이용하여 많은 상대방과 통신을 할 수 있는 이점이 있는 반면, 전송된 data의 손실이나 원치 않는 상대방으로부터의 data 수신과 같은 여러 문제가 발생할 수 있다. 이와 같은 문제를 해결하고 신뢰성을 보장하기 위해서, Host가 직접 손실된 data를 재전송하거나, 원치 않는 상대방으로부터의 수신 data를 무시해야 한다. UDP 통신은 unicast, broadcast, multicast 통신을 지원하며, 다음과 같은 통신 flow 를 따른다.

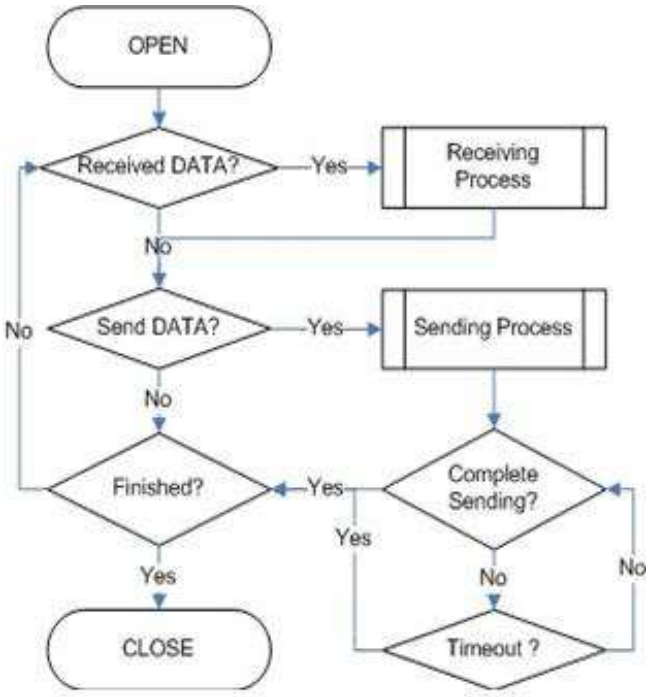


Figure 9.5 UDP Operation Flow

9.2.2.1 Unicast & Broadcast

Unicast 통신은 가장 일반적인 UDP 통신으로, 한번에 하나의 상대방에게 Data를 전송한다. 반면, Broadcast 통신은 Broadcasting IP address를 이용하여 한번의 통신으로 수신 가능한 모든 상대방에게 Data를 전달한다. 예로 A, B, C 에게 Data를 전송할 경우, Unicast 통신은 A, B, C 각각에 대해서 한번씩 Data를 전송을 한다. 이 때 A, B, C에 대한 Destination hardware address를 획득하는 과정(ARP-process)에서 ARP timeout이 발생할 수 있으며, ARP timeout이 발생한 상대방에게는 Data를 전송할 수가 없다.

Broadcast 통신은 Broadcasting IP address로 한번의 Data 전송을 통하여 A, B, C 모두에게 동시에 Data를 전달한다. 이 때 A, B, C에 대한 Destination hardware address를 획득할 필요

가 없으며, ARP timeout 역시 발생하지 않는다.

- Broadcast IP만드는 방법

=> HOST IP와 Subnet Mask의 보수를 OR 연산하면 Broadcast IP가 생성된다.

ex> IP: 222.98.173.123, Subnet Mask: 255.255.255.0 이면 broadcast IP: 222.98.173.255

Description	Decimal	Binary
HOST IP	222.098.173.123	11011110.01100010.10101101.01111011
Bit Complement Subnet mask	000.000.000.255	00000000.00000000.00000000.11111111
Bitwise OR	-	-
Broadcast IP	222.098.173.255	11011110.01100010.10101101.11111111

■ SOCKET Initialization

UDP data communication을 위해 SOCKET initialization 과정이 필요하다. 이는 SOCKET을 open하는 일이다. SOCKET open 과정은 W7100A의 8개의 SOCKET 중 하나를 선택하고, 선택된 SOCKET의 protocol mode(Sn_MR(P3:P0))와 상대방과의 통신에 사용할 source port number인 Sn_PORT0을 설정한 후, OPEN command를 수행한다. OPEN command 이후 Sn_SR이 SOCK_UDP으로 변경되면 SOCKET initialization 과정은 완료된다.

```
{
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

상대방으로부터의 UDP data 수신을 확인한다. TCP 통신과 동일한 방법으로 확인이 가능하다. 물론 TCP와 같은 이유로 Second method를 권장한다. “9.2.1.1 TCP SERVER”의 해당 절을 참조하라.

```
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_RSR != 0x00000000) goto Receiving Process stage;
}
```

■ Receiving process

이 과정에서는 RX memory에 수신된 UDP Data를 처리한다. 수신된 UDP data의 구조는 아래와 같다.

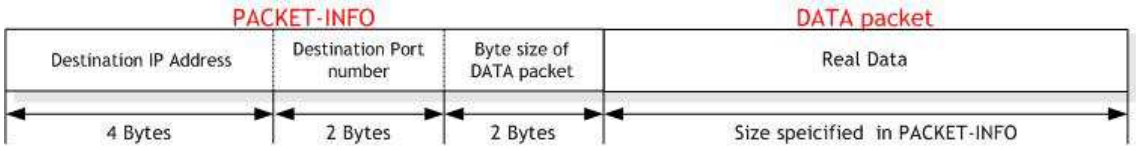


Figure 9.6 The received UDP data format

수신된 UDP data는 8bytes의 PACKET-INFO와 data packet으로 이루어지며, PACKET-INFO는 송신자의 정보(IP address, Port number)와 data packet의 길이가 포함된다. UDP는 많은 송신자로부터 UDP data를 수신할 수 가 있다. 송신자의 구분은 PACKET-INFO의 송신자 정보를 통해 확인할 수 있다. 송신자가 Broadcasting IP address를 이용하여 broadcast한 경우도 수신된다. Host는 PACKET-INFO의 송신자 정보를 분석하여 필요 없는 수신 data packet은 무시해야 한다.

송신자의 data 크기가 SOCKET의 RX memory free size보다 클 경우 그 data는 수신할 수 없으며, 또한 fragment된 data 역시 수신할 수 없다.

```
{/* 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_address */
        upper_size = (gSn_RX_MASK + 1) - src_mask;
        wizmemcpy((0xFE0000 + src_ptr), (0x000000 + header), upper_size);
        /* update header_addr*/
        header_addr += upper_size;
        /* copy left_size bytes of gSn_RX_BASE to header_address */
        left_size = header_size - upper_size;
        wizmemcpy((0xFE0000 + gSn_RX_BASE), (0x000000 + header), left_size);
        /* update src_mask */
        src_mask = left_size;
    }
    else
```

```
    {/* copy header_size bytes of get_start_address to header_address */
        wizmemcpy((0xFE0000 + src_ptr), (0x000000 + header), 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 + get_size) > (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 + get_size + header_size;
    /* set RECV command */
    Sn_CR = RECV;
}
```

■ Check send data / Sending process

전송할 data크기는 할당된 SOCKET의 TX memory보다 클 수 없으며, 전송할 data 크기가 MTU보다 클 경우 MTU 단위로 자동으로 나누어져 전송된다. Broadcast할 경우에는 Sn_DIPR0 을 Broadcasting IP로 설정한다.

