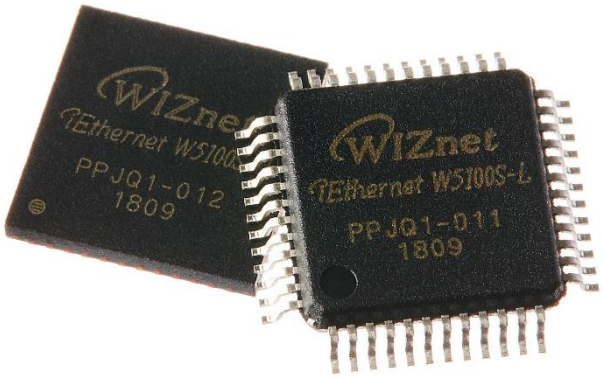
W5100S

(W5100S-L & W5100S-Q)

Version 1.1



注：中文版内对于IP地址寄存器加解锁寄存器描述反了。

https://www.wiznet.io

https://www.iwiznet.cn

W5100S

W5100S 是一款多功能的单芯片网络接口芯片，内部集成全硬件的 TCP/IP 协议栈，以太网 MAC 和 10Base-T/100Base-TX 以太网控制器。主要应用于高集成、高稳定、高性能和低成本的嵌入式系统中。

使用 W5100S，用户 MCU 可以方便的处理 IPv4，TCP，UDP，ICMP，IGMP，ARP，PPPoE 等各种 TCP/IP 协议。W5100S 分别拥有 8KB 的发送缓存和接收缓存，可以最大限度地减少 MCU 的开销。 主机还可以同时使用 W5100S 的 4 个独立的硬件 SOCKETs，并基于每个硬件 SOCKET 开发独立的互联网应用。

W5100S 支持 SPI 接口和并行系统总线接口。 它还提供低功耗/低热量设计，WOL（Wake On LAN），以太网 PHY 掉电模式等。

W5100S 是基于 W5100 改进的低成本网络接口芯片。W5100 使用的任何固件及程序都可以直接在 W5100S 上使用，无需任何修改。 此外，W5100S 采用 48 引脚 LQFP 和 QFN 无铅封装，明显小于 W5100 的 80 引脚封装，方便产品小型化。

特点

- 支持全硬件 TCP/IP 协议:
  TCP, UDP, WOL, ICMP, IGMPv1/v2, IPv4, ARP, PPPoE
- 支持 4 个独立的 Sockets
- 支持 SOCKET-less 指令:
  ARP-请求, PING-请求
- 支持以太网掉电模式和主时钟选通节能模式
- 支持基于 UDP 的网络唤醒（WOL）功能
- 支持 SPI 和并行总线接口
- 高速 SPI 接口（MODE 0/3）
- 系统总线接口（2 位地址线和 8 位数据线）
- 内置共计 16Kbytes 的发送/接收缓存
- 集成 10BaseT/100BaseTX 以太网 PHY
- 支持以太网自动协商（全/半双工, 10Base-T/100Base-TX)
- 支持 Auto-MDIX 功能（只在以太网自动协商模式下支持）
- 不支持 IP 分片功能
- 工作电压：3.3V（I/O 兼容 5V 信号电压）
- 网络指示灯（全/半双工，链接指示, 10Mb/100Mb 指示,传输指示 ）
- 48 管脚 LQFP 封装和 QFN 封装（无铅，7x7mm, 0.5mm 间距)

应用

W5100S 可用于多种嵌入式应用产品，包括：

- 原基于 W5100 开发的各种应用,无需更改固件
- 家用网络设备：机顶盒，PVRs，数字媒体适配器
- 串口转以太网：访问控制，LED 显示器，无线 AP 等
- 并口转以太网：POS/金融打印机，复印机
- USB 转以太网：存储设备，网络打印机
- GPIO 转以太网：家用网络传感器
- 安防系统：DVRs,网络照相机，终端机
- 工业和楼宇自动化
- 医用检测设备
- 嵌入式服务器
- 物联网 IOT 应用及 IOT 云应用

结构图

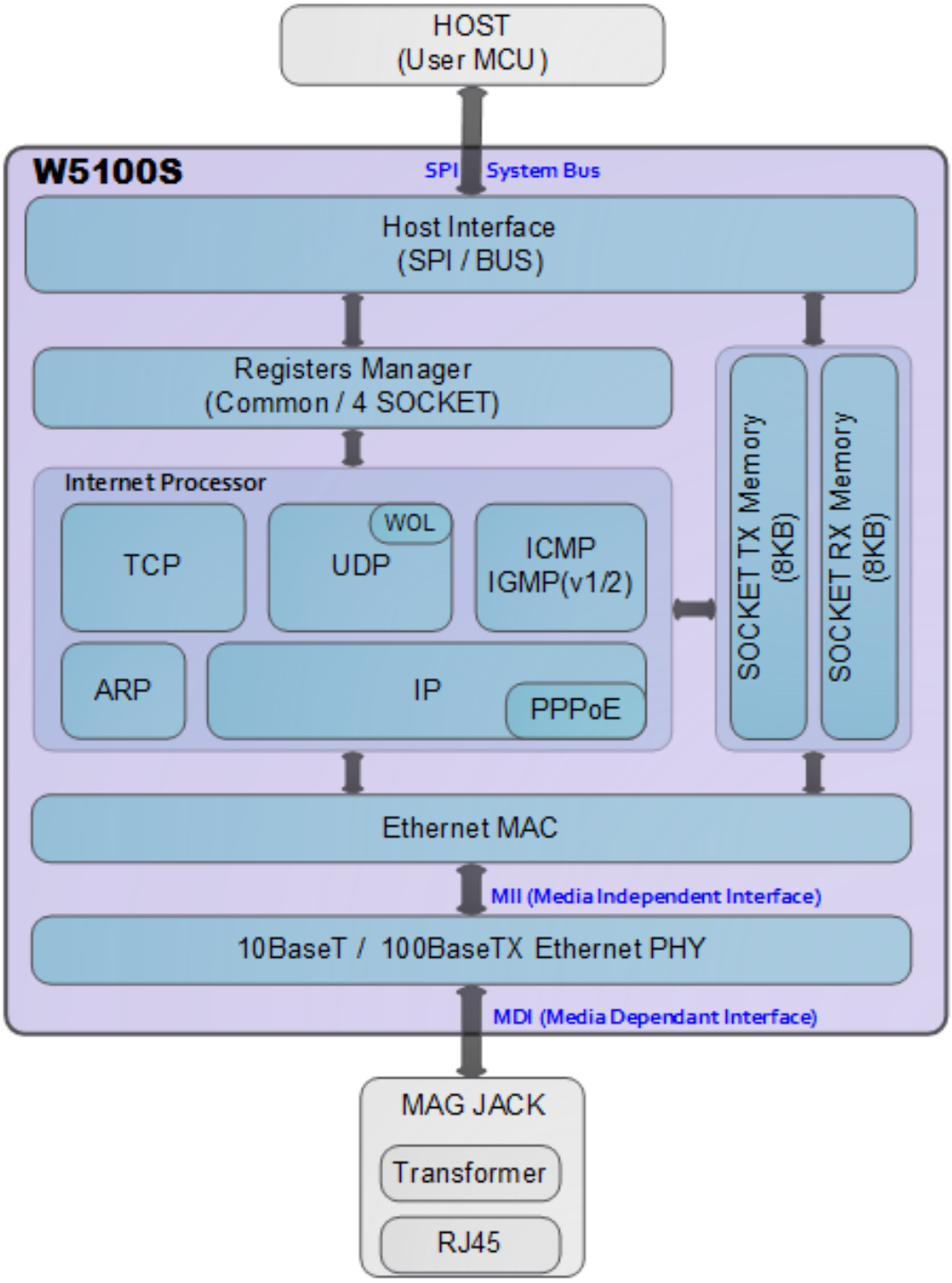

HOST
(User MCU)
W5100S
SPI
System Bus
Host Interface
(SPI / BUS)
Registers Manager
(Common / 4 SOCKET)
Internet Processor
TCP
WOL
UDP
ICMP
IGMP(v1/2)
ARP
IP
PPPoE
SOCKET TX Memory
(8KB)
SOCKET RX Memory
(8KB)
Ethernet MAC
MII (Media Independent Interface)
10BaseT / 100BaseTX Ethernet PHY
MDI (Media Dependant Interface)
MAG JACK
Transformer
RJ45

图 1 结构图

目录

图片列表

表格列表

1 引脚描述

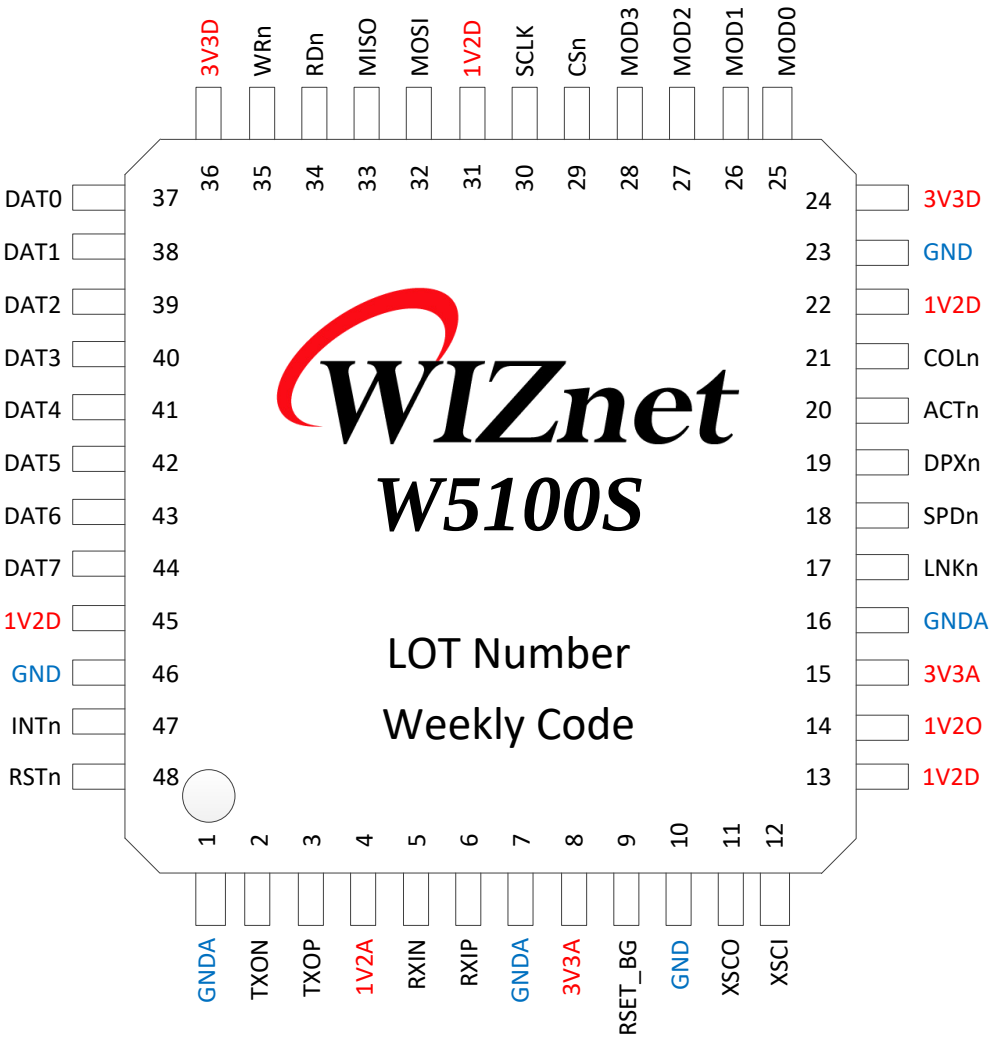


图 2 W5100S 管脚图

表 1 管脚类型

类型	描述
I	输入
O	输出
M	交替（多功能）信号
U	内部上拉 75KΩ 电阻
D	内部下拉式 75KΩ 电阻
A	模拟信号
P	电源脚

1.1 引脚描述

表 2 引脚描述

针脚 #	符号	类型	描述
1	GNDA	AP	**模拟电源地**
2	TXON	AO	**TXOP/TXON 信号组** 在 MDI 模式下通过 TXOP/TXON 信号组发送差分数据信号。
3	TXOP	AO	
4	1V2A	AP	**1.2V 模拟电源** 电压由 1V2O 管脚输出。
5	RXIN	AI	**RXIP/RXIN 信号组** 在 MDI 模式下通过 RXIP/RXIN 信号组接收差分数据信号。
6	RXIP	AI	
7	GNDA	AP	**模拟电源地**
8	3V3A	AP	**3.3V 模拟电源**
9	RSET_BG	AO	**片外偏压电阻** 必须通过外部 12.3KΩ 电阻(误差 1%)连接至模拟地。
10	GND	AP	**数字电源地**
11	XSCO	AO	**25MHz 时钟** 25MHz 有源晶振和无源晶振 W5100S 使用外部 25MHz 时钟源提供 25MHz（低频模式）或 100MHz（正常模式）作为内部时钟
12	XSCI	AI	**如果使用有源晶振，则必须使用 25MHz@1.2V，XSCI 必须连接，且 XSCO 脚必须悬空。** **更多信息，请参阅** *Clock Selection Guide*。
13	1V2D	P	**1.2V 数字电源** 电压由 1V2O 管脚输出。
14	1V2O	PO	**内部变压器 1.2V 输出** W5100S 的内部变压器输出，输出最大值为 150mA 的电流的 1.2V 电源。 确保为外接 3.3uF 电容，提供 1V2D 和 1V2A 的稳定电压输出。 必须使用磁珠隔开 1V2D 和 1V2A。 此电源输出仅适用于 W5100S。 它不能用于其他设备。
15	3V3A	AP	**3.3V 模拟电源**
16	GNDA	AP	**模拟电源地**
17	LNKn	OU	**连接 LED 指示** 它在 SPI 和并行总线模式上都是有效的

			低电平: Link 连接 高电平: Link 断开
18	SPDn	OU	**连接速度 LED 指示** 它在 SPI 和并行总线模式上都是有效的。 低电平: 100Mbps 高电平: 10Mbps
19	DPXn	OU	**全双工 LED 指示** 它在 SPI 和并行总线模式上都是有效的。 低电平: 全双工 高电平: 半双工
20	ACTn	OU	**连接活动 LED 指示** 它在 SPI 和并行总线模式上都是有效的。 未闪烁 ： 已连接但是没有收发数据。 闪烁: 已连接，并根据数据收发情况闪烁。 高电平:链接活动状态指示关闭
21	COLn	OU	**连接冲突检测 LED 指示** 它在 SPI 和并行总线模式上都是有效的。 它表示在数据传输过程中发生冲突。 低电平：发生冲突 高电平：无冲突
22	1V2D	P	**1.2V 数字电源** 电压由 1V2O 管脚输出。
23	GND	P	**数字电源地**
24	3V3D	P	**3.3V 数字电源**
25	MOD[0]	ID	**W5100S 接口模式选择** 接口模式由 MOD 【3~0】选择 “0000” ： SPI 模式 “010X” ： 并行总线模式
26	MOD[1]	ID	
27	MOD[2]	ID	
28	MOD[3]	ID	
29	CSn	IU	**W5100S 片选** 低电平: 被选中 高电平: 未选中
30	SCLK	ID	**SPI 时钟** 在 SPI 模式下，它被用于 SPI 时钟输入。 但在并行总线模式下，此引脚必须接地或者悬空
31	1V2D	P	**1.2V 数字电源** 电压由 1V2O 管脚输出。
32	MOSI /ADDR0	IDM	**SPI 主输出/从输入/ 地址总线 0** MOSI ： 在 SPI 模式下，接收到来自主机的 SPI 数据 ADDR0 :在并行总线模式下，它用于地址 0.

33	MISO /ADDR1	IOPM	SPI 主输入/从输出/ **Address 1** MISO ：在 SPI 模式下, SPI 数据发送到主机。 ADDR1：在并行总线模式下，它用于地址 1.
34	RDn	IU	**读使能** 在并行总线模式下，它表示读操作。 在 SPI 模式下，它必须连接到 3V3D 或悬空
35	WRn	IU	**写使能** 在并行总线模式下，它指示写操作。
36	3V3D	P	**3.3V 数字电源**
37	DAT0	IOU	**8 位数据总线** 在并行总线模式下，DAT [7:0] 用于接收主机或 W5100S 的数据 在 SPI 模式下，DAT[7:0]必须被悬空
38	DAT1	IOU	
39	DAT2	IOU	
40	DAT3	IOU	
41	DAT4	IOU	
42	DAT5	IOU	
43	DAT6	IOU	
44	DAT7	IOU	
45	1V2D	P	**1.2V 数字电源**
46	GND	P	**数字电源地**
47	INTn	OP	**中断** 当 W5100S 以太网通信期间发生中断事件时，INTn 通知主机。 低电平：中断产生 高电平：无中断 请参考 IEN(中断引脚使能)中的 MR2(模式寄存器 2)，INTPTMR(中断等待时间寄存器)、IMR(中断寄存器)、IMR2(中断寄存器 2)、SLIMR (SOCKET-less 中断寄存器)
48	RSTn	IP	**复位** 低电平初始化或重新初始化 W5100S 低电平持续时间不小于 500ns，在复位 RSTn 之后，W5100S 需要 60.3ms 进行初始化。(参考 ***错误!未找到引用源。 错误!未找到引用源。***) 低电平: W5100S 重新初始化。 高电平:正常操作

2 内存映射

W5100S 和 W5100 拥有一样的内存映射表，从而可以完全兼容原有的为 W5100 开发的程序，并附加增强功能或改进功能的新寄存器，下面的图 3 显示了 W5100S 完整存储器映射表。

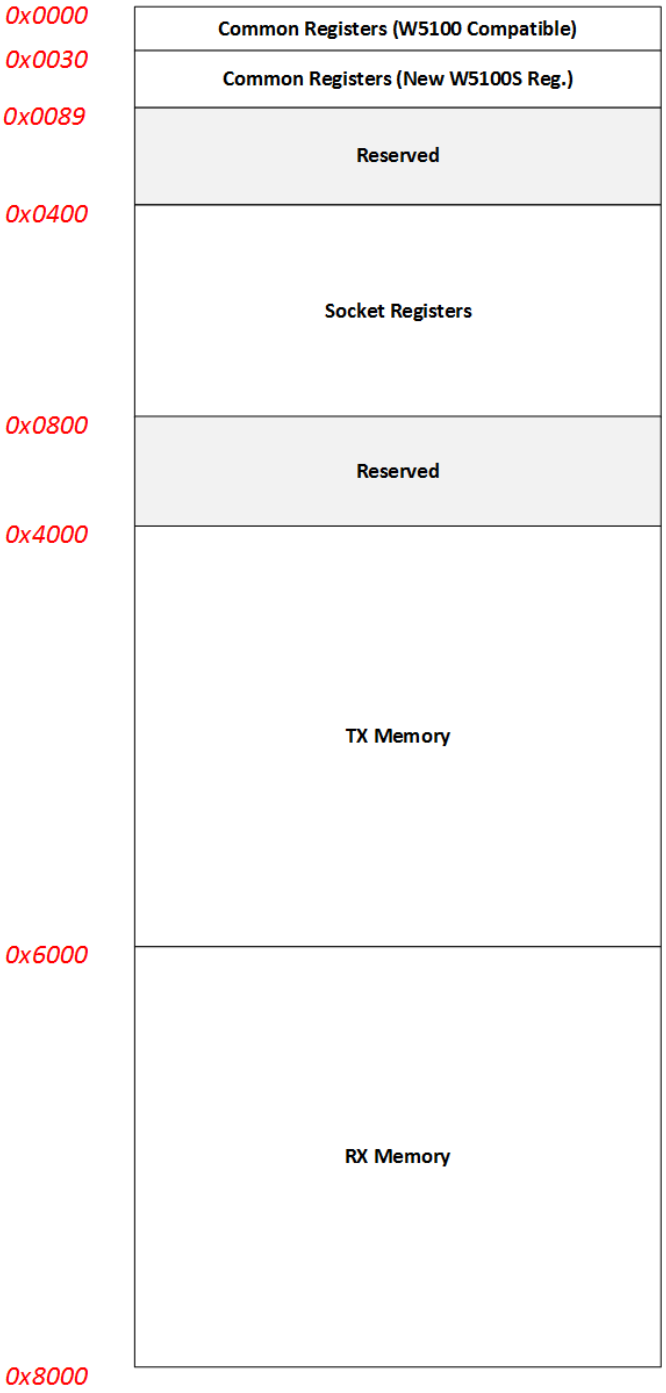


图 3 内存映射表

图 3 显示了通用寄存器模块， SOCKET 的寄存器模块和 TX/RX 缓存模块的偏移地址。

在 W5100S 复位后，每个 SOCKET 的 TX / RX 缓冲区大小会复位为默认值 2KB/2KB，但可以由 TMSR（TX 存储器容量寄存器）和 RMSR（RX 存储器容量寄存器）或 SOCKET 的 TX / RX 缓冲区大小寄存器（Sn_TXBUF_SIZE / Sn_RXBUF_SIZE）设置。 所有 SOCKET 的 TX / RX 的总缓冲区大小不得超过 8 KB。

2.1 W5100S 寄存器

2.1.1 通用寄存器

表 3 通用寄存器

地址	寄存器
0x0000	模式 (MR)
0x0001 0x0002 0x0003 0x0004	网关地址 (GAR0) (GAR1) (GAR2) (GAR3)
0x0005 0x0006 0x0007 0x0008	子网掩码 (SUBR0) (SUBR1) (SUBR2) (SUBR3)
0x0009 0x000A 0x000B 0x000C 0x000D 0x000E	源 MAC 地址 (SHAR0) (SHAR1) (SHAR2) (SHAR3) (SHAR4) (SHAR5)
0x000F 0x0010 0x0011 0x0012	源 IP 地址 (SIPR0) (SIPR1) (SIPR2) (SIPR3)
0x0013 0x0014	中断挂起时间 (INTPTMR0) (INTPTMR1)
0x0015	中断 (IR)
0x0016	中断屏蔽 (IMR)
0x0017 0x0018	发送超时时间 (RTR0) (RTR0)
0x0019	发送重传次数 (RCR)
0x001A	接收存储区大小 (RMSR)
0x001B	发送存储区大小 (TMSR)

地址	寄存器
0x001C ~ 0x001F	保留
0x0020	中断 2 (IR2)
0x0021	中断屏蔽 2 (IMR2)
0x0022 ~ 0x0027	保留
0x0028	PPP LCP 请求定时器 (PTIMER)
0x0029	PPP LCP 魔术数 (PMAGIC)
0x002A 0x002B 0x002C 0x002D	IP 地址无法到达 (UIPR0) (UIPR1) (UIPR2) (UIPR3)
0x002E 0x002F	端口无法到达 (UPORTR0) (UPORTR1)
0x0030	模式 2 (MR2)
0x0031	保留
0x0032 0x0033 0x0034 0x0035 0x0036 0x0037	PPPoE 模式下的目标 MAC 地址 (PHAR0) (PHAR1) (PHAR2) (PHAR3) (PHAR4) (PHAR5)
0x0038 0x0039	PPPoE 模式下会话 ID (PSIDR0) (PSIDR1)
0x003A 0x003B	PPPoE 模式下的最大接收单元 (PMRUR0) (PMRUR1)
0x003C	PHY 状态 (PHYSR0)

地址	寄存器
0x003E	PHY 地址值 (PHYAR)
0x003F	PHY 寄存器地址 (PHYRAR)
0x0040 0x0041	PHY 数据输入 (PHYDIR0) (PHYDIR1)
0x0042 0x0043	PHY 数据输出 (PHYDOR0) (PHYDOR1)
0x0044	PHY 访问控制 (PHYACR)
0x0045	PHY 分频(PHYDIVR)
0x0046 0x0047	PHY 控制 (PHYCR0) (PHYCR1)
0x0048 ~ 0x004B	保留
0x004C	SOCKET-less 控制 (SLCR)
0x004D 0x004E	SOCKET-less 重传超时时间 (SLRTR0) (SLRTR1)
0x004F	SOCKET-less 重传次数 (SLRCR)
0x0050 0x0051 0x0052 0x0053	SOCKET-less 目标 IP 地址 (SLPIPR0) (SLPIPR1) (SLPIPR2) (SLPIPR3)
0x0054 0x0055 0x0056 0x0057 0x0058 0x0059	SOCKET-less 目标 MAC 地址 (SLPHAR0) (SLPHAR1) (SLPHAR2) (SLPHAR3) (SLPHAR4) (SLPHAR5)
0x005A 0x005B	PING 序列号 (PINGSEQR0) (PINGSEQR1)
0x005C 0x005D	PING ID 号 (PINGIDR0) (PINGIDR1)

地址	寄存器
0x005E	SOCKET-less 中断屏蔽 (SLIMR)
0x005F	SOCKET-less 中断 (SLIR)
0x0060 ~ 0x006A	保留
0x0070	时钟锁定 (CLKLCKR)
0x0071	网络锁定 (NETLCKR)
0x0072	PHY 锁定 (PHYLCKR)
0x0073 ~ 0x007F	保留
0x0080	芯片版本 (VERR)
0x0081	保留
0x0082 0x0083	100us Tick 计数器 (TCNTR0) (TCNTR1)
0x0084 ~ 0x0087	保留
0x0088	Ticker 计数器清除 (TCNTCLR)

2.1.2 SOCKET 寄存器

表 4 Socket 寄存器

语法	描述	地址				
		Sn_	S0_	S1_	S2_	S3_
Sn_MR	SOCKET n 模式	0x0400+(0x0100 x n)	0x0400	0x0500	0x0600	0x0700
Sn_CR	SOCKET n 控制	0x0401+(0x0100 x n)	0x0401	0x0501	0x0601	0x0701
Sn_IR	SOCKET n 中断	0x0402+(0x0100 x n)	0x0402	0x0502	0x0602	0x0702
Sn_SR	SOCKET n 状态	0x0403+(0x0100 x n)	0x0403	0x0503	0x0603	0x0703
Sn_PORTR0	SOCKET n 源端口	0x0404+(0x0100 x n)	0x0404	0x0504	0x0604	0x0704
Sn_PORTR1		0x0405+(0x0100 x n)	0x0405	0x0505	0x0605	0x0705
Sn_DHAR0	SOCKET n 目标 MAC 地址	0x0406+(0x0100 x n)	0x0406	0x0506	0x0606	0x0706
Sn_DHAR1		0x0407+(0x0100 x n)	0x0407	0x0507	0x0607	0x0707
Sn_DHAR2		0x0408+(0x0100 x n)	0x0408	0x0508	0x0608	0x0708
Sn_DHAR3		0x0409+(0x0100 x n)	0x0409	0x0509	0x0609	0x0709
Sn_DHAR4		0x040A+(0x0100 x n)	0x040A	0x050A	0x060A	0x070A
Sn_DHAR5		0x040B+(0x0100 x n)	0x040B	0x050B	0x060B	0x070B
Sn_DIPR0	SOCKET n 目标 IP 地址	0x040C+(0x0100 x n)	0x040C	0x050C	0x060C	0x070C
Sn_DIPR1		0x040D+(0x0100 x n)	0x040D	0x050D	0x060D	0x070D
Sn_DIPR2		0x040E+(0x0100 x n)	0x040E	0x050E	0x060E	0x070E
Sn_DIPR3		0x040F+(0x0100 x n)	0x040F	0x050F	0x060F	0x070F
Sn_DPORTR0	SOCKET n 目标端口	0x0410+(0x0100 x n)	0x0410	0x0510	0x0610	0x0710
Sn_DPORTR0		0x0411+(0x0100 x n)	0x0411	0x0511	0x0611	0x0711
Sn_MSS0	SOCKET n 最大分段尺寸	0x0412+(0x0100 x n)	0x0412	0x0512	0x0612	0x0712
Sn_MSS1		0x0413+(0x0100 x n)	0x0413	0x0513	0x0613	0x0713
Sn_PROTOR	SOCKET n IP 协议	0x0414+(0x0100 x n)	0x0414	0x0514	0x0614	0x0714
Sn_TOS	SOCKET n IP 服务类型	0x0415+(0x0100 x n)	0x0415	0x0515	0x0615	0x0715
Sn_TTL	SOCKET n IP 生存时间	0x0416+(0x0100 x n)	0x0416	0x0516	0x0616	0x0716
保留	保留	0x0417+(0x0100 x n)	0x0417	0x0517	0x0617	0x0717
保留	保留	0x041D+(0x0100 x n)	0x041D	0x051D	0x061D	0x071D
Sn_RXBUF_SIZE	SOCKET n 接收缓冲区大小	0x041E+(0x0100 x n)	0x041E	0x051E	0x061E	0x071E
Sn_TXBUF_SIZE	SOCKET n 发送缓冲区大小	0x041F+(0x0100 x n)	0x041F	0x051F	0x061F	0x071F
Sn_TX_FSR0	SOCKET n 空闲发送缓存	0x0420+(0x0100 x n)	0x0420	0x0520	0x0620	0x0720
Sn_TX_FSR1		0x0421+(0x0100 x n)	0x0421	0x0521	0x0621	0x0721

Sn_TX_RD0	SOCKET n 发送读指针	0x0422+(0x0100 x n)	0x0422	0x0522	0x0622	0x0722
Sn_TX_RD1		0x0423+(0x0100 x n)	0x0423	0x0523	0x0623	0x0723
Sn_TX_WR0	SOCKET n 发送写指针	0x0424+(0x0100 x n)	0x0424	0x0524	0x0624	0x0724
Sn_TX_WR1		0x0425+(0x0100 x n)	0x0425	0x0525	0x0625	0x0725
Sn_RX_RSR0	SOCKET n 接收大小	0x0426+(0x0100 x n)	0x0426	0x0526	0x0626	0x0726
Sn_RX_RSR1		0x0427+(0x0100 x n)	0x0427	0x0527	0x0627	0x0727
Sn_RX_RD0	SOCKET n 接收读指针	0x0428+(0x0100 x n)	0x0428	0x0528	0x0628	0x0728
Sn_RX_RD1		0x0429+(0x0100 x n)	0x0429	0x0529	0x0629	0x0729
Sn_RX_WR0	SOCKET n 接收写指针	0x042A+(0x0100 x n)	0x042A	0x052A	0x062A	0x072A
Sn_RX_WR1		0x042B+(0x0100 x n)	0x042B	0x052B	0x062B	0x072B
Sn_IMR	SOCKET n 中断屏蔽	0x042C+(0x0100 x n)	0x042C	0x052C	0x062C	0x072C
Sn_FRAGR0	SOCKET n 在 IP 包头中的片段偏移	0x042D+(0x0100 x n)	0x042D	0x052D	0x062D	0x072D
Sn_FRAGR1		0x042E+(0x0100 x n)	0x042E	0x052E	0x062E	0x072E
Sn_MR2	SOCKET n 模式 2	0x042F+(0x0100 x n)	0x042F	0x052F	0x062F	0x072F
Sn_KPALVTR	SOCKET n Keep-alive 计时器	0x0430+(0x0100 x n)	0x0430	0x0530	0x0630	0x0730
Sn_RTR0	SOCKET n 重传超时时间	0x0432+(0x0100 x n)	0x0432	0x0532	0x0632	0x0732
Sn_RTR1		0x0433+(0x0100 x n)	0x0433	0x0533	0x0633	0x0733
Sn_RCR	SOCKET n 重传次数	0x0434+(0x0100 x n)	0x0434	0x0534	0x0634	0x0734

3 寄存器描述

寄存器描述示例

* 寄存器符号(寄存器的全名)

- [寄存器类型][寄存器地址偏移量][默认值]

寄存器描述…。

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

Sn_IR [3: 0] 表示寄存器 [高位：低位]。

Sn_IR [3: 0] = '0001'表示Sn_IR [3] ='0'，Sn_IR [2] ='0'，Sn_IR [1] ='0'，Sn_IR [0] ='1'。

[寄存器/位类型]：寄存器的类型和位。

- [RW]：可以读取和写入。

- [R = W]：读取和写入的值相同。

- [RO]：只读

- [WO]：只写

- [W]：只写

- [WC]：通过书写'1'清除。

- [W0]：必须仅写入'0'。

- [W1]：必须只写入'1'。

- [AC]：自动清除/自动清除

- [1]：始终读取'1'

- [0]：始终读取'0'

- [-] ： 无法使用

[Address Offset]：寄存器地址偏移

[Reset Value]：默认值。

例如)

3.1.1 MR (模式寄存器)

[RW][0x0000][0x03]

MR 是 Mode Register 的缩写。 该寄存器可以被读取和写入。 寄存器地址偏移量为'0x0000'，在复位后它被设置为'0x03'。

7	6	5	4	3	2	1	0
RST	-	-	PB	PPPoE	-	AI	IND
AC	W0	W0	R=W	R=W	-	1	1

Ex2) MR [RST]

MR [RST]表示 MR 中的 RST 位。

Ex3) MR [7:0]

MR [7：0]表示 MR 中第 7 位至第 0 位。

3.1 通用寄存器

3.1.1 MR (模式寄存器)

[RW][0x0000] [0x03]

MR 寄存器用于设置 S/W 复位，ping block 模式和 PPPoE 模式。

7	6	5	4	3	2	1	0
RST	-	-	PB	PPPoE	-	-	-

位	符号	描述
7	RST	**软件复位** 如被设置为“1”，芯片内部寄存器将被初始化。复位后自动清0.
[6:5]	-	保留
4	PB	**Ping Block**模式 0: 关闭Ping block 1: 启用Ping block 如果该位(bit)设置为'1'，接收 ping 请求时就没有响应。
3	PPPoE	**PPPoE 模式** 0: 关闭PPPoE 模式 1: 启用PPPoE 模式 如果你想使用 ADSL，改为需要设为 ‘1’ 。
[2:0]	-	保留

3.1.2 GWR (网关 IP 地址寄存器)

[R=W] [0x0001-0x0004] [0x00]

GWR 寄存器用于设置网关地址。

*该寄存器只有当 NETLCKR (网络参数锁定寄存器)为“未锁定”状态时才可更改。

例如） GWR = “192.168.0.1”

GWR0(0x0001)	GWR1(0x0002)	GWR2(0x0003)	GWR3(0x0004)
192 (0xC0)	168 (0xA8)	0 (0x00)	1 (0x01)

3.1.3 SUBR (子网掩码寄存器)

[R=W] [0x0005–0x0008] [0x00]

SUBR 寄存器用于设置子网掩码地址。

*该寄存器只有当 NETLCKR (网络参数锁定寄存器)为“未锁定”状态时才可更改。

例如） SUBR = “255.255.255.255”

SUBR0(0x0005)	SUBR0(0x0006)	SUBR0(0x0007)	SUBR0(0x0008)
255 (0xFF)	255 (0xFF)	255 (0xFF)	255 (0xFF)

3.1.4 SHAR (源 MAC 地址寄存器)

[R=W] [0x0009-0x000E] [0x00]