```
{/* first, get the free TX memory size */
FREESIZE:
```

```
    freesize = Sn_TX_FSR;
    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_DIPR = remote_ip;
    Sn_DPORT = remote_port;
    /* calculate offset address */
    dst_mask = Sn_TX_WR & 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_WR as length of len */
    Sn_TX_WR += len;
    /* set SEND command */
    Sn_CR = SEND;
}
```

■ Check complete sending / Timeout

다음 Data를 전송하기 위해선 반드시 이전 SEND command가 완료되었는지 확인해야 한다. Data의 크기가 클수록 SEND command 완료 시간도 길어지므로, 전송 Data를 적정한 크기로 나누어 전송하는 것이 유리하다. UDP data 전송 시 ARP timeout이 발생할 수 있고, ARP timeout이 발생할 경우 UDP data 전송은 실패한다.

```
First method :
{/* check SEND command completion */
```

```
  while(Sn_IR(SENDOK)==‘0’)   /* wait interrupt of SEND completion */
  {
    /* check ARP timeout */
    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

통신이 모두 끝난 경우 Socket *n*을 close한다.

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

9.2.2.2 Multicast

Broadcast 통신이 불특정 다수와 통신하는 반면, multicast 통신은 특정 multicast-group에 등록된 다수와 통신을 한다. A, B, C가 특정 multicast-group에 등록되어 있고, A가 등록된 multicast-group으로 data를 전송할 경우 B, C 역시 A의 전송 data를 수신할 수 있다. multicast 통신을 하기 위해선 IGMP protocol을 이용하여 multicast-group에 등록하여야 한다. multicast-group은 group hardware address, Group IP address, group port number로 구분된다. Group hardware address와 IP address는 이미 지정되어 있는 address를 사용하고, group port number는 임의로 사용할 수 있다.

Group hardware address는 지정 범위 (‘01:00:5e:00:00:00’에서부터 ‘01:00:5e:7f:ff:ff’) 내에서 선택되며, Group IP address는 D-class IP address 범위 (“224.0.0.0”에서 ‘239.255.255.255’까지, http://www.iana.org/assignments/multicast-addresses참조)내에서 선택된다. 이때 6bytes의 group hardware address와 4bytes의 IP address의 하위 23bit는 같도록 선택해야 한다. 예로, Group IP address를 ‘224.1.1.11’로 선택할 경우 group hardware address

는 '01:00:5e:01:01:0b'로 선택된다. 'RFC1112' 참조 (http://www.ietf.org/rfc.html).

W7100A에서는 multicast-group 등록에 필요한 IGMP 처리는 자동으로 이루어진다. SOCKET *n*을 multicast mode로 open할 경우 IGMP의 'Join' message, close할 경우 'Leave' message가 자동으로 전송된다. SOCKET open 이후 통신 시 주기적으로 'Report' message가 자동으로 전송된다.

W7100A는 IGMP version 1과 version 2만을 지원하며 상위 version을 사용하고자 한다면, IPRAW mode SOCKET을 이용하여 host가 직접 IGMP를 처리해야 한다.

■ SOCKET Initialization

Multicast 통신을 위해 8개의 SOCKET 중 하나를 선택하고, Sn_DHAR0을 multicast-group hardware address로 Sn_DIPR0을 multicast-group IP address로 설정한다. Sn_PORT0과 Sn_DPORT0을 multicast-group port number로 설정한다. Sn_MR(P3:P0)를 UDP로 Sn_MR (MULTI)를 '1'로 설정한 후 OPEN command를 수행한다. OPEN command 이후 Sn_SR이 SOCK_UDP으로 변경되면 SOCKET initialization 과정은 완료된다.

```
{
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_DPORT = 0x0BB8;      /* set Multicast-Group Port number(3000) */
    Sn_PORT = 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

Section 9.2.2.1 'Unicast & Broadcast' 참조.

■ Receiving process

Section 9.2.2.1 'Unicast & Broadcast' 참조.

■ Check send data / Sending Process

SOCKET initialization에서 이미 multicast-group에 대한 정보를 설정하였으므로, unicast통신처럼 상대방의 IP address와 port number를 설정할 필요가 없다. 따라서 전송할 data를 TX memory로 copy한 후 SEND command를 수행한다.

```
{/* first, get the free TX memory size */
FREESIZE:
    freesize = Sn_TX_FSR;
    if (freesize < len) goto FREESIZE;       // len is send size
    /* calculate offset address */
    dst_mask = Sn_TX_WR & 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_WR += send_size;
    /* set SEND command */
    Sn_CR = SEND;
}
```

■ Check complete sending / Timeout

Data 통신에 필요한 모든 Protocol 처리는 Host가 관장하므로 Timeout은 발생하지 않는다.

```
{/* check SEND command completion */
  while(Sn_IR(SENDOK)==‘0’);   /* wait interrupt of SEND completion */
  Sn_IR(SENDOK) = ‘1’;         /* clear previous interrupt of SEND completion */
}
```

■ Check finished / SOCKET close

Section 9.2.2.1 ‘Unicast & Broadcast’ 참조.

9.2.3 IPRAW

IPRAW는 TCP와 UDP의 하위 protocol 계층인 IP layer를 이용한 Data 통신이다. IPRAW는 protocol number에 따라 ICMP(0x01), IGMP(0x02)와 같은 IP layer의 protocol을 지원한다. ICMP의 ping이나 IGMP v1/v2는 W7100A에서 Hardware logic으로 이미 구현되어있다. 하지만 필요에 따라 Host는 Socket n을 IPRAW mode로 open하여 이를 직접 구현하여 처리할 수 있다. IPRAW mode SOCKET을 사용할 경우, 어떤 protocol을 사용할지 반드시 IP header의 protocol number field를 설정하여야 한다. Protocol number는 IANA에 의해 이미 정의되어 있다(http://www.iana.org/assignments/protocol-numbers 참조). Protocol number는 SOCKET Open 이전에 Sn_PROTO에 반드시 설정한다. W7100A는 IPRAW mode에서 TCP(0x06)나 UDP(0x11) protocol number는 지원하지 않는다. IPRAW mode SOCKET의 통신은 지정된 protocol number만의 통신을 허용한다. ICMP로 설정된 SOCKET은 IGMP와 같이 설정되지 않은 그 외의 Protocol Data를 수신할 수 없다.

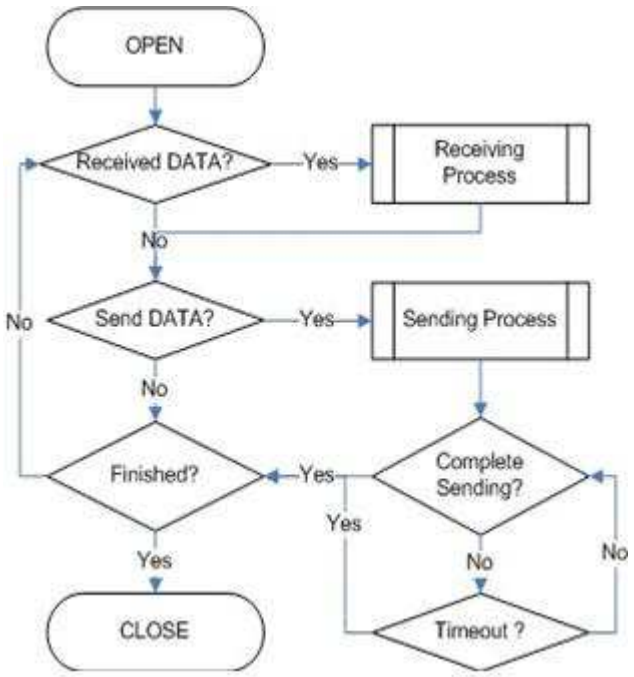


Figure 9.7 IPRAW Operation Flow

■ SOCKET Initialization

SOCKET을 선택하고 Protocol number를 설정한다. Sn_MR (P3:P0)를 IPRAW mode로 설정하고 OPEN command를 수행한다. OPEN command 이후 Sn_SR이 SOCK_IPRAW로 변경되면 SOCKET initialization 과정은 완료된다.