SHAR 寄存器用于设置源 MAC 地址。

*该寄存器只有当 NETLCKR (网络参数锁定寄存器)为“未锁定”状态时才可更改。

例如） SHAR = “11:22:33:AA:BB:CC”

SHAR0(0x0009)	SHAR1(0x000A)	SHAR2(0x000B)
0x11	0x22	0x33
SHAR3(0x000C)	SHAR4(0x000D)	SHAR5(0x000E)
0xAA	0xBB	0xCC

3.1.5 SIPR (源 IP 地址寄存器)

[R=W] [0x000F-0x0012] [0x00]

SIPR 寄存器用于设置源 IP 地址。

*该寄存器只有当 NETLCKR (网络参数锁定寄存器)为“未锁定”状态时才可更改。

例如） SIPR = “192.168.0.100”

SIPR0(x000F)	SIPR1(0x0010)	SIPR2(0x0011)	SIPR3(0x0012)
192 (0xC0)	168 (0xA8)	0 (0x00)	100(0x64)

3.1.6 INTPTMR (中断挂起时间寄存器)

[RW][0x0013-0x0014][0x0000]

INTPTMR 用于设置内部中断挂起计时器计数。 当 INTn 取消置为高电平时，定时器计数初始化为 INTPTMR，并且每 SYS_CLK x 4 从初始值减 1。当发生中断并且相应的中断屏蔽被设置且 INTPTMR 为 0 时，INTn 被置为低。

例如） INTPTMR = 1000(0x03EB)

INTPTMR0(0x0013)	INTPTMR1(0x0014)
0x03	0xEB

3.1.7 IR (中断寄存器)

[RW] [0x0015] [0x00]

当 W5100S 或 SOCKET n 有中断事件发生时，IR 中的相应位被设置为'1'。 如果事件发生并且 IMR 中相应的中断屏蔽位被设置为 1 并且内部中断挂起定时器计数器为 0，则 INTn 被置为低。当事件被清除或相应的屏蔽位被设置为'0'时，INTn 被置为高电平。

7	6	5	4	3	2	1	0
CONFLICT	UNREACH	PPPTERM	-	S3_INT	S2_INT	S1_INT	S0_INT
WC	WC	WC	-	AC	AC	AC	AC

位	符号	描述
7	CONFLICT	**IP 冲突** 在收到 APR 请求时，如果发现发送方 IP 与本地 IP 重复，该位将被置‘1’。
6	UNREACH	**目标端口不可抵达** 当接收到 ICMP（目的端口不可达）包后，该位将被置 ‘1’。 当该位为 ‘1’ 时，通过相应的 UIPR 和 UPORTR 寄存器，可以查询到目标信息。如：IP 地址和端口号。
5	PPPTERM	**PPPoE 连接关闭** 当 PPP0E 模式下接收到 PPPT 和 LCPT 包，该位将被置 ‘1’。
4	-	保留
[3:0]	Sn_INT	**SOCKET n 中断** 当 Socket “n” 发生中断，第 “n” 位会被置 ‘1’。 当 Sn_IR 寄存器是 ‘0x00’ Sn_INT 位会自动置零。

3.1.8 IMR (中断屏蔽寄存器)

[R=W] [0x0016] [0x00]

中断屏蔽寄存器（IMR）用于屏蔽中断源。

7	6	5	4	3	2	1	0
CNFT	UNREACH	PPPTERM	-	S3_INT	S2_INT	S1_INT	S0_INT
R=W	R=W	R=W		R=W	R=W	R=W	R=W

位	符号	描述
7	CNFT	**IP冲突中断屏蔽** 0：关闭IP冲突中断 1：启用IP冲突中断
6	UNREACH	**屏蔽目的地址不能抵达中断** 0：关闭目的地址不能抵达 1：开启目的地址不能抵达
5	PPPTERM	**屏蔽PPPoE模式下的**PADT/LCPT**包中断** 0：关闭PADT/LCPT中断 1：开启 PADT/LCPT 中断
4	-	**保留**
3	S3_INT	**屏蔽 SOCKET 3 中断** 0：屏蔽SOCKET 3中断 1：开启 SOCKET 3 中断
2	S2_INT	**屏蔽 SOCKET 2 中断** 0：屏蔽SOCKET 2中断 1：开启 SOCKET 2 中断
1	S1_INT	**屏蔽 SOCKET 1 中断** 0：屏蔽SOCKET 1中断 1：开启 SOCKET 1 中断
0	S0_INT	**屏蔽 SOCKET 0 中断** 0：屏蔽SOCKET 0中断 1：开启 SOCKET 0 中断

3.1.9 RTR (重传超时时间值寄存器)

[R=W] [0x0017-0x0018] [0x07D0]

RTR 寄存器用于设置重传超时的时间值。每一单位数值为 100us。初始化时值设为 2000（0x07D0），即相当于 200 毫秒（100us X 2000）。

在 RTR 配置的时间内，W5100S 等待 Sn-CR(CONNECT, DISCON, CLOSE, SEND, SEND_MAC, SEND_KEEP command)发送后，来自对方的回应。如果在 RTR 时间段内没有回应，W5100S 进行包重传或触发超时中断。(参考 *4.7 重传*)

例如） RTR = 5000 (0x1388)

5000*100us = 0.5s

RTR0(0x0017)	RTR1(0x0018)
0x13	0x88

3.1.10 RCR (重传次数寄存器)

[R=W] [0x0019] [0x08]

RCR 寄存器用于设置重传传送的次数。当第 ‘RCR+1’ 次重传时，超时中断就会置 ‘1’ 。 (中断寄存器（Sn_IR）的 ‘中断’ 位(‘TIMEOUT’ 位)被置'1'）。

*RTR 和 RCR 寄存器设置 ARP 和 TCP 发送超时时间和重传次数。 (参考 *4.7 重传)*

3.1.11 RMSR (接收缓存大小寄存器)

[R=W] [0x001A] [0x55]

RMSR 寄存器用于设置每个 SOCKET n 的接收缓存大小。 所有 SOCKET n 的接收缓存总共不能超过 8 Kbytes。 (参考 *Sn_RXBUF_SIZE (SOCKET n 接收缓存大小寄存器)*)

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

每个 SOCKET 的接收缓存大小都是通过 RMSR 中 SOCKET 所对应的 S0 位和 S1 位设置的。

缓存大小	S1	S0
1 KB	0	0
2 KB	0	1
4 KB	1	0
8 KB	1	1

3.1.12 TMSR (发送缓存大小寄存器)

[R=W] [0x001B] [0x55]

TMSR 寄存器用于设置每个 SOCKET n 的发送缓存大小。 所有 SOCKET n 的发送缓存总共不能超过 8 Kbytes。(参考 *Sn_TXBUF_SIZE (SOCKET n 发送缓存大小寄存器)*)

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

每个 SOCKET 的发送缓存大小都是通过 TMSR 中 socket 对应 S0 位和 S1 位设置的。

缓存大小	S1	S0
1 KB	0	0
2 KB	0	1
4 KB	1	0
8 KB	1	1

3.1.13 IR2 (中断寄存器 2)

[RW] [0x0020] [0x00]

发生 WOL 事件时，IR2 [WOL]被设置为'1'。 如果事件发生且 IMR2 [WOL]设置为 1 并且内部中断挂起定时器计数器为 0，则 INTn 置为低电平。 当事件被清除或 IMR2 [WOL] 被设置为'0'时，INTn 被置为高电平。

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	WOL
-	-	-	-	-	-	-	WC

位	符号	描述
[7:1]	-	保留
0	WOL	**WOL 魔术包中断** 1： 接收到基于 UDP 的 WOL Magic 数据包。 0 : -

3.1.14 IMR2 (中断屏蔽寄存器 2)

[R=W] [0x0021] [0x00]

IMR2 寄存器用于打开或屏蔽相应的 IR2 中断位。

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	WOL
-	-	-	-	-	-	-	R=W

位	符号	描述
[7:1]	-	保留
0	WOL	**屏蔽 WOL 魔术包中断** 1： 打开 WOL 魔术包中断 0： 屏蔽 WOL 魔术包中断

3.1.15 PTIMER (PPP LCP 请求定时器寄存器)

[R=W] [0x0028] [0x28]

PRIMER 寄存器用于设置发送 LCP Echo（响应请求）所需要的时间间隔。 单位是 25ms。

例如） PTIMER = 200 (0xC8),

200 * 25ms = 5s

3.1.16 PMAGIC (PPP LCP 魔术数寄存器)

[R=W] [0x0029] [0x00]

PMAGIC 寄存器用于设置 LCP Echo（响应请求）4 个字节的魔术数。

例如） PMAGIC = 0x01

PMAGIC(0x0029)
0x01

LCP Magic Number = 0x01010101

3.1.17 UIPR (IP 地址无法到达寄存器)

[RO] [0x002A-0x002D] [0x0000]

当W5100S收到无法到达的数据包（IR [UNR] ='1'）时，数据包中的对方IP地址被写入UIPR寄存器。

例如） UIPR = “192.169.0.21”

UIPR0(0x002A)	UIPR1(0x002B)	UIPR2(0x002C)	UIPR3(0x002D)
192(0xC0)	168(0xA8)	0(0x00)	21(0x15)

3.1.18 UPORTR (端口号无法到达寄存器)

[RO] [0x002E-0x002F] [0x0000]

当W5100S收到无法到达的数据包（IR [UNR] ='1'）时，数据包中的对方端口号被写入UPORTR寄存器。

例如） UPORTR = 3000 (0x0BB8)

UPORTR0(0x002E)	UPORTR1(0x002F)
0x0B	0xB8

3.1.19 MR2 (模式寄存器 2)

[R=W] [0x0030] [0x40]

MR2 配置系统操作时钟（SYS_CLK）选择，中断管脚激活，TCP 和 UDP 端口扫描攻击预防，WOL（局域网唤醒）和强制 ARP。

7	6	5	4	3	2	1	0
CLKSEL	IEN	NOTCPRST	UDPURB	WOL	-	FARP	-
R=W	R=W	R=W	R=W	R=W	W0	R=W	W0

位	符号	描述
7	CLKSEL	**系统时钟（SYS_CLK）选择** **只有 CLKLCKR（时钟锁定寄存器）解锁时，该位才可被设置。** (参考 *3.1.40_ CLKLCKR (时钟锁定寄存器)* [WO] ***[0x0070] [0x00]***) 1 : 25MHz 0 : 取决于 PHYCR1[PWDN] 通过 PHCR1 配置选择 SYS_CLK PHYCR1[PWDN] | SYS_CLK 0 | 100 MHz 1 | 25 MHz
6	IEN	**INTn 启用/禁用** 1：INTn 启用 0：INTn 禁止（INTn 始终为高）
5	NOTCPRST	**TCP RST 数据包** 如果对端将 TCP 数据包发送到 W5100S 不存在的端口上。W5100S 会自动发送 RST 包。 但它可能是端口扫描攻击的目标。 但是，如果该位设置为'1'，则 W5100S 不会对具有错误端口的 TCP 数据包发送 RST 数据包。 1：阻止发送 RST 包 0：正常
4	UDPURB	**UDP 端口不可达的数据包** UDP 模式下，如果对端将 UDP 数据包发送到 W5100S 不存在的端口上。W5100S 自动向对端发送 ICMP 数据包（目标端口不可达），但它可能是端口扫描攻击的目标。 但是如果此位设置为'1'，W5100S 不会发送 ICMP 数据包（目标端口不可达）。

		1：阻止发送 ICMP 数据包（目标端口不可到达消息） 0：正常
3	WOL	**Wake On LAN** 是否接收 WOL 数据包。 1：接收 WOL 数据包 0：不接收 WOL 数据包
2	-	保留
1	FARP	**强制 ARP 模式** UDP 模式下，当 SOCKET 将数据连续发送到同一个对端时，SOCKET 会在第一个 UDP 数据包之前发送一次 ARP 请求。 但是，如果此位设置为'1'，它会为每个 UDP 数据包发送 ARP 请求。 1：打开强制 ARP 模式 0：关闭强制 ARP 模式
0	-	保留

3.1.20 PHAR (PPPoE 模式下目标 MAC 地址寄存器)

[R=W] [0x0032-0x0037] [0x0000]

PHAR 需要在 PPPoE 连接过程中写入 PPPoE 服务器的 MAC 地址。

例如） PHAR = “11:22:33:AA:BB:CC”

PHAR0(0x0032)	PHAR1(0x0033)	PHAR2(0x0034)
0x11	0x22	0x33
PHAR3(0x0035)	PHAR4(0x0036)	PHAR5(0x0037)
0xAA	0xBB	0xCC

3.1.21 PSIDR (PPPoE 模式下会话 ID 寄存器)

[R=W] [0x0038-0x0039] [0x0000]

PSID 需要在 PPPoE 连接过程中写入 PPPoE 服务器的会话 ID。

例如） PSIDR = 0x1234

PSIDR0(0x0038)	PSIDR1(0x0039)
0x12	0x34

3.1.22 PMRUR (PPPoE 模式下最大接收单元)

[R=W] [0x003A-0x003B] [0xFFFF]

PMRU 规定了 PPPoE 模式下的最大接收单元。PMRUR 寄存器的值不能大于 1472.而且 PMRUR 的值只能在 SOCKET 处于“OPEN” (Sn_CR [OPEN] = ‘ 1’)状态前进行设置。

例如） PMUR = 1000 (0x03E8)

PMUR0(0x0038)	PMUR1(0x0039)
0x03	0xE8

3.1.23 PHYSR (PHY 状态寄存器)

[RO] [0x003C] [0x00]

PHYSR 指示 PHY 当前的操作模式和 LINK 状态，PHY 的操作模式和 LINK 状态是由 PHYCR0 寄存器配置的(PHY Control Register 0)。

7	6	5	4	3	2	1	0
CABOFF	-	AUTO	SPD	DPX	FDPX	FSPD	LINK
RO		RO	RO	RO	RO	RO	RO

位	符号	描述
7	CABOFF	**网线插入状态** 1：网线未插入 0：网线已插入
6	-	保留
5	AUTO	**以太网自动协商（Auto Negotiation），通过 PHYCR0 [2] 配置** **1：禁用自动协商** **0：启用自动协商**
4	SPD	**以太网速度状态，通过（PHYCR0[1]）配置** 1 : 10Mbps 0 : 100Mbps
3	DPX	**双工状态，通过（PHYCR0[0]）配置** 1：半双工 0：全双工
2	FDPX	**双工模式标志位** 1：半双工 0：全双工
1	FSPD	**速度标志位** 1 : 10Mbps 0 : 100Mbps
0	LNK	**PHY 连接标志位** 1：上线 0：下线

3.1.24 PHYRAR (PHY 寄存器地址寄存器)

[R=W] [0x003F] [0x00]

PHYRAR 寄存器配置内部以太网 PHY 的 PHY 寄存器地址。

7	6	5	4	3	2	1	0
-	-	-	A4	A3	A2	A1	A0
			R=W	R=W	R=W	R=W	R=W

位	符号	描述
7	-	保留
6	-	保留
5	-	保留
[4:0]	A[4:0]	**PHY 寄存器地址**

3.1.25 PHYDIR (PHY 数据输入寄存器)

[R=W] [0x0040-0x0041] [0x0000]

PHTDIR 向 PHYAR 指定的 PHY 寄存器内写入数据。

例如） PHYDIR = 0x1234

PHYDIR0(0x0040)	PHYDIR1(0x0041)
0x34	0x12

3.1.26 PHYDOR (PHY 数据输出寄存器)

[RO] [0x0042-0x0043] [0x0000]

PHYDOR 读取 PHYAR 指定的 PHY 寄存器内的数据

例如） PHYDOR = 0x1234

PHYDOR0(0x0042)	PHYDPR1(0x0043)
0x34	0x12

3.1.27 PHYACR (PHY 访问控制寄存器)

[AC] [0x0044] [0x00]

PHYACR 配置由 PHYAR 指定的 PHY 寄存器的访问类型。

通信方式	值
Write	0x01
Read	0x02

3.1.28 PHYDIVR (PHY 分频寄存器)

[R=W] [0x0045] [0x01]

内部以太网 PHY 使用系统时钟（SYS_CLK）的分频时钟。 这个分频时钟不能超过 2.5MHz

Value	Divider	SYS_CLK=100MHz	SYS_CLK=25MH
0x00	1/32	3.125MHz (N/A)	781.25KHz
0x01	1/64	1.5625MHz	390.625KHz
others	1/128	781.25KHz	195.3125KHz

3.1.29 PHYCR0 (PHY 控制寄存器 0)

[WO] [0x0046] [0x00]

PHYCR 寄存器配置以太网 PHY 的工作模式例如自动协商，速度选择和全双工选择

在设置 PHYCR 之前，PHYLCKR（PHY 锁定寄存器）必须处于解锁模式

7	6	5	4	3	2	1	0
-	-	-	-	-	AUTO	SPD	DPX
					WO	WO	WO

位	符号	描述
[7:3]	-	保留
2	AUTO	**自动协商** 如果 AUTO 位设置为'1'，则 SPD 和 DPX 将被忽略 1：禁用自动协商 0：启用自动协商
1	SPD	**10/100 速度** 当 AUTO 为'0'时，它的速率可以配置为 10Mbps 或 100Mbps 1 : 10 Mbps 0 : 100 Mbps
0	DPX	**Full/Half Duplex 全/半双工** 当 AUTO 为 0 时，它可以配置为半双工或全双工模式 1 : HDX 0 : FDX

3.1.30 PHYCR1 (PHY 控制寄存器 1)

[R=W] [0x0047] [0x41]