```
{
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

Section 9.2.2.1 ‘Unicast & Broadcast’ 참조.

■ Receiving process

RX Memory에 수신된 IPRAW Data를 처리한다. 수신된 IPRAW Data의 구조는 아래와 같다.

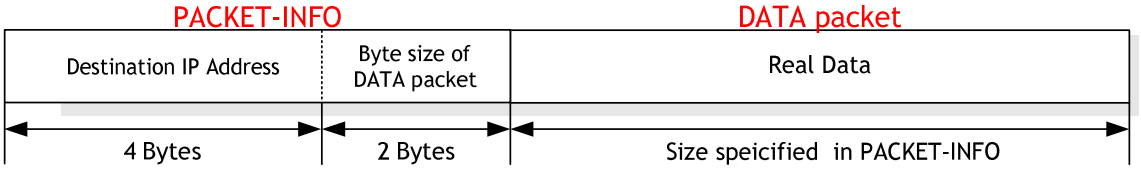


Figure 9.8 The received IPRAW data format

IPRAW data는 6 bytes의 PACKET-INFO와 data packet으로 이루어지며, PACKET-INFO는 송신자의 정보(IP address)와 data packet의 길이가 포함된다. IPRAW mode의 data 수신은 UDP의 PACKET-INFO에서 송신자의 port number 처리를 제외하고는 UDP data 수신과 모두 동일하다. section 9.2.2.1 'Unicast & Broadcast' 참조.

송신자의 Data 크기가 SOCKET *n*의 RX memory free size보다 클 경우 그 data는 수신할 수 없으며, 또한 fragmented data 역시 수신할 수 없다.

■ Check send data / Sending process

전송할 data 크기는 할당된 SOCKET *n*의 TX memory보다 클 수 없고, default MTU보다 클 수 없다. IPRAW data 전송은 UDP data 전송에서 destination port number를 설정하는 것을 제외하고 모두 동일하다. 자세한 사항은 section 9.2.2.1 'Unicast & Broadcast'를 참조하기 바란다.

■ Complete sending / Timeout

UDP와 동일, section 9.2.2 'UDP'참조.

■ Check finished / SOCKET closed

UDP와 동일, section 9.2.2 'UDP' 참조.

9.2.4 MACRAW

MACRAW 통신은 Ethernet MAC을 기반으로 그 상위 protocol을 host가 목적에 맞게 유연하게 사용할 수 있도록 하는 통신방법이다.

MACRAW mode는 오직 SOCKET 0만 사용 가능하다. SOCKET 0를 MACRAW로 사용할 경우 SOCKET 1에서 7까지는 hardwired TCP/IP stack을 그대로 사용할 뿐만 아니라, SOCKET 0를 마치 NIC(Network Interface Controller)처럼 사용할 수 있어 software TCP/IP stack을 구현할 수 있다. 이와 같이 W7100A는 hardwired TCP/IP와 software TCP/IP를 모두 구현할 수 있는 hybrid TCP/IP stack을 지원한다. W7100A이 지원하는 8개의 SOCKET보다 더 많은 SOCKET들이 요구될 경우, 높은 성능을 요구하는 SOCKET들은 hardwired TCP/IP stack으로 구현하고, 그 외는 MACRAW mode를 이용하여 software TCP/IP로 구현하여 SOCKET 수의 한계를 극복할 수 있다. MACRAW mode의 SOCKET 0는 SOCKET 1에서 7까지 사용되고 있는 protocol들을 제외한 모든 protocol들을 처리할 수 있다. MACRAW 통신은 아무런 처리 없이 순수 Ethernet packet만의 통신이므로 MACRAW 설계자는 이러한 protocol을 분석하고 처리할 수 있는 software TCP/IP stack를 직접 구현해야 한다. MACRAW data는 Ethernet MAC을 기반으로 하기 때문에 6bytes의 source hardware address, 6bytes의 destination hardware address, 2 bytes

의 Ethernet type 총 14bytes을 기본으로 포함해야 한다.

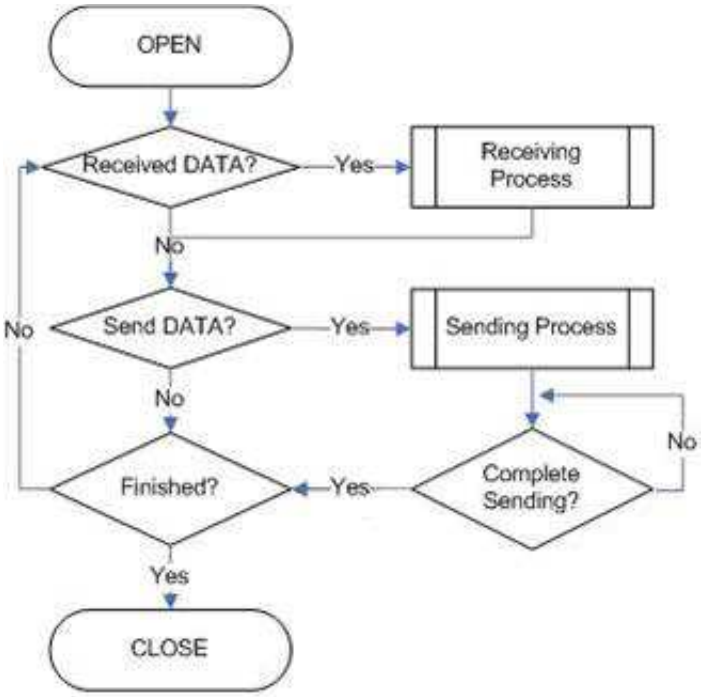


Figure 9.9 MACRAW Operation Flow

■ SOCKET Initialization

SOCKET을 선택하고 Sn_MR(P3:P0)를 MACRAW mode로 설정한 후 OPEN command를 수행한다. OPEN command 이후 Sn_SR이 SOCK_MACRAW로 변경되면 SOCKET initialization 과정은 완료된다. 이때 통신에 필요한 모든 정보 (Source hardware address, Source IP address, Source port number, Destination hardware address, Destination IP address, Destination port number, 각종 Protocol header, ETC)는 MACRAW Data의 일부분이므로 이와 관련된 별도의 register 설정은 필요 없다.

```
{
START:
    /* sets MAC raw mode */
    S0_MR = 0x04;
    /* sets OPEN command */
    S0_CR = OPEN;
    /* wait until Sn_SR is changed to SOCK_MACRAW */
    if (S0_SR != SOCK_MACRAW) S0_CR = CLOSE; goto START;
}
```

■ Check received data

Section 9.2.2.1 'Unicast & Broadcast' 참조.

■ Receiving process

SOCKET 0의 RX memory에 수신된 MACRAW data를 처리한다. MACRAW data의 구조는 Figure 9.11과 같다.

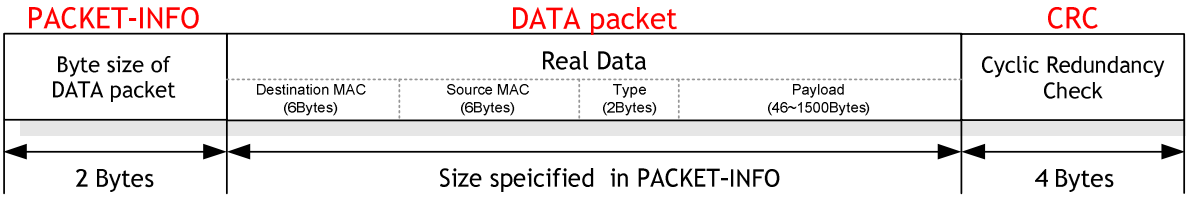


Figure 9.10 The received MACRAW data format

MACRAW data는 2 bytes의 PACKET-INFO, data packet, 4bytes의 CRC로 이루어진다. PACKET-INFO는 data packet의 길이이며, data packet은 6bytes destination MAC address, 6bytes source MAC address, 2bytes type, 46~1500 bytes payload로 이루어진다. Data packet의 Payload는 Type에 따라 ARP, IP와 같은 Internet protocol로 이루어진다. Type에 관한 정보는 http://www.iana.org/assignments/ethernet-numbers 를 참조하기 바란다.

```
{/* calculate offset address */
    src_mask = S0_RX_RD & gS0_RX_MASK;       // src_mask is offset address
    /* calculate start address(physical address) */
    src_ptr = gS0_RX_BASE + src_mask;       // src_ptr is physical start address
    /* get the size of packet */
    len = get_Byte_sizeof_DATA_packet();       // Read the 2bytes PACKET-INFO
    /* if overflow SOCKET RX memory */
    If((src_mask + len) > (gS0_RX_MASK + 1))
    {   /* copy upper_size bytes of get_start_address to destination_address */
       upper_size = (gS0_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 */
    S0_RX_RD += len;
    /* extract 4 bytes CRC from RX memory and then ignore it */
```

```
    wizmemcpy((0xFE0000 + src_ptr), (0x000000 + dummy), len);
    /* set RECV command */
    S0_CR = RECV;
}
```

<Notice>

RX memory의 free size가 W7100A이 수신해야 할 MACRAW data의 크기보다 작을 경우, 수신되어서는 안 되는 MACRAW data의 PACKET-INFO와 data packet의 일부가 RX memory에 저장되는 문제가 간혹 발생할 수 있다. 이는 상기 sample code에서 PACKET-INFO 분석의 오류를 야기시켜 올바른 MACRAW data 수신 처리를 할 수 없게 된다. 이 문제는 RX memory가 Full에 가까울수록 발생할 확률이 높아진다. 이 문제는 MACRAW data의 소실을 어느 정도 감안한다면 아래와 같이 해결할 수 있다.

- RX memory의 처리를 최대한 빨리 하여 Full에 도달하는 것을 방지한다.
- SOCKET Initialization 과정의 Sample code에서 S0_MR의 MF(MAC Filter) bit를 설정하여 자신에 해당하는 MACRAW data만을 수신하도록 하여 수신부하를 줄인다.

```
{
START:
/* sets MAC raw mode with enabling MAC filter */
S0_MR = 0x44;
/* sets OPEN command */
S0_CR = OPEN;
/* wait until Sn_SR is changed to SOCK_MACRAW */
if (S0_SR != SOCK_MACRAW) S0_CR = CLOSE; goto START;
}
```

- RX memory의 free size가 1528 - default MTU(1514)+PACKET-INFO(2) + data packet(8) + CRC(4) - 보다 작을 경우 SOCKET0을 close한 후 지금까지 수신한 모든 MACRAW data를 처리하고 다시 SOCKET 0를 open하여 정상 처리한다. 이때 SOCKET 0 close이후 수신되는 MACRAW data는 손실될 수도 있다.

```
{/* check the free size of RX memory */
  if((S0_RXMEM_SIZE(0) * 1024) - S0_RX_RSR(0) < 1528)
  {
      recved_size = S0_RX_RSR(0);  /* backup Sn_RX_RSR */
      S0_CR = CLOSE;               /* SOCKET Closed */
      while(S0_SR != SOCK_CLOSED);          /* wait until SOCKET is closed */
      /* process all data remained in RX memory */
      while(recved_size > 0)
      {/* calculate offset address */
          src_mask = S0_RX_RD & gS0_RX_MASK;     // src_mask is offset address
```

```
            /* calculate start address(physical address) */
            src_ptr = gS0_RX_BASE + src_mask;    // src_ptr is physical start address
            /* if overflow SOCKET RX memory */
            If((src_mask + len) > (gS0_RX_MASK + 1))
            {
            /* copy upper_size bytes of get_start_address to destination_address */
               upper_size = (gS0_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 */
            S0_RX_RD += len;
            /* extract 4 bytes CRC from RX memory and then ignore it */
            wizmemcpy((0xFE0000 + src_ptr), (0x000000 + dummy), len);
            /* calculate the size of remained data in 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 (S0_SR != SOCK_MACRAW);
      }
      else /* process normally the DATA packet from RX memory */
      {/* This block is same as the code of “Receiving process” stage*/
      }
}
```