PHYCR 寄存器配置以太网 PHY 操作模式，如 PHY 掉电模式，PHY 复位。

在设置 PHYCR 之前，PHYLCKR（PHY 锁定寄存器）必须处于解锁模式。

7	6	5	4	3	2	1	0
-	-	PWDN	-	-	-	-	Reset
W0	W0	R=W	W0	W0	W0	W0	AC

位	符号	描述
[7:6]	-	保留
5	PWDN	**PHY 掉电模式** 1：使能掉电模式和 SYS_CLK 切换为 25MHz 0：禁止掉电模式且 SYS_CLK 由 MR2 [CLKSEL]决定 (参考 *7.4.1 复位时序*)
[4:1]	-	保留
0	RST	**PHY 复位** 当 PHY 复位位设置为 0 时，SYS_CLK 切换到 25MHz。 PHY 复位完成后，该位自动清零并且 SYS_CLK 回到前一个时钟。 (参考 *7.4.1 复位时序*) 1：正常 0：PHY H / W 复位

MR2[CLKSEL]	SYS_CLK
0	100 MHz
1	25 MHz

3.1.31 SLCR (SOCKET-less 控制寄存器)

[RW] [0x004C] [0x00]

SLCR 寄存器配置 ARP 和 PING 请求发送命令。 每个命令不能同时执行，并且在 SLCR 寄存器清除之前不能重新配置。 每个命令的结果通过 SLIR (SOCKET-less 中断 Register)查看。

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ARP	PING
-	-	-	-	-	-	AC	AC

位	符号	描述
[7:2]	-	保留
1	ARP	**ARP 请求发送命令** 1：发送 ARP 请求数据包 0：准备就绪
0	PING	**PING 请求发送命令** 1：发送 PING 请求 0：准备就绪

3.1.32 SLRTR (SOCKET-less 重传超时时间寄存器)

[R=W] [0x004D-0x004E] [0x07D0]

SLRTR 寄存器设置 SLCR 重传超时时间。 单位是 100us。 如果对 SLCR 发送的 ARP 或 PING 请求数据包没有响应，则 W5100S 每 SLRTR 自动重传请求数据包。(参考 *4.7 重传*)

例如） SLRTR = 5000 (0x1388),

5000 * 100us = 0.5s

SLRTR0(0x004D)	SLRTR1(0x004E)
0x013	0x88

3.1.33 SLRCR (SOCKET-less 重传次数寄存器)

[R=W] [0x004F] [0x00]

SLRCR 寄存器设置 SLCR 重传次数。 如果重传次数超过 SLRCR，则发生 SOCKET-less 超时（SLIR [TIMEOUT] ='1'）。 (参考 *4.7 重传*)

3.1.34 SLPIPR (SOCKET-less 目标 IP 地址寄存器)

[R=W] [0x0050-0x0053] [0x00000000]

SLPIPR 寄存器为 SLCR 发送的 ARP 或 Ping 请求数据包配置目标 IP 地址。

例如） SLPIPR = “192.169.0.21”

SLPIPR0(0x0050)	SLPIPR1(0x0051)	SLPIPR2(0x0052)	SLPIPR3(0x0053)
192(0xC0)	168(0xA8)	0(0x00)	21(0x15)

3.1.35 SLPHAR (SOCKET-less 目标 MAC 地址寄存器)

[RO] [0x0054-0x0059] [0x000000000000]

如果 W5100S 收到针对 SLCR [ARP]的 ARP 应答并且 SLIPR [ARP]被设置为'1'。ARP 应答中的对方 MAC 地址被写入 SLPHAR 寄存器。

例如） SLPHAR = “11:22:33:AA:BB:CC”

SLPHAR0(0x0054)	SLPHAR1(0x0055)	SLPHAR2(0x0056)
0x11	0x22	0x33
SLPHAR3(0x0057)	SLPHAR4(0x0058)	SLPHAR5(0x0059)
0xAA	0xBB	0xCC

3.1.36 PINGSEQR (PING 序列号寄存器)

[R=W] [0x005A-0x005B] [0x0000]

PINGSEQR 寄存器为 PING 请求数据包配置序列号，并且不会自动增加。

例如） PINGSEQR = 1000 (0x03E8)

PINGSEQR0(0x005A)	PINGSEQR1(0x005B)
0x03	0xE8

3.1.37 PINGIDR (PING ID 寄存器)

[R=W] [0x005C-0x005D] [0x0000]

PINGIDR 寄存器配置 PING 请求数据包 ID。

例如） PINGIDR = 256 (0x0100)

PINGIDR0(0x005C)	PINGIDR1(0x005D)
0x01	0x00

3.1.38 SLIMR (SOCKET-less 中断屏蔽寄存器)

[R=W] [0x005E] [0x00]

SLIME 用于屏蔽相应的 SLUR（SOCKET-less 中断寄存器）中断位。

7	6	5	4	3	2	1	0
-	-	-	-	-	TIMEOUT	ARP	PING
-	-	-	-	-	R=W	R=W	R=W

位	符号	描述
[7:3]	-	保留
2	TIMEOUT	**屏蔽 TIMEOUT 中断** 1：启用 TIMEOUT 中断 0：禁用 TIMEOUT 中断
1	ARP	**屏蔽 ARP 中断** 1：启用 ARP 中断 0：禁用 ARP 中断
0	PING	**屏蔽 PING 中断** 1：启用 PING 中断 0：禁止 PING 中断

3.1.39 SLIR (SOCKET-less 中断寄存器)

[RW] [0x005F] [0x00]

发生 SOCKET-less 事件时，SLIR 中相应的位被设置为'1'。 如果事件发生且 SLIMR 中对应的中断屏蔽位设置为 1 且内部中断挂起计时器计数器为 0，则 INTn 置为低电平。 当事件被清除或相应的屏蔽位被设置为'0'时，INTn 被置为高电平。

7	6	5	4	3	2	1	0
-	-	-	-	-	TIMEOUT	ARP	PING
-	-	-	-	-	WC	WC	WC

位	符号	描述
[7:3]	-	保留
2	TIMEOUT	**TIMEOUT 中断** 当 TIMEOUT 发生时，该位被设置为'1'。
1	ARP	**ARP 中断** 当收到 ARP 应答时，该位被设置为'1'。
0	PING	**PING 中断** 当收到 PING 应答时，该位被设置为'1'。

3.1.40 CLKLCKR (时钟锁定寄存器)

[WO] [0x0070] [0x00]

CLKLCKR 状态必须解锁才能设置 MR2 [CLKSEL]。 在改变 CLKLCKR 状态之前，CLKLCKR 保持当前状态。

解锁	锁定
0xCE	其它

3.1.41 NETLCKR (网络锁定寄存器)

[WO] [0x0071] [0x00]

NETLCKR 状态必须解锁才能设置 GWR，SUBR，SHAR 和 SIPR。

在更改 NETLCKR 状态之前，LETLCKR 保持当前状态。

解锁	锁定
0xC5	0x3A

3.1.42 PHYLCKR (PHY 锁定寄存器)

[WO] [0x0072] [0x00]

PHYLCKR 状态必须解锁才能设置 PHYCR0 和 PHYCR1. 在改变 PHYLCKR 状态之前，PHYLCKR 保持当前状态。

解锁	锁定
0x53	其它

3.1.43 VERR (芯片版本寄存器)

[RO] [0x0080] [0x51]

VERR 寄存器显示 W5100S 芯片版本。

3.1.44 TCNTR (Ticker 计数器寄存器)

[RO][0x0082-0x0083][0x0000]

TCNTR 是 W5100S 内部计数器，它自 SYS_CLK 操作开始自动增加。

单位是 100us。

3.1.45 TCNTCLR (Ticker 计数器清除寄存器)

[WO][0x0088][0x00]

通过 TCNTCLR 写访问，TCNTR 计数器值被初始化。

3.2 SOCKET 寄存器

3.2.1 Sn_MR (SOCKET n 模式寄存器)

[R=W] [0x0000+0x0100*(n+4)] [0x00]

该寄存器用于配置所有 SOCKET 的选项或协议类型。

Sn_MR 寄存器的值只能在 SOCKET 处于“OPEN” (Sn_CR [OPEN] = ‘ 1’)状态前设置。

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

位	符号	描述
7	MULTI	**UDP组播模式** 0：关闭组播 1：开启组播 该位只有当UDP模式(P[3:0] = ‘0010’)，才能生效。(参考 *4.3.3 UDP组播*) 使用组播模式，需要Sn_DIPR和Sn_DPORT在Socket n通过Sn_CR的打开配置命令打开之前，单独配置组播IP地址及端口号。
6	MF	**MAC地址过滤** 0：关闭MAC地址过滤 1：启用MAC地址过滤 该位只有在MACRAW(P[3:0] = ‘0100’)模式期间，才能生效。 如果设置为 ‘1’,W5100S 只接收广播包以及发送给他本身的包。当该位设置为 ‘0’ ，W5100S可以接收到来自网络的所有包。如果用户希望实现混合TCP、IP协议栈，建议设置该位为 ‘1’ ，以降低主机处理所有接收包的负载。
5	ND / MC	**使用无延时ACK** 0：关闭无延时ACK选项 1：开启无延时 ACK 选项 在 TCP 模式下，如果设置为 ‘1’ ，TCP 模式下当从对方接收到数据包，该 SOCKET 不等待超时时间(通过 RTR 设置)，直接发送 ACK 包。参考) Sn_CR [RECV]命令运行后，如果 TCP 模式 SOCKET 窗口大小小于 MSS 值，则立即发送 ACK 数据包（忽视 ND 位）。 当该位为 ‘0’ ，W5100S 发送 ACK 包需要 RTR 设定的超时时间做延时。 组播 IGMP 版本选择(MC)

		在 UDP 模式(Sn_MR[3:0] = ‘UDP’ & Sn_MR[MULTI] = ‘1’)下，配置 IGMP 的消息版本。 0：使用IGMP 版本2 1：使用 IGMP 版本 1
4	-	保留
[3:0]	P[3:0]	**协议选择** 该位配置Socket n使用的协议模式，MACRAW模式只能在SOCKET 0上使用。

P3	P2	P1	P0	协议
0	0	0	0	SOCKET 关闭
0	0	0	1	TCP
0	0	1	0	UDP
0	0	1	1	IPRAW
0	1	0	0	MACRAW

3.2.2 Sn_CR (SOCKET n 控制寄存器)

[RW][AC] [0x0001+0x0100*(n+4)] [0x00]

Sn_CR 寄存器用于设置 Socket n 的配置命令如 OPEN、CLOSE、CONNECT、LISTEN、END 和 RECEIVE。W5100S 执行 SOCKET 命令后，相应的 Sn_CR 位会自动清零。
在先前的 SOCKET 命令位被清除之后，必须配置下一个 SOCKET 命令。

值	符号	描述
0x01	OPEN	**SOCKET OPEN 指令** 在执行OPEN指令之前，必须通过Sn_MR寄存器设置SOCKET的相关参数。执行OPEN指令之后，Sn_SR寄存器指示SOCKET状态。

Sn_MR (P[3:0])	Sn_SR
Sn_MR_CLOSE (‘0000’)	SOCK_CLOSED (0x00)
Sn_MR_TCP (‘0001’)	SOCK_INIT (0x13)
Sn_MR_UDP (‘0010’)	SOCK_UDP (0x22)
Sn_MR_IPRAW (‘0011’)	SOCK_IPRAW (0x32)
S0_MR_MACRAW (‘0100’)	SOCK_MACRAW (0x42)

0x02	LISTEN	**TCP LISTEN 指令** 执行 LISTEN 指令后，该 SOCKET 作为 TCP 模式下的服务器，会等待 TCP 的客户端发出的链接请求（SYN 包）。 (参考 *4.2.1 TCP Server*)
0x04	CONNECT	**TCP CONNECT 指令** 执行 CONNECT 指令后，该 SOCKET 作为 TCP 模式下的客户端，会主动向 TCP 的服务器发出连接请求（SYN 包）。 (参考 *4.2.2 TCP Client*)
0x08	DISCON	**TCP DISCON 指令** 执行 DISCON 指令后，处于 SOCK_ESTABLESHED (Sn_SR = '0x17') 或 SOCK_CLOSE_WAIT (Sn_SR = '0x1C')状态下的该 SOCKET 会向 TCP 链接的对端发送断开连接的请求包（FIN 包）。
0x10	CLOSE	**SOCKET CLOSE 指令** 执行 CLOSE 指令后，该 SOCKET 关闭(Sn_SR = '0x00')。 警告：Sn_SR 直接更改为 SOCK_CLOSE 状态，在 TCP 模式下，此 SOCKET 直接关闭而不会发送 FIN 数据包。
0x20	SEND	**SOCKET SEND 指令** SEND 命令后，SOCKET 发送的数据量与 Sn_TX_WR（SOCKET n TX 写指针寄存器）和 Sn_TX_RD（SOCKET n RX 读指针寄存器）之间的差值相同。发送的数据不得超过 Sn_RX_FSR（SOCKET n TX Free Buffer Size Register）。在 Sn_IR [SENDOK]设置为'1'后，主机必须执行下一个 SEND 命令。 在 TCP 和 UDP 模式下，如果计算的值超过 MSS（最大段值），则数据按 MSS 尺寸分包并分别发送。 另一方面，在 IPRAW 和 MACRAW 模式下，如果数据超过 MSS，则主机必须将其分为小于 MSS 值的包。 在 TCP 模式下，如果对端收到数据，Sn_RX_FSR 自动增加已发送数据大小，如果对端没有收到数据 Sn_IR [TIMEOUT]置 '1'，Sn_SR 寄存器变为 SOCK_CLOSED 状态。 在 UDP,IPRAW 和 MACRAW 模式下，Sn_IR [SENDOK]置 '1' 后 Sn_TX_FSR 增加已发送数据的大小。
0x21	SEND_MAC	**SOCKET SEND_MAC 指令** SEND_MAC 命令仅用于 UDP 和 IPRAW 模式。 唯一与 SEND 命令不同的是它会跳过 ARP 过程。 在使用 SEND_MAC 命令之前，主机必须用对方 MAC 地址设置 Sn_DHAR（SOCKET n 目标 MAC 地址寄存器）。
0x22	SEND_KEEP	**SOCKET SEND_KEEP 指令** SEND_KEEP 命令仅用于 TCP 模式。 在使用 SEND_KEEP 命令之前，TCP 模式 SOCKET 必须发送多于 1 个字节的数据到对端。 在 SEND_KEEP 命令后，TCP 模式 SOCKET 继续发送 Keep-alive（KA）数

		据包并检查 TCP 连接。 如果对端没有响应，则 KA 分组被重传。 重传超时后，Sn_IR [TIMEOUT]设置为'1'，Sn_SR 更改为 SOCK_CLOSED。 (参考 *4.2.5 Keep Alive*)
0x40	RECV	**SOCKET RECV 指令** 从 SOCKET n 接收缓冲区读取接收到的数据后，主机用 RECV 命令更新 Sn_RX_RD（SOCKET n 读指针寄存器）。

3.2.3 Sn_IR (SOCKET n 中断寄存器)

[RW] [0x0002+0x0100*(n+4)] [0x00]

Sn_IR 寄存器用于提供给 Socket n 中断类型信息，如建立(Establishment)、终止(Termination)、接收数据(Receiving data)和超时(Timeout)。当触发一个中断即 Sn_IMR 的对应位是'1'的时候，Sn_IR 的对应位也将会变成 ‘1’ 。

7	6	5	4	3	2	1	0
-	-	-	SENDOK	TIMEOUT	RECV	DISCON	CON
			WC	WC	WC	WC	WC

位	符号	描述
[7:5]	-	保留
4	SENDOK	**Sn_IR(SENDOK)**中断 当 SEND 命令完成时，此位被置 ‘1’ 。
3	TIMEOUT	**Sn_IR(TIMEOUT)** 中断 当 ARPTO 或者 TCPTO 超时被触发时，此位被置 ‘1’ 。
2	RECV	**Sn_IR(RECV)** 中断 无论何时，只要接收到了对方数据，此位被置 ‘1’ 。
1	DISCON	**Sn_IR(DISCON)** 中断 当接收到对方的 FIN or FIN/ACK 包时，此位被置 ‘1’ 。
0	CON	**n_IR(CON)** 中断 当成功与对方建立连接，且 Sn_SR 变为 SOCK_ESTABLISHED 状态时，此位被置 ‘1’ 。

3.2.4 Sn_SR (SOCKET n 状态寄存器)

[RO] [0x0003+0x0100*(n+4)] [0x00]

Sn_SR 指示了 Socket n 的状态。通过数据的发送/接收或 Sn_CR 寄存器设置 Sn_SR 寄存器 FIN 包而改变。

值	符号	描述
0x00	SOCK_CLOSED	该位指示了Socket n处于关闭状态，资源被释放。 当 DICON, CLOSE 命令生效或当触发超时中断时， W5100S 对应的 Socket n 会无视之前的状态，变为 SOCK_CLOSED。
0x13	SOCK_INIT	该位指示了Socket n 端口打开并处于TCP工作模式。 当 Sn_MR (P[3:0]) = ‘0001’ 且 OPEN 命令生效时， Sn_SR 变为 SOCK_INIT。之后，用户才可以使用 LISTEN 或 C ONNECT 命令。
0x14	SOCK_LISTEN	此位指示着Socket n工作在TCP服务器模式下，且等待对方 （TCP客户端）的连接请求（SYN Packet）。 当连接请求被成功接收以后，Socket_SR会变为SOCK_ESTALB LISHED状态。否则将会在触发TCPTO超时中断之后，变为 SOCK_CLOSED状态。
0x17	SOCK_ESTABLISHE D	指示了Socket n的连接状态。 SOCK_LISTEN状态下，当TCP服务器处理TCP客户端的SYN请 求包或当CONNECT命令配置成功时，变为SOCK_ESTABLISH ED。 在此状态下，可以使用 SEND 或者 RECV 命令进行数据包传 输。
0x1C	SOCK_CLOSE_WAI T	指示了Socket n接收到了来自连接对方发来的断开连接请求 （FIN packet）。这是一个半关闭状态，可以进行数据传输。 若要全部关闭，需要使用DISCON命令。而如果是要关闭Socke t，需要使用CLOSE命令。
0x22	SOCK_UDP	指示了Socket n处于UDP模式下 (Sn_MR(P[3:0]) = ‘0010’)。 当Sn_MR(P[3:0]) = ‘0010’ 且OPEN命令生效时，Sn_SR改 变为SOCK_UDP。 不同于TCP模式，在这个模式下，数据包可以在无连接过程的 情况下传输。
0x32	SOCK_IPRAW	指示了 Socket 0 工作在 IPRAW 模式下。
0x42	SOCK_MACRAW	指示了Socket 0工作在MACRAW模式下。 MACRAW 模式仅在 Socket 0 下生效。当 S0_MR(P[3:0] = ‘0100’ 且 OPEN 命令生效时，Sn_SR 改变为 SOCK_MACRAW。如 UDP 模式一样，Socket 0 工作在 MACRAW模式下时，也能在无连接过程的情况下，实现MAC 数据包（以太网帧）传输。

下表显示了 Socket n 状态改变时的临时状态。

值	符号	描述
0x15	SOCK_SYNSENT	指示Socket n已经发送连接请求（SYN Packet）到对方。
0x16	SOCK_SYNRECV	指示Socket n成功的从对方收到了连接请求包(SYN packet)。
0x18	SOCK_FIN_WAIT	指示SOCKET n正在关闭。 这显示的是断开连接（主动关闭或被动关闭）的过程。
0X1B	SOCK_TIME_WAIT	
0X1D	SOCK_LAST_ACK	

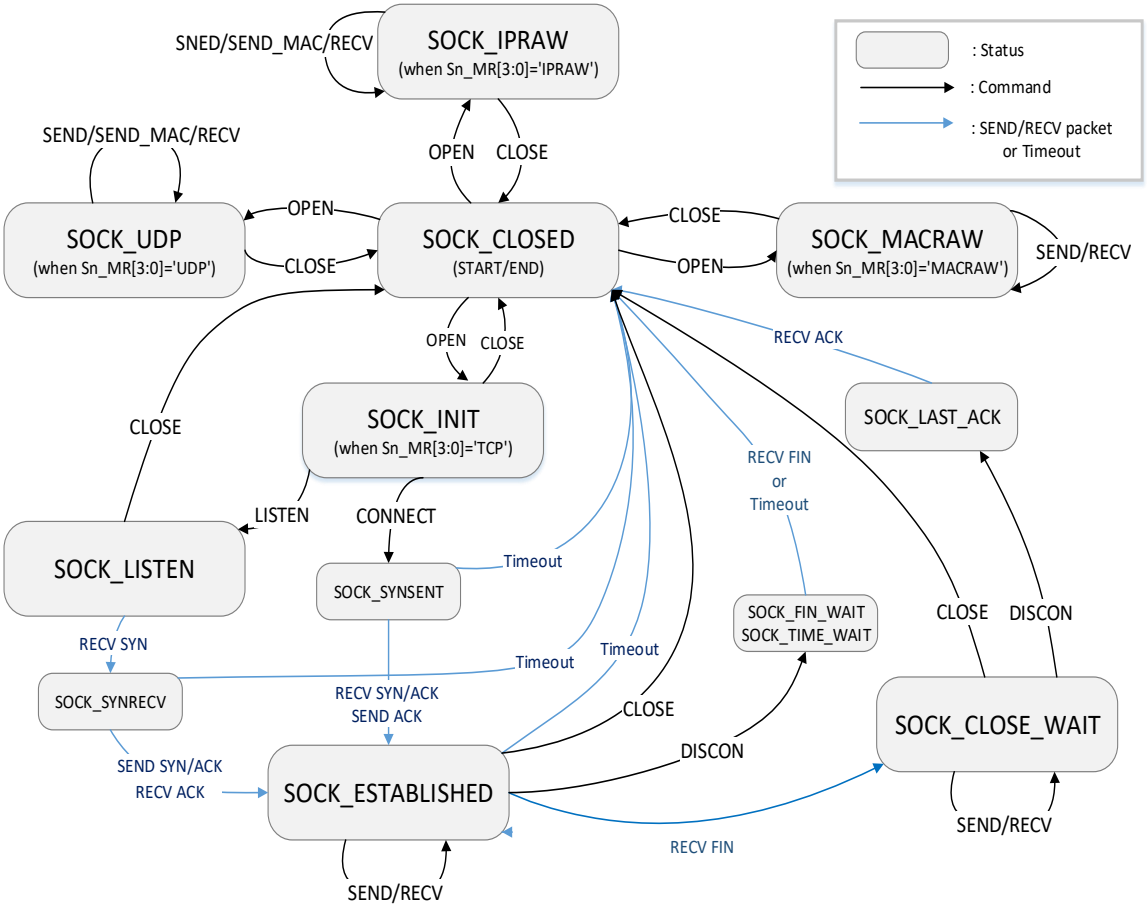


图 4 SOCKET 状态图

3.2.5 Sn_PORTR (SOCKET n 源端口寄存器)

[R=W] [0x0004+0x0100*(n+4), 0x0005+0x0100*(n+4)] [0x0000]

该寄存器配置了Socket n的源端口号。当Socket n工作在TCP或UDP模式下，该寄存器生效。

注意：必须在OPEN命令生效前，完成对该寄存器的设置。

例如） S0_PORTR = 5000 (0x1388)

S0_PORTR0(0x0404)	S0_PORTR1(0x0405)
0x013	0x88

3.2.6 Sn_DHAR (SOCKET n 目标 MAC 地址寄存器)

[RW] [0x0006+0x0100*(n+4), 0x0007+0x0100*(n+4), 0x0008+0x0100*(n+4), 0x0009+0x0100*(n+4), 0x000A+0x0100*(n+4), 0x000B+0x0100*(n+4)] [0x000000000000]

Sn_DHAR寄存器存储的是目标主机MAC地址。

在TCP模式下，Sn_DHAR在CONNECT命令后有效。

在UDP和IPRAW模式下，Sn_DHAR在SEND命令后有效。

如果使用SEND_MAC命令，则必须在发送SEND_MAC指令之前将对方MAC地址设置为Sn_DHAR。

在UDP组播模式下，必须使用组播组MAC地址来配置Sn_DHAR。

(参考 *4.3.3 UDP 组播*)

例如） S0_DHAR = “11:22:33:AA:BB:CC”

S0_DHAR0(0x0406)	S0_DHAR1(0x0407)	S0_DHAR2(0x0408)
0x11	0x22	0x33
S0_DHAR3(0x0409)	S0_DHAR4(0x040A)	S0_DHAR5(0x040B)
0xAA	0xBB	0xCC

3.2.7 Sn_DIPR (SOCKET n 目标 IP 地址寄存器)

[RW] [0x000C+0x0100*(n+4), 0x000D+0x0100*(n+4), 0x000E+0x0100*(n+4), 0x000F+0x0100*(n+4)] [0x00000000]

Sn_DIPR 配置或指示的为 Socket n 的目标主机 IP 地址。

在 TCP 客户端模式下，在 CONNECT 配置命令前，该寄存器设置了 TCP 服务器的 IP 地址。

在 TCP 服务器模式下，他显示了在成功建立连接后，TCP 客户端的 IP 地址。

在 UDP 模式或 IPRAW 模式下，它配置了对方主机的 IP 地址以供 SEND 或 SEND_MAC 配置命令发送 UDP 包。

在 UDP 组播模式下，Sn_DIPR 需要配置组播组的 IP 地址(参考 *4.3.3 UDP 组播*)

注意：在 UDP, UDP 组播 或 IPRAW 模式下，对端的 IP 地址与数据一起存储在 SOCKET n 接收缓存中。

例如） S0_DIPR = “192.168.0.11”

S0_DIPR0(0x040C)	S0_DIPR1(0x040D)	S0_DIPR2(0x040E)	S0_DIPR3(0x040F)
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

3.2.8 Sn_DPORTR (SOCKET n 目标端口寄存器)

[R=W] [0x0010+0x0100*(n+4), 0x0011+0x0100*(n+4)] [0x0000]

Sn_DPORT 配置或指示了Socket n的目标主机端口号，在TCP/UDP模式下生效。

在TCP客户端模式下，在CONNET配置命令前，该寄存器配置了TCP Server监听的端口号。

在 TCP 服务器模式下，他显示了在成功建立连接后，TCP 客户端的端口号。

在 UDP 或 IPRAW 模式下，发送数据前必须用目标端口号设置 Sn_DPORTR 寄存器。

在 UDP 组播模式下，发送数据前必须先通过设置 Sn_DPORTR 以设置组播发送的端口号 (参考 *4.3.3 UDP 组播*)

注意：在 UDP, UDP 组播 或 IPRAW 模式下，对端的端口号与数据一起存储在 SOCKET n 接收缓存中。

例如） S0_DPORTR = 5000 (0x1388),

S0_DPORTR0(0x0410)	S0_DPORTR1(0x0411)
0x13	0x88

3.2.9 Sn_MSS (SOCKET n 最大分段寄存器)

[R=W] [0x0012+0x0100*(n+4), 0x0013+0x0100*(n+4] [0xFFFF]

Sn_MSS寄存器配置Socket n的最大传输单元MTU(最大imum Transfer 单位)。

必须在Sn_CR[OPEN]命令之前配置Sn_MSS寄存器。

在 TCP/UDP 模式下，该寄存器默认设定为最大传输单元值。

Mode	Normal (MR [PPPoE]=’0’)	PPPoE (MR [PPPoE]=’1’)
TCP	1~1460	1~1452
UDP	1~1472	1~1464
IPRAW	1480	1472
MACRAW	1514	

例如） S0_MSS = 1460 (0x05B4),

S0_MSS0(0x0412)	S0_MSS1(0x0413)
0x05	0xB4

3.2.10 Sn_PROTOR (SOCKET n IP 协议寄存器)

[R=W] [0x0014+0x0100*(n+4)] [0x0000]

Sn_PROTOR 可以配置除了 TCP（0x06），UDP（0x11）和 IGMP（0x01）以外的 IP 包头的协议号(参考 *IANA Protocol Number*)。

在 IPRAW 模式下，只能发送和接收在 Sn_PROTR 中配置的协议。

例如） ICMP (Internet Control Message Protocol) = 0x01

3.2.11 Sn_TOS (SOCKET n IP 服务类型寄存器)

[R=W] [0x0015+0x0100*(n+4)] [0x00]

Sn_TOS 寄存器设置在 IP 头的 TSO 字段（服务器类型）。(参考 *IANA_IP Parameters*)

3.2.12 Sn_TTL (SOCKET n IP 生存时间寄存器)

[R=W] [0x0016+0x0100*(n+4)] [0x80]

Sn_TTL 寄存器用于设置 IP 数据包头中的生存期（TTL 生存时间）字段的值。(参考 *IANA_IP Parameters*)

3.2.13 Sn_RXBUF_SIZE (SOCKET n 接收缓存大小寄存器)

[RW] [0x001E+0x0100*(n+4)] [0x02]

Sn_RXBUF_SIZE 配置了 Socket n 的接收缓存大小。Socket n 接收缓存大小寄存器可以配置为 0,1，2，4，和 8 Kbytes。接收缓存大小寄存器从 SOCKET 0 到 SOCKET 3 顺序分配。Sn_RXBUF_SIZE 的总和不得超过 8 KB。Sn_RXBUF_SIZE 也可以由 RMSR 寄存器进行配置。

Value (Dec)	0	1	2	4	8
Buffer size	0KB	1KB	2KB	4KB	8KB

例如） S0_RXBUF_SIZE = 8KB

S0_RXBUF_SIZE(0x041E)
0x08

3.2.14 Sn_TXBUF_SIZE (SOCKET n 发送缓存大小寄存器)

[RW] [0x001F+0x0100*(n+4)] [0x02]

Sn_TXBUF_SIZE 配置了 Socket n 的发送缓存大小。Socket n 发送缓存大小寄存器可以配置为 0,1,2,4 和 8 Kbytes。 发送缓存大小寄存器从 SOCKET 0 到 SOCKET 3 顺序分配。Sn_TXBUF_SIZE 的总和不得超过 8 KB。 Sn_TXBUF_SIZE 也可以由 TMSR 寄存器进行配置。

Value (Dec)	0	1	2	4	8
Buffer size	0KB	1KB	2KB	4KB	8KB

例如） S0_TXBUF_SIZE= 4KB

S0_TXBUF_SIZE(0x041F)
0x04

3.2.15 Sn_TX_FSR (SOCKET n 空闲发送缓存寄存器)

[RO] [0x0020+0x0100*(n+4), 0x0021+0x0100*(n+4)] [0x0800]

Sn_TX_FSR 寄存器指示 SOCKET n 发送缓存空闲大小。

在UDP, IPRAW 或 MACRAW 模式下, Sn_TX_FSR = | Sn_TX_WR(1) – Sn_TX_RD(2) | + 1 在TCP模式下, Sn_TX_FSR = | Sn_TX_WR – Internal Pointer(3) | + 1

(1) SOCKET n TX写指针寄存器
(2) SOCKET n TX读指针寄存器
(3) 由W5100S管理的TCP ACK指针

要存储在 SOCKET n 发送缓冲区中的数据大小不得大于 Sn_TX_FSR。

例如) S0_TX_FSR = 1024 (0x0400)

S0_TX_FSR0(0x0420)	S0_TX_FSR1(0x0421)
0x04	0x00

3.2.16 Sn_TX_RD (SOCKET n 发送读指针寄存器)

[RO] [0x0022+0x0100*(n+4), 0x0023+0x0100*(n+4)] [0x0000]

Sn_TX_RD 表示发送后的 SOCKET n 发送缓冲区中的最后一个数据地址。 Sn_TX_RD 由 Sn_CR [OPEN]初始化。 在 TCP 模式下，Sn_TX_RD 在 TCP 连接过程中重新配置。 通过 Sn_CR [SEND]和 Sn_CR [SEND_MAC]，Sn_TX_WR（SOCKET n TX 写指针寄存器）增加到数据大小。发送数据后，Sn_TX_RD 增加 Sn_TX_WR 并触发 Sn_IR [SENDOK]中断。

例如) S0_TX_RD = 0xd4b3

S0_TX_RD0(0x0422)	S0_TX_RD1(0x0423)
0xd4	0xb3

3.2.17 Sn_TX_WR (SOCKET n 发送写指针寄存器)

[RW] [0x0024+0x0100*(n+4), 0x0025+0x0100*(n+4)] [0x0000]

Sn_TX_WR 表示 SOCKET n 发送缓冲区中最后存储的数据地址。 Sn_TX_RD 由 Sn_CR [OPEN]初始化。 在 TCP 模式下，Sn_TX_RD 在 TCP 连接过程中重新配置。 在 Sn_CR [SEND]和 Sn_CR [SEND_MAC]之前，必须将 Sn_TX_WR 设置为发送数据大小。

例如) S0_TX_WR = 0x0800

S0_TX_WR0(0x0424)	S0_TX_WR1(0x0425)
0x08	0x00

3.2.18 Sn_RX_RSR (SOCKET n 接收大小寄存器)

[RO] [0x0026+0x0100*(n+4), 0x0027+0x0100*(n+4)] [0x0000]

Sn_RX_RSR 表示在 SOCKET n 接收缓冲区中接收的数据大小。

在TCP, UDP, IPRAW 或MACRAW模式下, Sn_RX_RSR = | Sn_RX_WR(1) – Sn_RX_RD(2) |

(1) SOCKET n RX写指针寄存器
(2) SOCKET n RX读指针寄存器

例如) S0_RX_RSR = 2048 (0x0800)

S0_RX_RSR0(0x0426)	S0_RX_RSR1(0x0427)
0x08	0x00

3.2.19 Sn_RX_RD (SOCKET n 接收读指针寄存器)

[RW] [0x0028+0x0100*(n+4), 0x0029+0x0100*(n+4)] [0x0000]

Sn_RX_RD 是 SOCKET n 接收缓冲区中读取的最后一个数据地址。SOCKET n 接收缓冲区中可读数据范围从 Sn_RX_RD 到 Sn_RX_WR。在读取完接收到的数据后，将 Sn_RX_RD 的值更新为所读数据大小，主机必须设置 Sn_CR [RECV]以更新 Sn_RX_RD。

例如) S0_RX_RD =1536(0x0600)

S0_RX_RD0(0x0428)	S0_RX_RD1(0x0429)
0x06	0x00

3.2.20 Sn_RX_WR (SOCKET n 接收写指针寄存器)

[RO] [0x002A+0x0100*(n+4), 0x002B+0x0100*(n+4)] [0x0000]

Sn_RX_WR 是 SOCKET n 接收缓冲区中最后收到的数据地址。
如果收到的数据小于 Sn_RX_RSR，则将其存储在 SOCKET n 接收缓冲区中，并且 Sn_RX_WR 增加已存储的数据的大小。

例如) S0_RX_WR = 1536(0x0600)

S0_RW_WR0(0x042A)	S0_RW_WR1(0x042B)
0x06	0x00

3.2.21 Sn_IMR (SOCKET n 中断屏蔽寄存器)

[R=W] [0x002C+0x0100*(n+4)] [0xFF]