■ Check send data / Sending process

전송할 Data 크기는 할당된 SOCKET 0의 TX memory보다 클 수 없고, 또한 default MTU보다 클 수 없다. Host는 'Receiving process'과정과 같이 data packet 형식과 동일한 MACRAW data를 생성하고 그 data를 전송한다. 이 때 생성된 data의 크기가 60 bytes 미만일 경우 실제 Ethernet으로 전송되는 packet은 자동으로 60bytes가 되도록 'Zero padding'하여 전송한다.

```
{/* first, get the free TX memory size */
FREESIZE:
    freesize = S0_TX_FSR;
    if (freesize < send_size) goto FREESIZE;
    /* calculate offset address */
    dst_mask = S0_TX_WR & gS0_TX_MASK;       // dst_mask is offset address
    /* calculate start address(physical address) */
    dst_ptr = gS0_TX_BASE + dst_mask;       // dst_ptr is physical start address
    /* if overflow SOCKET TX memory */
    if ( (dst_mask + len) > (gS0_TX_MASK + 1) )
    {/* copy upper_size bytes of source_addr to dst_ptr */
        upper_size = (gS0_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 + gS0_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 */
    S0_TX_WR += send_size;
    /* set SEND command */
    S0_CR = SEND;
}
```

■ Check complete sending

Data 통신에 필요한 모든 protocol 처리는 host가 관리하기 때문에 timeout은 발생하지 않지 않는다.

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

■ Check finished / SOCKET close

Section 9.2.2.1 ‘Unicast & Broadcast’ 참조.

10 Electrical Specification

10.1 Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	-0.5 to 3.6	V
V_{IN}	DC input voltage	-0.5 to 5.5 (5V tolerant)	V
V_{OUT}	DC output voltage	2 to 3.3 (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	-40 to 85	°C
T_{STG}	Storage temperature	-55 to 125	°C

***COMMENT**: Device에 'Absolute Maximum Ratings'를 넘어서는 스트레스를 가할 경우 심각한 damage의 원인이 될 수 있다. Simultaneously

10.2 DC Characteristics(Input, Output, I/O)

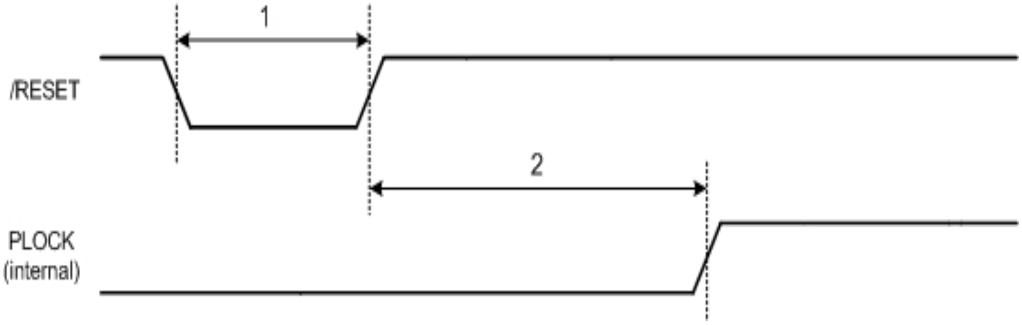
Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
V_{DD}	DC Supply voltage	Junction temperature -55°C ~ 125°C	3.0	3.3	3.6	V
V_{IH}	High level input voltage		2.0		5.5	V
V_{IL}	Low level input voltage		-0.5		0.8	V
V_{OH}	High level output voltage	I_{OH} = 8 mA	2.4			V
V_{OL}	Low level output voltage	I_{OL} = 8 mA			0.4	V
I_{lkg}	Input Leakage Current	V_{IN} = V_{DD} or 0	-10	±1	10	μA
	Input Leakage Current with pull-up resistance	V_{IN} = 0	-15	-45	-85	μA
	Input Leakage Current with pull-down resistance	V_{IN} = V_{DD}	15	45	85	μA
I_{OZ}	Tri-state output leakage current	V_{OUT} = V_{DD}	2		8	μA
I_{OL}	Low level output current	V_{OL} = 0.8V, 25°C, V_{DD} = 3.3V		52		mA
I_{OH}	High level output current	V_{OL} = 2.4V, 25°C, V_{DD} = 3.3V		52		mA

10.3 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
$I_{Power-down}$	Current consumption	Power-down mode	108	mA

10.4 AC Characteristics

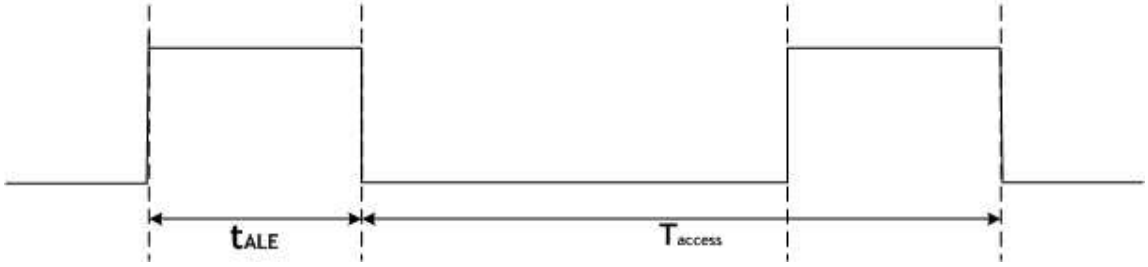
Reset timing



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

External memory access timing

ALE signal for external memory



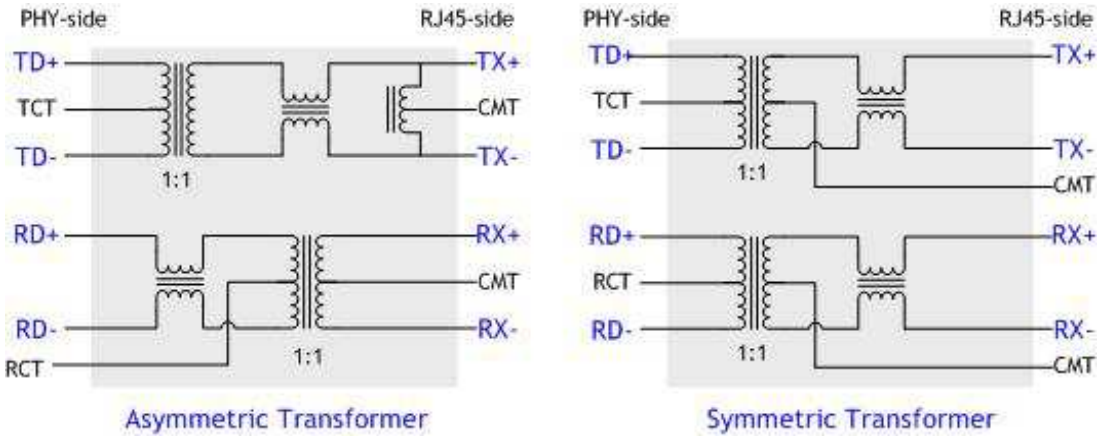
Description		Min	Max
t_{ALE}	ALE signal duration = ALECON register value + 1 clock (clock speed = 88.4736MHz)	1 clock	256 clock
T_{access}	External memory access period = 3us + EXTWTST register value	3 us	744us

10.5 Crystal Characteristics

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

10.6 Transformer Characteristics

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



내부 PHY모드를 사용하는 경우, Auto MDI/MDIX (Crossover)를 위해서 symmetric transformer를 사용해야 한다.