Sn_IMR 用于屏蔽相应的 Sn_IR 中断位。

7	6	5	4	3	2	1	0
-	-	-	SENDOK	TIMEOUT	RECV	DISCON	CON
-	-	-	R=W	R=W	R=W	R=W	R=W

位	符号	描述
[7:5]	-	保留
4	SENDOK	**屏蔽 Sn_IR[SENDOK]中断**
3	TIMEOUT	**屏蔽 Sn_IR[TIMEOUT]中断**
2	RECV	**屏蔽 Sn_IR[RECV]中断**
1	DISCON	**屏蔽 Sn_IR[DISCON]中断**
0	CON	**屏蔽 Sn_IR[CON]中断**

3.2.22 Sn_FRAGR (SOCKET n IP 包头片段偏移寄存器)

[R=W] [0x002D+0x0100*(n+4), 0x002E+0x0100*(n+4)] [0x4000]

Sn_FRAGR 配置 IP 报头中的片段偏移量。

例如) S0_FRAG0 = 0x0000 (Don’ t Fragment)

S0_FRAGR0(0x042D)	S0_FRAGR1(0x042E)
0x00	0x00

3.2.23 Sn_MR2 (SOCKET n 模式寄存器 2)

[R=W] [0x002F+0x0100*(n+4)] [0x00]

Sn_MR2 配置 SOCKET n 选项。 Sn_MR2 必须在 Sn_CR [OPEN] ='1'之前置位。

7	6	5	4	3	2	1	0
-	MBBLK	MMBLK	IPV6BLK	-	-	BRDB	UNIB
-	R=W	R=W	R=W	-	-	R=W	R=W

位	符号	描述
7	-	保留
6	MBBLK	**在 MACRAW 模式下广播阻止** 在 MACRAW 模式下，该位设置为'1'以阻止广播包。 0：禁用广播阻止 1：启用广播阻止
5	MMBLK	**MACRAW 模式下的组播阻止** 在 MACRAW 模式下，该位设置为'1'来阻止组播数据包。 0：禁用组播阻止 1：启用组播阻止
4	IPV6BLK	**MACRAW 模式下的 IPv6 数据包阻止** 在 MACRAW 模式下，该位设置为'1'来阻止 IPv6 数据包。 0：禁用 IPv6 阻止 1：启用 IPv6 阻止
[3:2]	-	保留
1	BRDB	**在 UDP 模式下广播阻止/在 TCP 模式下强制 PSH** * UDP 模式下的广播阻止 在 UDP 模式下，该位设置为'1'以阻止 UDP 广播数据包。 0：禁用广播阻止 1：启用广播阻止 *在 TCP 模式下强制 PSH 在 TCP 模式下，该位设置为'1'以在所有发送数据包中设置 PSH 标志。 1：强制 PSH 标志 0：没有强制 PSH 标志
0	UNIB	**UDP 组播模式下的单播阻止** 在 UDP 组播模式下，该位设置为'1'以阻止 UDP 单播数据包。 0：禁用单播阻止 1：启用单播阻止

3.2.24 Sn_KPALVTR (SOCKET n Keep Alive 计时器寄存器)

[RO] [0x0030+0x0100*(n+4)] [0x00]

Sn_KPALVTR 设置'Keep Alive（KA）'数据包传输时间。 单位是 5 秒。

TCP 模式 SOCKET 每隔 Sn_KPALVTR 发送 KA 数据包。

在发送 KA 数据包之前，SOCKET 必须至少发送一次数据（超过 1 字节）数据包。

通过 Sn_CR [SENDKEEP]，在可以在没有设置 Sn_KPALVTR 的情况下发送 KA 数据包。

例如) S0_KPALVTR = 10 (0x0A),

10 * 5s = 50s

S0_KPALVRT(0x0430)
0x0A

3.2.25 Sn_RTR (SOCKET n 重传超时时间寄存器)

[R=W] [0x0032+0x0100*(n+4), 0x0033+0x0100*(n+4)] [0x0000]

Sn_RTR 寄存器设置 SOCKET n TCP 模式下的发送超时时间。

如果 Sn_RTR 寄存器被配置为 ‘0’， SOCKET n 发送超时时间由 RTR 寄存器决定。

(参考 *4.7 重传*)

例如） S0_RTR = 5000 (0x1388),

5000 * 100us = 0.5s

S0_RTR0(x0432)	S0_RTR1(0x0433)
0x013	0x88

3.2.26 Sn_RCR (SOCKET n 重传次数寄存器)

[R=W] [0x0034+0x0100*(n+4)] [0x00]

Sn_RCR 寄存器设置 SOCKET n TCP 模式下的发送重传次数。

如果 Sn_RCR 寄存器被配置为 ‘0’， SOCKET n 重传次数由 RCR 寄存器决定。

(参考 *4.7 重传*)

4 功能描述

W5100S 通过配置寄存器实现网络通信。在本章中，将通过分析代码来逐步介绍“W5100S 的初始化”，“TCP,UDP，IPRAW 和 MACRAW 等通讯过程”，“附加功能实现”等过程。

4.1 W5100S 复位

- 复位前的硬件配置。 (参考 *7.4.1 复位时序*)
- 通过 MOD[3:0]引脚配置主机接口模式。
- 在 RSTn 引脚上提供超过 500ns 的复位信号。
- 等待 T_{STA} (参考 *7.4.1 复位时序*)。

4.1.1 初始化

在 W5100S 初始化过程中需要配置网络参数和设置 SOCKET n TX / RX 缓冲区。

4.1.2 基本设置

操作 W5100S，需要在 MCU 应用程序中设置以下寄存器。

- 模式寄存器（MR)
- 中断屏蔽寄存器（IMR)
- 重传超时时间值寄存器（RTR)
- 重传次数寄存器（RCR)

有关上述寄存器的更多详细信息，可在每个寄存器描述中找到。

4.1.3 网络信息设置

设置网络通信的基本网络信息

```
网络设置:
{
    /* W5100S MAC 地址, 11:22:33:AA:BB:CC */
    SHAR[0:5] = { 0x11, 0x22, 0x33, 0xAA, 0xBB, 0xCC };

    /* W5100S 网关 IP 地址, 192.168.0.1 */
    GAR[0:3] = { 0xC0, 0xA8, 0x00, 0x01 };
```

```
    /* W5100S 子网掩码地址, 255.255.255.0 */
    SUBR[0:3] = { 0xFF, 0xFF,, 0xFF, 0x00};

    /* W5100S IP 地址, 192.168.0.100 */
    SIPR[0:3] = {0xC0, 0xA8,0x00, 0x64};
}
```

4.1.4 SOCKET 发送/接收缓冲区设置

利用 TMSR/RMSR 或 Sn_TXBUF_SIZE/Sn_RXBUF_SIZE 确定 SOCKET n TX / RX 缓冲区大小。SOCKET n TX / RX 缓冲区是一个环形缓冲区结构。因此，必须精确计算 SOCKET n TX / RX 缓冲区控制的基地址和 MASK 值。

请确保 SOCKET n TX / RX 缓冲区大小的总和不超过 8 KB。

以下是设置 SOCKET n TX/RX 缓冲区的代码描述：

```
In case of, assign 2KB TX/RX memory per SOCKET
{
  //设置 SOCKET n 的 TX / RX 存储器的基地址
  gS0_RX_BASE = 0x8000;   // TX 存储器基地址
  gS0_RX_BASE = 0xC000;   // RX 存储器基地址
  TxTotalSize = 0;              //用于检查 SOCKET n 发送缓冲区的总大小
  RxTotalSize = 0;              //用于检查 SOCKET n RX 缓冲区的总大小

  for (n=0; n<3; n++)  {
    Sn_TXBUF_SIZE = 2; // 为每个 SOCKET 各分配 2K 的发送缓冲区
    Sn_RXBUF_SIZE = 2; // 为每个 SOCKET 各分配 2K 的 RX 缓冲区
    // 0x07FF, for getting offset address within assigned SOCKET n TX/RX memory
    gSn_TX_MASK = (1024 * Sn_TXBUF_SIZE) – 1;
    gSn_RX_MASK = (1024 * Sn_RXBUF_SIZE) - 1;
    if( n != 0)  {
      gSn_TX_BASE = gSn-1_TX_BASE + (1024 * Sn-1_TXBUF_SIZE);
      gSn_RX_BASE = gSn-1_RX_BASE + (1024 * Sn-1_RXBUF_SIZE);
    } // end if
    TxTotalSize = TxTotalSize + Sn_TXBUF_SIZE;
    RxTotalSize = RxTotalSize + Sn_RXBUF_SIZE;
    If( TxTotalSize > 8 or RxTotalSize > 8 )  goto ERROR;  //TX/RX 缓冲区分别不能超过
8K
  } // end for
}
```

4.2 TCP

TCP（传输控制协议）是基于传输层上的双向数据传输协议。它通过使用端口号提供应用程序之间的通信。

TCP 通信需要连接过程，例如向对方发送连接请求或从对方接收连接请求。

在 TCP 连接过程中，发送连接请求的一方是'TCP CLIENT'，而接收连接请求的一方是'TCP SERVER'。

TCP 可提供可靠，有序和有异常判断的基于 IP 网络的应用层间的数据流通讯。

TCP 服务器“和”TCP 客户端“在 TCP 连接终止前，会一直维持通讯连接。

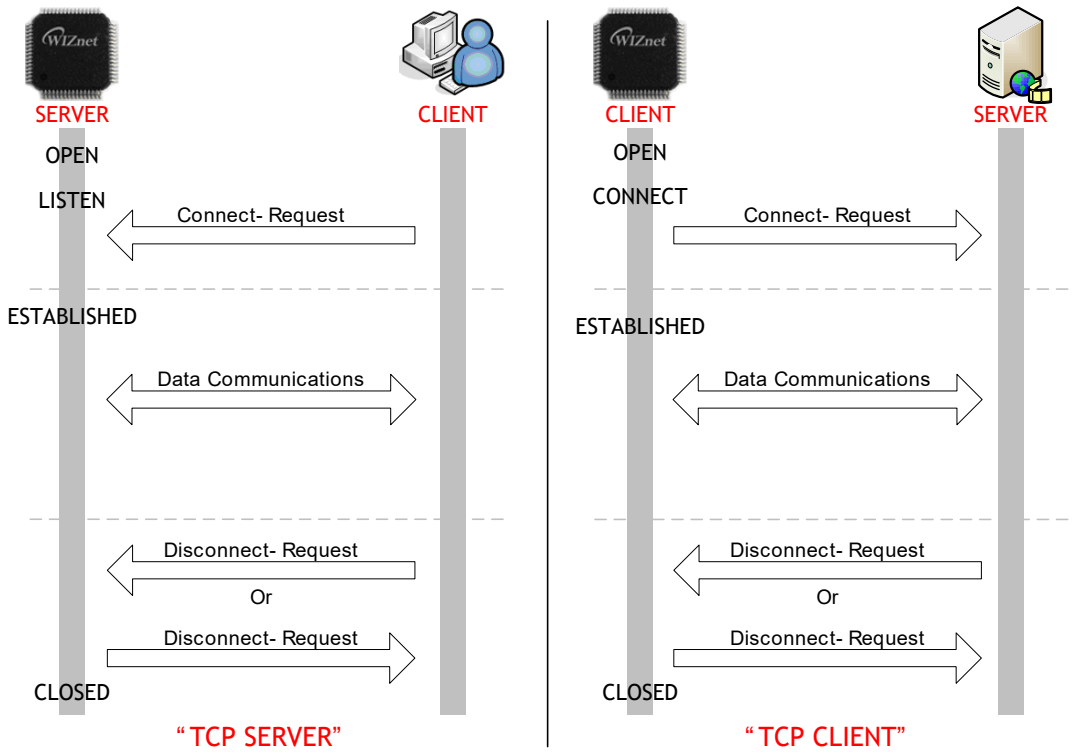


图 5 TCP 服务器 和 TCP 客户端

4.2.1 TCP Server

如图 6 所示‘TCP 服务器’操作流程 ‘

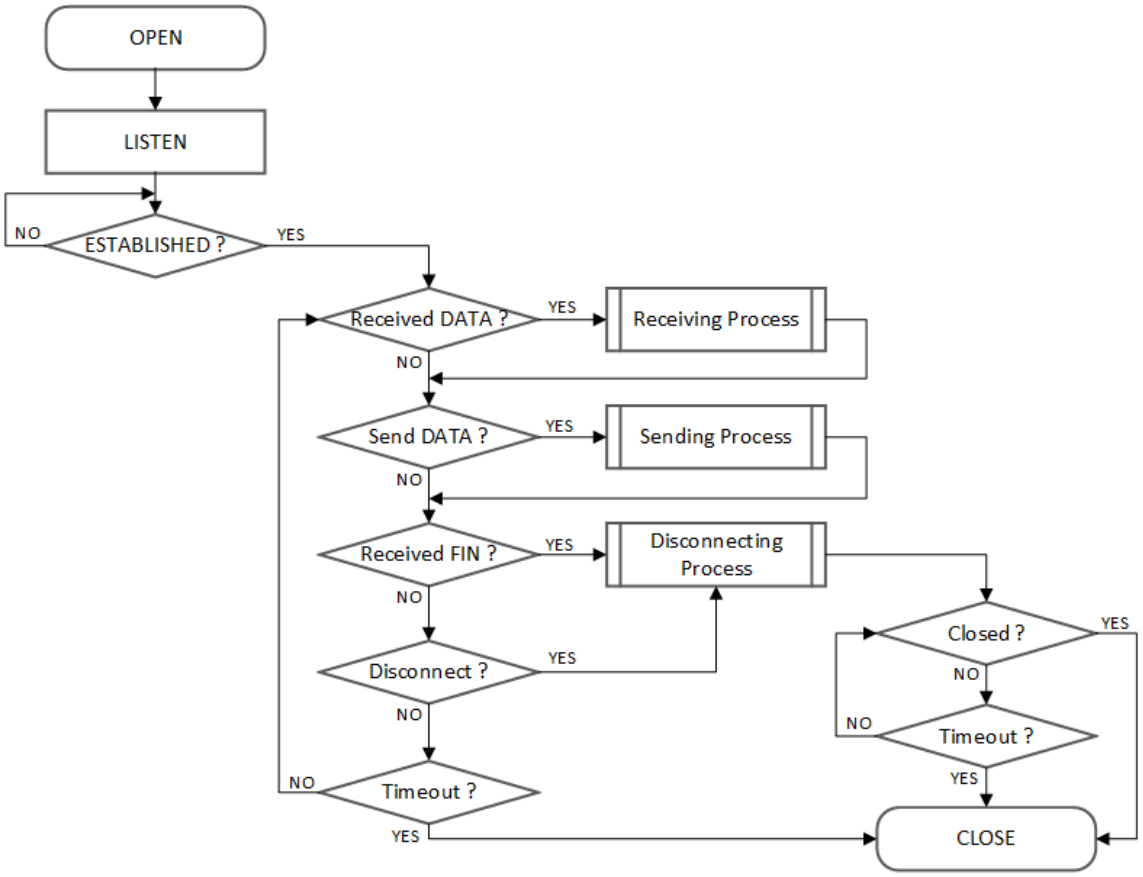


图 6 TCP 服务器操作流程

•打开

打开/端口初始化，并将 SOCKET n 配置为 TCP 模式

```
{
开始：
  Sn_MR[3:0] = ‘0001’; /* 设置 TCP 模式*/
  Sn_PORTR[0:1] = {0x13,0x88}; /* 设置端口号, 5000(0x1388) */

  /* configure SOCKET Option when you need it。必要时，可以配置 SOCKET 选项 */
  // Sn_MR[ND] = ‘1’; /* 配置无延时 ACK */

  Sn_CR[OPEN] = ‘1’; /* 设置 OPEN 命令*/
  while(Sn_CR != 0x00); /* 等待 OPEN 命令 自动清零 */

  if(Sn_SR != SOCK_INIT) goto START; /*检查 SOCKET 状态*/
}
```

- **监听**

配置 Sn_CR [LISTEN]后，SOCKET n 将作为'TCP SERVER'。

TCP SERVER 保持 Sn_SR（SOCK_LISTEN）状态直到收到 TCP Client 发送的 SYN 数据包。

```
{
  Sn_CR = LISTEN; /* 设置 LISTEN 命令*/
  while(Sn_CR != 0x00); /*等待 LISTEN 命令被清除*/
  if(Sn_SR != SOCK_LISTEN) goto OPEN; /* 检查 SOCKET 状态*/
}
```

- **建立连接**

‘TCP 服务器’ 会保持 Sn_SR（SOCK_LISTEN）状态，直到收到 SYN 包。 如果'TCP SERVER'从'TCP CLIENT'接收到 SYN 数据包，它将 SYN / ACK 数据包发送到'TCP CLIENT'，'TCP SERVER'和'TCP CLIENT'之间的连接握手过程完成。

如果在重传超时时间内没有来自对方所发送的 SYN 包或 SYN / ACK 包的响应，则 Sn_IR [TIMEOUT]位被设置为'1'。

```
第一种方法：
{
    /* 检查 SOCKET 中断 */
    if (Sn_IR[CON]  ==  ‘1’)
    {
        /* 清除 SOCKET 中断*/
        Sn_IR[CON] =  ‘1’;
        goto Received DATA?;  /* 或者转到 接收数据; */
    }
    else if(Sn_IR[TIMEOUT] ==  ‘1’)  goto Timeout?;
}
```

```
第二种方法：
{
    if (Sn_SR == SOCK_ESTABLISHED)
    {
        /* 清除 SOCKET 中断*/
        Sn_IR[CON] =  ‘1’;
        goto Received DATA?    /*或者转到 接收数据; */
    }
    else if(Sn_IR[TIMEOUT] ==  ‘1’) goto Timeout?;
}
```