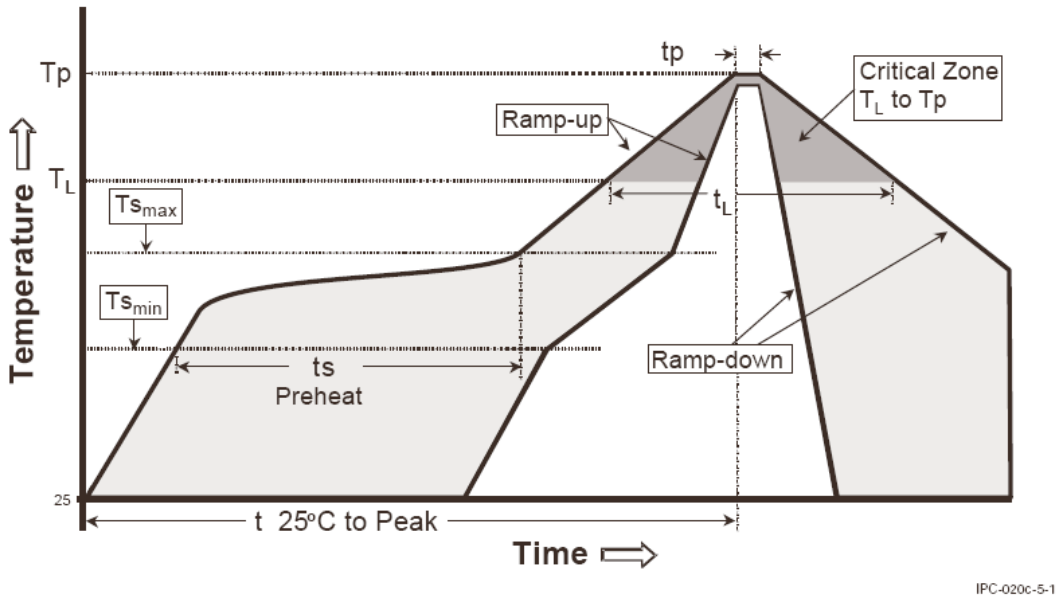
외부 PHY모드를 사용하는 경우, 외부 PHY의 specification과 맞는 transformer를 사용해야 한다.

11 IR Reflow Temperature Profile (Lead-Free)

Moisture Sensitivity Level: 3

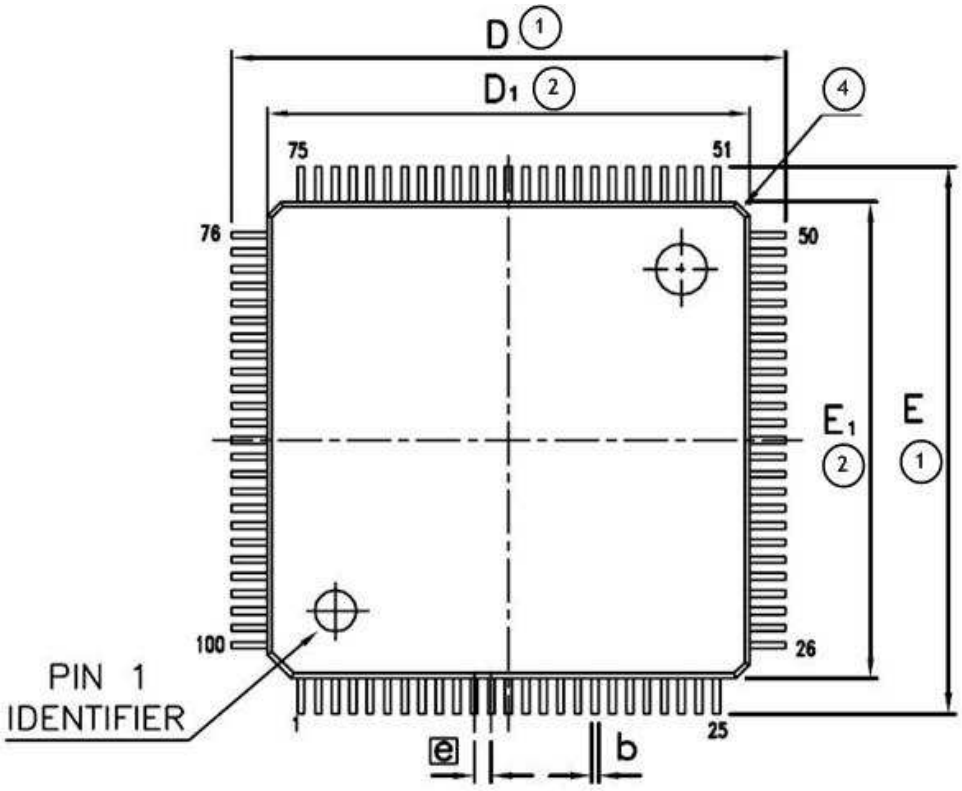
Dry Pack Required: Yes

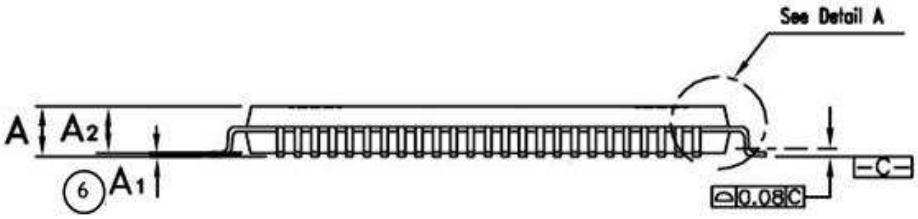
Average RAMp-Up Rate (Ts_{max} to Tp)	3° C/second max.
Preheat - Temperature Min (Ts_{min}) - Temperature Max (Ts_{max}) - Time (ts_{min} to ts_{max})	 150 °C 200 °C 60-180 seconds
Time maintained above: - Temperature (TL) - Time (tL)	 217 °C 60-150 seconds
Peak/Classification Temperature (Tp)	260 +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

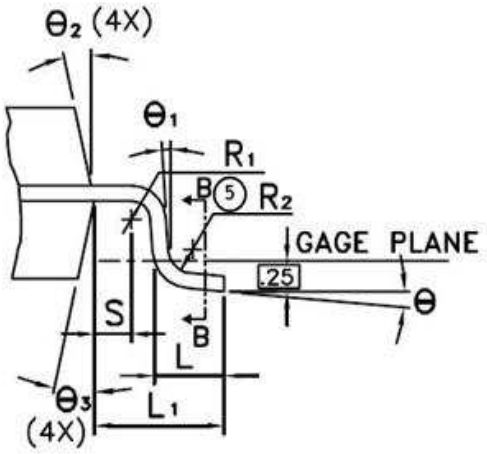


12 Package Descriptions

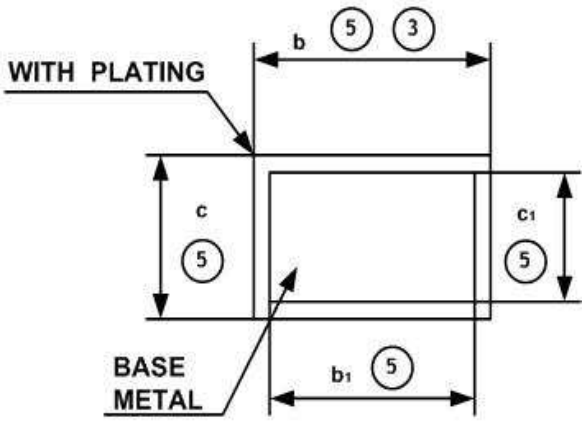
12.1 Package type: LQFP 100







DETAIL A



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

Note:

① To be determined at seating plane - C -.

② Dimensions 'D_1' and 'E_1' do not include mold protrusion. D_1' and 'E_1' are 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.

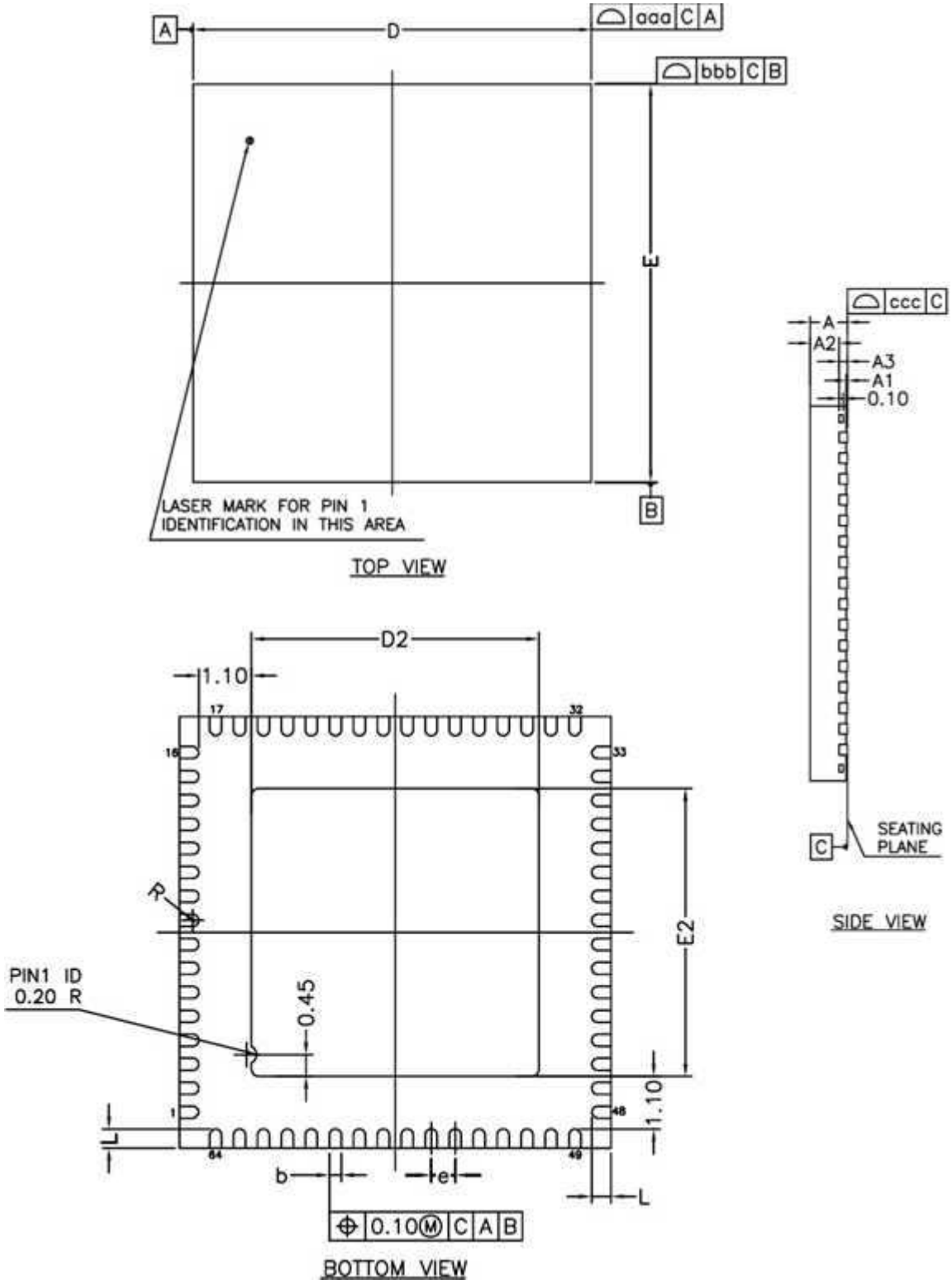
⑥ A_1 is defined as the distance from the seating plane to the lowest point of the package body.

⑦ Controlling dimension : Millimeter

⑧ Reference Document : JEDEC MS-026 , BED.

12.2 Package type: QFN 64

* CONTROLLING DIMENSION: mm



SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	-	-	0.90	-	-	0.035
A1	-	-	0.05	-	-	0.002
A2	-	0.65	0.70	-	0.026	0.028
A3		0.200 REF.			0.008 REF.	
b	0.18	0.25	0.30	0.007	0.010	0.012
D	15.85	16.00	16.15	0.624	0.630	0.636
D		9.00 bsc			0.354 bsc	
D2	5.90	6.00	6.10	0.232	0.236	0.240
E		9.00 bsc			0.354 bsc	
E2	5.90	6.00	6.10	0.232	0.236	0.240
L	0.35	0.40	0.45	0.014	0.016	0.018
e		0.50 bsc			0.020 bsc	
R	0.09	-	-	0.004	-	-
TOLERANCES OF FORM AND POSITION						
aaa		0.10			0.004	
bbb		0.10			0.004	
ccc		0.05			0.002	

Note:

① All dimensions are in millimeters.

② Die thickness allowable is 0.305mm maximum (0.012 inches maximum).

③ Dimensioning & tolerances conform to ASME Y14.5M. -1994.

④ The pin #1 identifier must be placed on the top surface of the package by using indentation mark of other feature of package body.

⑤ Exact shape and size of this feature is optional.

⑥ Package WARPAGE max 0.08mm.

⑦ Applied for exposed pad and terminals, exclude embedding part of exposed pad from measuring.

⑧ Applied to terminals.

13 Appendix: Performance Improvement about W7100A

이 section에서는 일반적인 8051과 W7100A의 연산능력을 비교해 보았다.

13.1 Summary

W7100A과 80C51의 8-bit 덧셈, 뺄셈, 곱셈, 나눗셈 operation cycle을 아래 표에 나타내었다. W7100A과 'wizmemcpy' 함수를 동시에 사용한 경우 일반 80C51보다 약 9배 빠른 성능을 보인다.