- **接收数据**

通过以下方法判断 Sn_IR [RECV]或 Sn_RX_RSR 是否接收到 SOCKET n 的数据。

```
第一种方法：
{
  /*检查 SOCKET RX 存储器接收到数据的大小*/
  if (Sn_RX_RSR > 0)   goto Receiving Process;
}
```

```
Second method :
{
  if (Sn_IR[RECV] ==  '1' )
  {
    /*检查 SOCKET 中断*/
    Sn_IR[RECV] =  '1' ;     /*  清除 SOCKET  中断*/
    goto Receiving Process;
  }
}
```

- **接收处理**

从 SOCKET n 接收缓冲区中读取接收的数据。

RX 存储器模块中接收数据的读取偏移地址通过 gSn_RX_BASE, gSn_RX_MASK 和 Sn_RX_RD 计算。 (参考 *4.1.4 SOCKET 发送/接收缓冲区设置*)。

在读取接收到的数据后，Sn_RX_RD 寄存器必须增加所读取数据的大小，并且必须将 Sn_CR [RECV]设置为'1'。 如果在执行 Sn_CR [RECV]命令之后 SOCKET n 接收缓冲区中仍有数据，则 Sn_IR [RECV]被设置为'1'。

计算读取偏移地址时，需注意是否溢出

SOCKET n 接收缓冲区的边界地址（n=0,1,2 : gSn_RX_BASE ~ gSn+1_RX_BASE, n=3 : gS3_RX_BASE ~ 0xFFFF）。

```
{
    /*获取接收大小 */
    get_size = Sn_RX_RSR;

    /*  计算 SOCKET n RX  缓冲区大小和偏移地址  */
    gSn_RX_MAX = Sn_RXBUF_SIZE * 1024;
    get_offset = Sn_RX_RD & gSn_RX_MASK;

    /*  计算读数据的偏移地址  */
    get_start_address = gSn_RX_BASE + get_offset;
```

```
    /*如果超过缓冲区存储数据的最大值 */
    If( (get_offset + get_size) > gSn_RX_MAX )
    {
      /*复制 upper_size 字节数的数据到目标地址*/
      upper_size = gSn_RX_MAX – get_offset;
      memcpy(get_start_address, destination_address, upper_size);
      destination_address += upper_size;
      /* 复制剩余的数据到目标地址*/
      remained_size = get_size – upper_size;
      memcpy(gSn_RX_BASE, destination_address, remained_size);
    }
    else
    {
      /* 把起始地址数据复制到目标地址*/
      memcpy(get_start_address, destination_address, get_size);
    }

    /* 增加 Sn_RX_RD 的长度和 get_size 相同 */
    Sn_RX_RD += get_size;

    /* 设置 RECV 命令*/
    Sn_CR[RECV] = ‘1’;
    while(Sn_CR != 0x00); /* 等待 RECV 命令被清除*/
}
```

- **发送数据 / 发送处理**

在发送 SOCKET n 发送缓冲区中写入数据

TX 存储区中写偏移地址是通过 gSn_TX_BASE, gSn_TX_MASK 和 Sn_TX_WD 计算的。(参考 *4.1.4 SOCKET 发送/接收缓冲区设置*)。 并且从写偏移地址开始发送被写入的数据。在写入数据之后，Sn_TX_WD 寄存器值必须增加被写入数据的大小，并通过 Sn_CR [SEND]发送数据。

只有 Sn_IR[SENDOK]= ‘1’之后，下一个数据传输过程才会被执行。在传输数据之后，到达 Sn_IR[SENDOK= ‘1’]需要的时间，取决于打开的 SOCKET 数目、数据大小和网络流量。也会发生 Sn_IR [TIMEOUT] (Ref *4.7.2 TCP 重传*)

当计算偏移地址时，需注意是否溢出 SOCKET n 发送缓冲区的边界地址（(n=0,1,2 : gSn_TX_BASE ~ gSn+1_TX_BASE, n=3 : gS3_TX_BASE ~ 0xC000) 。

如果在重传超时时间内数据接收方对发送的数据包没有响应，那么 Sn_IR [TIMEOUT]将被设置为'1'。

```
{
  /* 计算 SOCKETn TX 缓冲区大小和偏移地址*/
  gSn_TX_MAX = Sn_TXBUF_SIZE * 1024;
  get_offset = Sn_TX_WR & gSn_TX_MASK;

  /* 检查可发送数据最大值*/
  if( send_size > gSn_TX_MAX ) send_size = gSn_TX_MAX;
  while(send <= Sn_TX_FSR);   // 等待发送缓冲区空闲 */
  /* If you don't want to wait TX Buffer Free
     send_size = Sn_TX_FSR; // write Data as size of Free Buffer
  */

  /*计算写入偏移地址 */
  get_start_address = gSn_TX_BASE + get_offset;

  /*如果溢出发送缓冲区大小 */
  If( (get_offset + send_size) > gSn_TX_MAX )
  {
    /* 复制 upper size bytes of source_address to get_start_address
       - source_address is the start address of user data */
    upper_size = gSn_TX_MAX – get_offset;
    memcpy(source_address, get_start_address, upper_size);

    /* copy the Remained Size Bytes of source_address to gSn_TX_BASE */
    source_address += upper_size;
    remained_size = send_size – upper_size;
    memcpy(source_address, gSn_TX_BASE, remained_size);
  }
  else
  {
    /* copy send_size bytes of source_address to get_start_address
     - source_address is the start address of user data */
    memcpy(source_address, get_start_address, send_size);
  }
```

```
  /* increase Sn_TX_WR as send_size */
  Sn_TX_WR += send_size;

  /* 设置 SEND 命令*/
  Sn_CR = SEND;
  while(Sn_CR != 0x00); /* 等待 SEND 命令被清除*/

  /*等待发送命令完成或超时中断发生*/
  while(Sn_IR[SENDOK] == '0' and Sn_IR[TIMEOUT] = '0');

  /* 清除 SOCKET 中断*/
  if(Sn_IR[SENDOK] == '1') Sn_IR[SENDOK] = '1';
  else goto Timeout?;
}
```

- **断开连接（被动断开）**

当接收到对方发送的的 FIN 包时，需要处理 SOCKET 关闭的流程。

```
第一种方法:
{
    If(Sn_SR == SOCK_CLOSE_WAIT) goto Disconnecting Process;
}
```

```
第二种方法:
{
     If(Sn_IR[DISCON] == '1') goto Disconnecting Process;
}
```

- **断开连接（主动断开）**

当发送 FIN 包到对方时

```
{
     /* 发送 FIN 数据包*/
     Sn_CR[DISCON] = '1';
     while(Sn_CR != 0x00); /* 等待 DISCON 命令被清除*/
     goto Disconnecting Process;
}
```

- **断开连接过程（Disconnecting Process）**

在被动关闭模式下，如果接收到从对方发来的 FIN 数据包，并且没有数据要发送时，则 SOCKET 将发送 FIN 数据包后并将其关闭。 如果在重传超时时间内对方没有对发送的 FIN 数据包进行响应，则 Sn_IR [TIMEOUT]被设置为'1'。

在主动关闭模式下，如果 SOCKET 将 FIN 数据包传送给对方，则 SOCKET 等待接收对方发送 FIN 数据包。 接收到对方发送的 FIN 数据包后，SOCKET 将被关闭。 如果在重传超时时间内对方没有对发送的 FIN 数据包进行响应，则 Sn_IR [TIMEOUT]被设置为'1'。

```
被动关闭: /* 接收到从对方发来的数据包 */
{
  /*发送 FIN 数据包*/
  Sn_CR = DISCON;
  while(Sn_CR != 0x00); /*等待 DISCON 命令被清除*/

  /* 等待接收 ACK 数据包*/
  while(Sn_IR[DISCON] == ‘0’ and Sn_IR[TIMEOUT] == ‘0’);
  if (Sn_IR[DISCON] == ‘1’)
  {
    /* 清除中断 */
    Sn_IR[DISCON] = ‘1’;
    goto CLOSED;
  }
  else  goto Timeout?;
}
```

```
主动关闭：/* 将 FIN 数据包发送给对方 */
{
  /*等到接收 FIN 数据包*/
  while(Sn_IR[DISCON] == ‘0’ and Sn_IR[TIMEOUT] == ‘0’);
  if (Sn_IR[DISOCN] == ‘1’)
  {
    /* 清除中断 */
    Sn_IR[DISCON] = ‘1’;
    goto CLOSED;
  }
  else  goto Timeout?;
}
```

- **超时**

如果在重传超时时间内对方没有响应发送的 SYN 或 SYN / ACK 或 FIN 或数据包，则 Sn_IR [TIMEOUT]设置为'1'。(参考 *4.7.2 TCP 重传*)

```
{
  /* 检查 TIMEOUT 中断*/
  if(Sn_IR[TIMEOUT] == ‘1’)
  {
    /*清除中断 */
    Sn_IR[TIMEOUT] = ‘1’;
    goto CLOSE;
  }
}
```

- **关闭**

SOCKET n 由 Disconnect 过程关闭，Sn_IR [TIMEOUT] = ‘1’ 和 Sn_CR [CLOSE] = ‘1’。

```
{
  /* 等待 SOCKET n 关闭*/
  while(Sn_SR != SOCK_CLOSED);
}
```

4.2.2 TCP Client

如图 7 所示 TCP CLIENT’ 操作流程

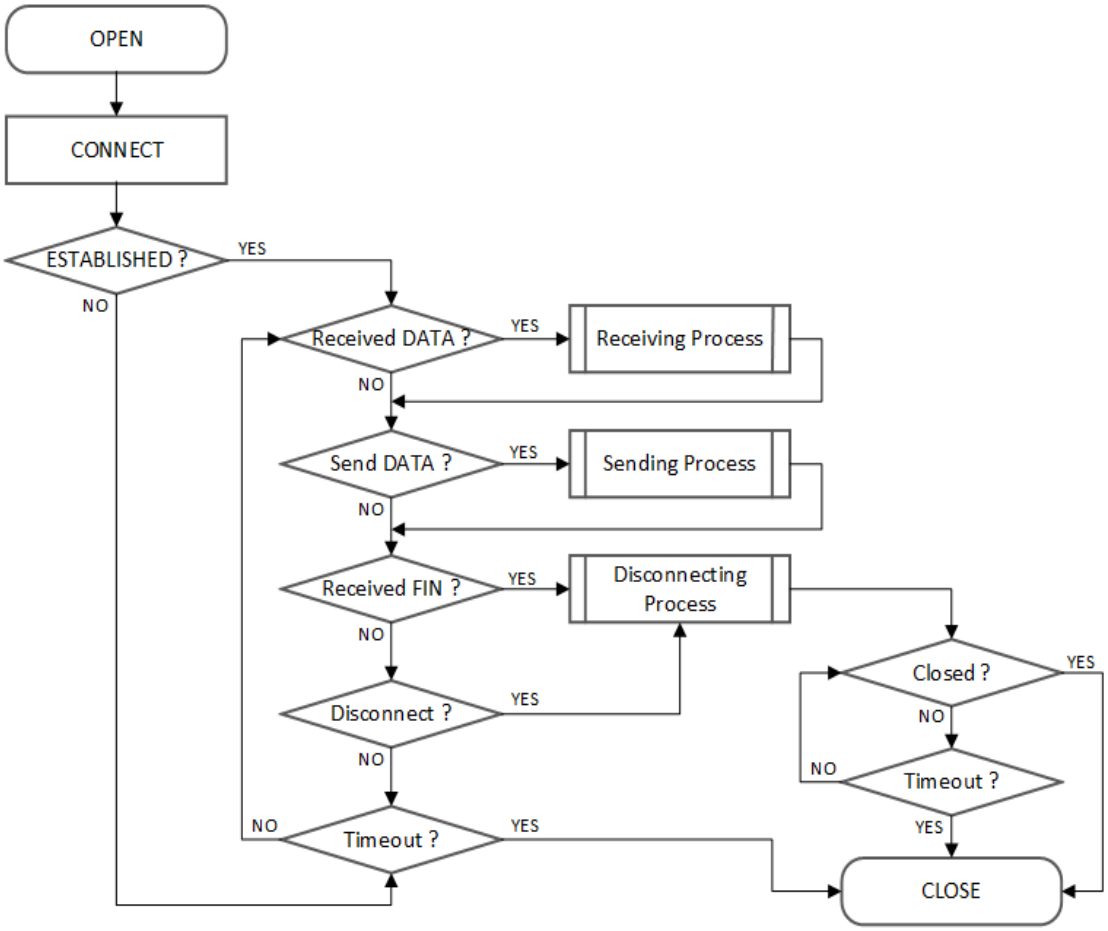


图 7 TCP 客户端 操作流程

- **打开**

参考 *4.2.1 TCP Server : OPEN*

- **连接**

SOCKET 通过 Sn_CR [CONNECT]配置为 TCP Client。同时通过 Sn_CR [CONNECT]命令发送 SYN 包

```
{
    /* 设置目标 IP 地址, 192.168.0.11 */
    Sn_DIPR[0:3] ={ 0xC0, 0xA8, 0x00, 0x0B};

    /* 设置目标端口号, 5000(0x1388) */
    Sn_DPORTR[0:1] = {0x13, 0x88};
```

```
    /* 设置 CONNECT 命令*/
    Sn_CR = CONNECT;
    while(Sn_CR != 0x00); /* 等待 CONNECT 命令被清除*/
    goto ESTABLISHED?;
}
```

- **建立连接**

'TCP CLIENT'处于 Sn_SR（SOCK_SYNSENT）状态直到接收到来自对方'TCP SERVER'的 SYN / ACK 数据包。 如果从'TCP 服务器'收到 SYN / ACK 数据包，'TCP 服务器'和'TCP 客户端'之间的连接过程完成。

如果在重传超时时间内对方没有对发送的 SYN 数据包的响应，则 Sn_IR [TIMEOUT]设置为'1'。(参考 *4.2.1 TCP Server :接收数据*)

- **其他**

参考 *4.2.1 TCP Server* 。

4.2.3 其他功能

4.2.4 TCP SOCKET 设置

在 Sn_CR [OPEN]设置为'1'之前，SOCKET 选项可以通过 Sn_MR 和 Sn_MR2 来设置。

- **无延迟 ACK：Sn_MR [NDACK] ='1'**

如果设置无延迟 ACK（Delayed ACK）选项，则 SOCKET 发送 ACK 数据包中不使用延迟功能。

- **延迟 ACK : Sn_MR [NDACK] = '0'**

如果没有设置无延迟 ACK（No Delayed ACK）选项，SOCKET 在 RTR 或 TCP 窗口大小增加后发送 ACK 数据包给对方。

- **强制 PSH：Sn_MR2 [BRDB] ='1'**

如果强制 PSH 选项被设置，则 SOCKET 将在每个要发送的数据包中添加 PSH 标志。

- **自动 PSH：Sn_MR2 [BRDB] ='0'**

如果强制 PSH 选项没有设置，则 SOCKET 将会在最后一个发送的数据包中添加 PSH 标志。

4.2.5 Keep Alive

Keep Alive (KA) 是通过接收对方发送的消息来检查链接是否依然存在，或防止链接被破坏。Keep Alive 发送前一个数据包的最后一个字节的数据。 因此，必须在 Keep Alive 之前传输大于 1 个字节的数据。

如果在重传超时时间内对方没有响应 KA 数据包，则 Sn_IR [TIMEOUT]被设置为'1'。

KA 数据包发送的周期由 Sn_KPALVTR 设置。 如果 Sn_KPALVTR 为 '0' ，KA 数据包可可以通过 Sn_CR [SEND_KEEP]发送。

4.3 UDP

UDP（用户数据报协议） 是一种不可靠的、无连接的数据传输方式，不能保证 IP 层上方传输层的稳定性。它也提供使用端口号来在应用层间通信的方式。
UDP 可以与多个目标进行通信，并且不需要连接过程。 另一方面，由于 UDP 不能保证可靠性，在数据传输时可以接收来自任何目标的数据，并有数据丢失的情况。 UDP 发送方法基于数据发送/接收范围可分为单播，广播和组播。

如图 8 所示为 UDP 操作流程

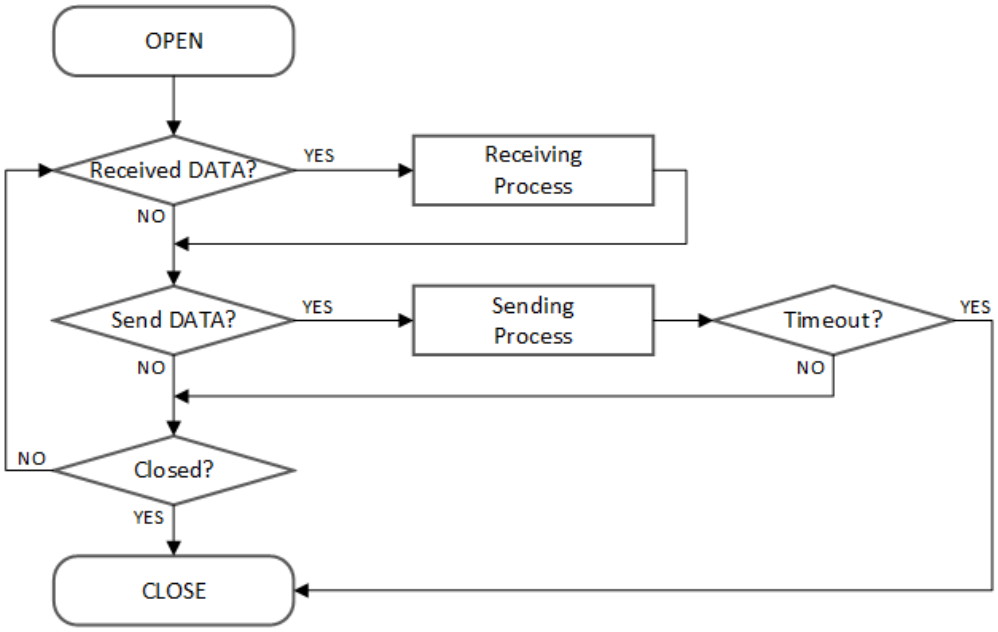


图 8 UDP 操作模式

4.3.1 UDP 单播

UDP 单播是第一种一对一的通信方式。 在数据发送之前，SOCKET 执行 ARP 过程。 在 ARP 过程中，如果在重传超时时间内对端对于 ARP 请求没有响应，Sn_IR [TIMEOUT]将被设置为'1'。
(参考 *4.7.1 ARP 或 PING 重传*)