	80C51 cycle	**W7100A** cycle / with user code	**W7100A** cycle / with wizmemcpy
ADD	36	12	**4**
SUB	36	12	**4**
MUL	96	12	**6**
DIV	96	20	**10**

다음 section들은 좀 더 자세한 성능비교에 대해 알아볼 것이다.

13.2 8-Bit Arithmetic Functions

13.2.1 Addition

■ Immediate data

다음 code는 직접 데이터 (상수)를 8-bit register에 더하는 것이다.

RX = RX + *#n*

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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'함수는 W7100A의 Boot ROM에 구현되어있다. 'W7100A Driver Guide'참조

■ Direct addressing

다음 code는 8-bit register로 직접 addressing하여 더하는 것이다.

Rx = Rx + (dir)

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 8-bit register에 간접 addressing하여 더하는 것이다.

Rx = Rx + (@Rx)

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 8-bit register끼리 더하는 것이다.

Rx = Rx + Ry

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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	W7100A 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

다음 code는 8-bit register로부터 직접 addressing하여 뺄셈하는 것이다.

Rx = Rx – (dir)

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 8-bit register로부터 간접 addressing하여 뺄셈하는 것이다.

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

Rx = Rx – (@Ry)

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 8-bit register끼리 뺄셈하는 것이다.

Rx = Rx - Ry

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 8-bit register끼리 곱셈하는 것이다.

Rx = Rx * Ry

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 8-bit register끼리 나눗셈하는 것이다.

Rx = Rx / Ry

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 16-bit 덧셈동작이다. 첫 번째 피 연산자는 registers pair RaRb이고, 두 번째 피 연산자는 registers pair RxRy이다.

RaRb = RaRb + RxRy

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 16-bit 뺄셈동작이다. 첫 번째 피 연산자는 registers pair RaRb이고, 두 번째 피 연산자는 registers pair RxRy이다.

RaRb = RaRb - RxRy

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

13.3.3Multiplication

다음 code는 16-bit 곱셈동작이다. 첫 번째 피 연산자는 registers pair RaRb이고, 두 번째 피 연산자는 registers pair RxRy이다.

RaRb = RaRb * RxRy

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 32-bit 덧셈동작이다. 첫 번째 피 연산자는 4개 registers RaRbRcRd이고, 두 번째 피 연산자는 4개 registers RvRxRyRz이다.

RaRbRcRd = RaRbRcRd + RvRxRyRz

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 32-bit 뺄셈동작이다. 첫 번째 피 연산자는 4개 registers RaRbRcRd이고, 두 번째 피 연산자는 4개 registers RvRxRyRz이다.

RaRbRcRd = RaRbRcRd - RvRxRyRz

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

다음 code는 32-bit 곱셈동작이다. 첫 번째 피 연산자는 4개 registers RaRbRcRd이고, 두 번째 피 연산자는 4개 registers RvRxRyRz이다.

RaRbRcRd = RaRbRcRd * RvRxRyRz

Mnemonic		Opcode	Byte	80C51 Cycle	W7100A 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

Document History Information

Version	Date	Descriptions
Ver.0.9.1 **Beta**	Sep. 2009	Release with W7100 launching
Ver.0.9.2	Dec. 2009	1. Deleted **"How to program FLASH memory in W7100"** document 2. Modify **at *2.5.10*, "APP Entry RD/WR Enable" => "APP Entry(0xFFF7 ~ 0xFFFF) RD/WR Enable"(P.40)** 3. Deleted **"tri-state signals in I/O pins** at ***3 I/O Ports*"**
Ver.0.9.3	Feb. 2010	Modify **"XTLP0 and XTLP1 explanations at *1.4.2.5 Interrupt/Clock*"(P.19)**
Ver.0.9.4	Apr. 2010	1. Modify **"Sn_IMR initial value 0x00 => 0xFF"** 2.Added **"INTLEVEL register to TCPIP Core common register at *8.2.1 Common Register*" (P.72)**
Ver.0.9.5	May. 2010	Deleted **"PPPoE protocol cause of errata"**
Ver.0.9.6	Jan. 2011	1. Added **"at *2.5 SFR definition* and PPPoE function at *1.3.2 TCPIPCore*" (P.34, P.14)** 2. Modify **"GPIO in/out voltage for revision for W7100A at *10.1 Absolute Maximum Ratings*" (P.137)**
Ver.1.0.0	Mar. 2011	1. Deleted **"information about external memory accessing"** 2. Added **"64pin number information at *1.4.2 Pin Description*" (P.17)**
Ver.1.1.0	May. 2011	Define **"voltage of OSC (clock source) input"(P.137)**
Ver.1.1.1	May. 2011	1. Modify **"Pin Layout(nWR, nRD) at *1.4.1 Pin Layout*" (P.16)** 2. Added **"at *1.4.2.6 GPIO*" (P.20)** 3. Added **"at *1.4.2.7 External Memory*" (P.21)**
Ver.1.1.2	May. 2011	Corrected **"GPIO pull-up, pull-down resistor value at *4 I/O Port*" (P.46)**

Ver.1.1.3	Nov. 2011	Modify **"VCC3V3 power supply signal descriptions (KR) at *1.4.2.10 Power supply signal*" (P.22)**
Ver.1.1.4	Jan. 2012	1. Added **"product endurance at W7100A Features at *1.2 W7100A Features*" (P.11)** 2. Added **"recommendation note at *2. Memory*" (P.25)** 3.Modify **"pin number of PLOCK in table at *1.4.2.1 configuration*" (P.18)** (table 1.4.2.1, add the QFN64 pin number of PLOCK)
Ver.1.1.5	May. 2012	Modify **"*Table 2.2 WTST Register Values* of WTST at *2.5.2 Program Code Wait States Register*" (P.33)** (The access time of WTST[2:0] values '3' changed to 'Not Used')
Ver. 1.1.6	Jun. 2012	1. Modify **"description of Figure 2.14 at *2.5.2 Program Code Wait States Register*" (P.33)** 2. Modify **"WTST value '3' changed to '4' at *2.5.2 Program Code Wait States Register*" (P.33)**
Ver. 1.2.0	Sep. 2012	1. Modify **"Figure 2.6 and 2.7 at *2.3.1 Standard 8051 Interface*"** (according to the W7100A errata sheet (erratum 3)) **(P.29~P.30)** 2. Added **"recommendation note of erratum 3 at *2.3.1 standard 8051 interface*" (P.29)**
Ver. 1.2.1	Jan. 2013	1. Added **"Figure 4.1 Port0 Pull-down register at *4 I/O port*"(P.46)** 2. Modify **"Table at *10.2 DC Characteristics*" (P.137)** (Add the I_{lkg}(leakage), and I_{OZ}, Modify the V_{OH} and V_{OL})

Copyright Notice

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr
For more information, visit our website at http://www.wiznet.co.kr