如果之前的目标和当前目标相同，将会跳过 ARP 过程。 此外，通过 Sn_CR [SEND_MAC]命令和设置 Sn_DHAR 也会跳过 ARP 进程。

- **打开**

通过 OPEN 命令将 SOCKET n 配置为 UDP 模式。

```
{
开始：
  /* 设置 UDP 模式*/
  Sn_MR[3:0] = ‘0010’;

  /*设置源端口号, 5000(0x1388) */
  Sn_PORTR[0:1] = {0x13, 0x88};

  /*设置 SOCKET 选项，如广播模式。 */
  // refer to 3.2.23 Sn_MR2 (SOCKET n 模式寄存器 2)
  // Sn_MR2[BRDB] = ‘1’;

  /* 设置 OPEN 命令*/
  Sn_CR = OPEN;
  while(Sn_CR != 0x00); /* 等待 OPEN 命令被清除*/

  /*检查 UDP 模式的 SOCKET 状态 */
  if(Sn_SR != SOCK_UDP) goto START;
}
```

- **接收数据**

参考 *4.2.1 TCP Server :接收数据*

- **接收处理（Receiving Process）**

在 UDP 模式下，SOCKET 可以接收来自多个目标的数据包。 如图 9 所示，接收到的包含”PACKET INFO”的数据包存储在 SOCKET n RX 缓冲区内，如图 9 所示。如果接收到的数据大于 SOCKET n RX 缓冲区大小，它将被丢弃。

(参考 *4.2.1 TCP Server :接收处理*)

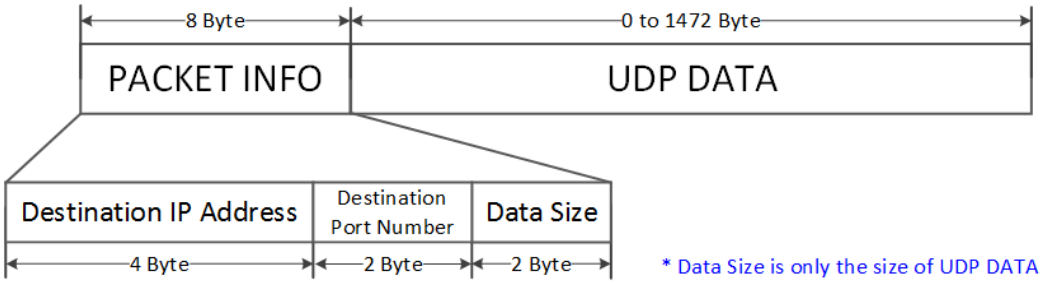


图 9 在 SOCKET n 接收缓冲区中接收到 UDP 数据

```
{
```

```
  /* 接收 PACKINFO */
  goto  4.2.1 TCP Server : Receiving Process 接收处理 with get_size = 8;

  /*提取数据包信息中目标 IP，端口，大小*/
  dest_ip[0:3] = destination_address[0:3];
  dest_port = (destination_address[4] << 8) + destination_address[5];
  data_size = (destination_address[6] << 8) + destination_address[7];

  /* 读取 UDP 数据*/
  goto 4.2.1 TCP Server : Receiving Process 接收处理 with get_size = data_size;
}
```

- **发送数据/发送处理**

参考 *4.2.1 TCP Server :* **发送数据/发送处理**

```
{
    /* 设置目标 IP 地址, 192.168.0.11 */
    Sn_DIPR[0:3] = {0xC0, 0xA8, 0x00, 0x0B};

    /* 设置目标端口号, 5000(0x1388) */
    Sn_PORTR[0:1] = {0x13, 0x88};

    /*用于使用 SEND_MAC 命令: */
    // 参考 4.3.4.1 UDP MAC 发送
    /* 设置目标 MAC 地址, 11:22:33:AA:BB:CC
       Sn_DHAR[0:5] = {0x11, 0x22, 0x33, 0xAA, 0xBB, 0xCC};
    */
    goto 4.2.1 TCP Server : 接收处理;

    /* 使用 SEND_MAC 命令: */
    // 参考 4.3.4.1 UDP MAC 发送
    /*
      goto 4.2.1 TCP Server : Sending Process replaced Sn_CR[SEND] with
      Sn_CR[SEND_MAC];
    */
}
```

- **超时**

如果首次发送数据包给目标方，或者要接收目标的数据包与最后一个目标不同，则在发送数据包之前执行 ARP 进程。

在 ARP 执行过程中，如果在重传超时时间内对于 ARP 请求没有响应，那么 Sn_IR [TIMEOUT]被设置为'1'。

UDP 模式下 SOCKET 不会因 Sn_IR [TIMEOUT]被关闭，因为它支持 1：N 通信模式。(参考 *4.7.1 ARP 或 PING 重传*)

```
{
  /* 检查 SOCKET 状态*/
  if(Sn_IR[TIMEOUT] ==  ‘1’ )
  {
    /*清除中断 */
     Sn_IR[SENDOK] =  ‘1’ ;
     goto Received DATA?    /* or goto Send DATA? or goto CLOSE */
  }
}
```

- **关闭**

当操作完成后，由 Sn_CR [CLOSED]命令关闭。

```
{
    /* 设置 CLOSE 命令*/
    Sn_CR = CLOSE;
    while(Sn_CR != 0x00); /* 等待 CLOSE 被清除*/

    /*等到 SOCKET n 关闭*/
    while(Sn_SR == SOCK_CLOSED);
}
```

4.3.2 UDP 广播

有两种类型的广播。指向网络的广播和指向子网的广播。

UDP 广播是第二种通信方法，将数据发送到相同子网的所有设备。

- **发送数据 /发送处理**

UDP 模式下，当使用广播模式发送数据时，在 Sn_CR [SEND]命令之前，必须将 Sn_DIPR 设置为广播地址。

所有节点广播:

```
{
  /*设置广播地址, 255.255.255.255 */
  Sn_DIPR[0:3] = {0xFF, 0xFF, 0xFF, 0xFF};

  /*  设置目标端口号, 5000(0x1388) */
  Sn_PORTR[0:1] = {0x13,0x88};

  goto 4.2.1 TCP Server : Sending Process 接收处理;
}
```

Subnet Broadcasting : Assume SIPR = “192.168.0.10” & SUBR = “255.255.255.0”

```
{
  /*设置广播地址, 192.168.0.255 */
  Sn_DIPR[0:3] = {0xC0, 0xA8, 0x00, 0xFF};

  /*  设置目标端口号, 5000(0x1388) */
  Sn_PORTR[0:1] = {0x13,0x88};

  goto 4.2.1TCP Server : Sending Process 接收处理;
}
```

4.3.3 UDP 组播

UDP 组播是一对多的通信方法。组播地址范围为 224.0.0.0～239.255.255.255(参考 *IANA_Multicast Address*)，对应的 MAC 地址为 01：00：5E：00：00：00～01：00：5E：FF：FF：FF。 MAC 地址的低 23 位具有与组播组相同的地址。(参考 *rfc1112*)

- **打开**

在 Sn_CR [OPEN]命令之前，必须设置组播组信息和 Sn_MR [MULTI]。

IGMP（Internet 组管理协议）JOIN 信息由 Sn_CR [OPEN]命令发送。
IGMP 版本由 Sn_MR [MS]设置为版本 1 或版本 2.

```
{
开始 :
    /*设置多点传送 MAC 地址, 01:00:5E:00:00:64 */
    Sn_DHAR[0:5] = {0x01, 0x00, 0x5E, 0x00, 0x00, 0x64};
```

```
    /* 设置多点传送 IP 地址, 224.0.0.100 */
    Sn_DIPR[0:3] = {0xE0, 0x00, 0x00, 0x64};

    /* 设置多点传送端口号, 3000(0x0BB8) */
    Sn_DPORTR[0:1] = {0x0B, 0xB8};

    /* 设置 UDP 组播*/
    Sn_MR[MULTI] = ‘1’;

    /* 设置 IGMP 版本*/
    Sn_MR[MC] = ‘1’; /* Sn_MR[MC] = ‘1’ : IGMPv1 , Sn_MR[MC] = ‘0’ :
IGMPv2 */

    /*设置 SOCKET 选项，如单播模式和广播。模式
     参考 3.2.23 Sn_MR2 (SOCKET n 模式寄存器 2) */
    // Sn_MR2[UNIB] = ‘1’;
    // Sn_MR2[BRDB] = ‘1’;

    /* 设置 UDP 模式*/
    Sn_MR[3:0] = 4’ b0010;

    /*设置源端口号, 3000(0x0BB8) */
    Sn_PORTR[0:1] = {0x0B, 0xB8};

    /* 设置 OPEN 命令*/
    Sn_CR = OPEN;

    /*检查 UDP 模式的 SOCKET 状态*/
    if(Sn_SR != SOCK_UDP) goto START;
}
```

- **发送数据/发送处理**

参考 *4.2.1 TCP Server : 发送处理*

4.3.4 其他功能

4.3.4.1 UDP MAC 发送

当目标 MAC 地址已知时，UDP 模式下 SOCKET 通过设置 Sn_DHAR 来发送没有 ARP 进程的数据包。（参考 *4.3.1 UDP 单播：发送数据 / 发送处理* ）

4.3.4.2 UDP SOCKET 设置

在 UDP 模式下，可以收到单播和广播数据包。但是如果 Sn_MR2 [BRDB]设置为'1'，则广播数据包将被阻止接收。

在 UDP 组播模式下，可以接收单播，广播和组播包。 但是，如果 Sn_MR2 [UNIB]或 Sn_MR2 [BRDB]设置为'1'，则单播或广播包被阻塞。

这些屏蔽位必须在 SOCKET 打开 Sn_CR [OPEN]之前设置。

Sn_MR[MULTI]	Sn_MR2[BRDB]	Sn_MR2[UNIB]	Unicast	Multicast	Broadcast
0	0	Don’ t Care	O	X	O
0	1	Don’ t Care	O	X	X
1	0	0	O	O	O
1	0	1	X	O	O
1	1	0	O	O	X
1	1	1	X	O	X

4.3.4.3 端口无法到达

如果对方将 UDP 数据包发送到 W5100S 上不存在的端口上时。 W5100S 自动向对方发送 ICMP 数据包（Destination Port Unreachable）。 但它可能是端口扫描攻击的目标。

但是，如果 MR2 [UDPURB]设置为'1'，则 W5100S 不会发送 ICMP 数据包（Destination Port Unreachable）。

4.4 IPRAW

IPRAW 模式下

SOCKET 支持 Internet 协议（IPv4）层通信，Internet 协议由 Sn_PROTOR 设置。(参考 *IANA Protocol Number*)

在 IPRAW 模式下 SOCKET 不支持 IPv6，TCP 和 UDP 通信。

表 5 在 IPRAW 模式下支持的网络协议

Protocol	Number	Semantic	W5100S Support
ICMP	1	Internet Control Message Protocol	O
IGMP	2	Internet Group Management	O
IPv4	4	IPv4 encapsulation	O
TCP	6	Transmission Control	X
UDP	17	User Datagram	X
IPv6	-	Protocols over IPv6	X
others	-	Other Protocols	O

如果打开 IPRAW 模式下的 SOCKET，并且把 Sn_PROTOR 设置为 ICMP 模式，则 W5100S 不回复对方 PING 请求和 IPRAW 模式的 PING 应答，SOCKET 将对方 PING 请求数据包存储在 SOCKET 接收缓区中。

如图 10 所示为 IPRAW 模式 SOCKET 操作流程。

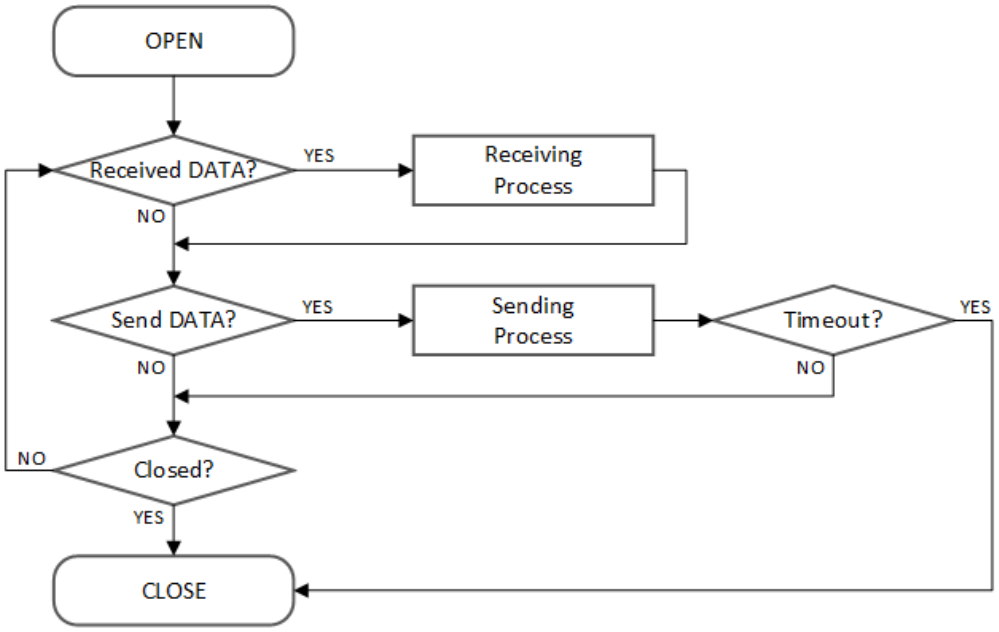


图 10 IPRAW 操作流程

- **打开**

在 IPRAW 模式下打开 SOCKET。

```
{
开始:
  /*设置协议号 */
  Sn_PROTOR = protocol_num;

  /* 设置 IPRAW 模式*/
  Sn_MR[3:0] = “0011”;

  /* 设置 OPEN 命令*/
  Sn_CR[OPEN] = ‘1’;
  while(Sn_CR != 0x00); /* 等待 OPEN 命令被清除*/

  /*检查 SOCKET 是否为 IPRAW 模式
  if(Sn_SR != SOCK_IPRAW) goto START;
}
```

- **接收数据**

参考 *4.2.1 TCP Server：接收数据*

- **接收处理（Receiving Process）**

在 IPRAW 模式下 SOCKET 可以使用和对方完全相同的网络协议并可接收任何目标的数据包。IPRAW 模式下 SOCKET 将“PACKET INFO”的数据存储在 SOCKET RX 缓冲区中，如下图 11 所示。如果接收到的数据大于 SOCKET n 接收缓冲区大小，则将其丢弃。

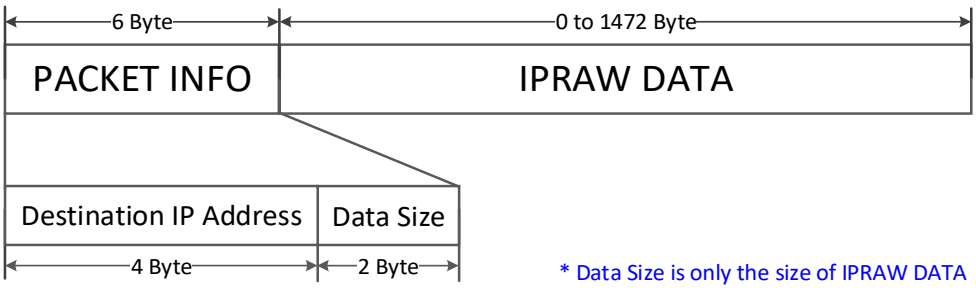


图 11 在 IPRAW 模式下接收数据 SOCKET RX 缓冲区

```
{
  /* 接收 PACKINFO */
  goto 4.2.1TCP Server :接收处理 with get_size = 6;
```

```
  /*提取数据包信息中目标 IP，大小*/
  dest_ip[0:3] = destination_address[0:3];
  data_size = (destination_address[4] << 8) + destination_address[5];

  /* 读取 UDP 数据*/
  goto 4.2.1 TCP Server :接收处理 with get_size = data_size;
}
```

•发送数据/发送处理

要发送的数据不得超过 SOCKET n 发送缓冲区大小。 如果数据大于 IPRAW MSS，则必须将数据分为小于 MSS 的数据。

```
{
    /*设置目标 IP 地址, 192.168.0.11 */
    Sn_DIPR[0:3] = {0xC0, 0xA8, 0x00, 0x0B};

    /* 使用 SEND_MAC 命令:
     refer to 错误!未找到引用源。 错误!未找到引用源。 */
    /*设置目标 MAC 地址, 11:22:33:AA:BB:CC
      Sn_DHAR[0:5] = {0x11, 0x22, 0x33, 0xAA, 0xBB, 0xCC};
    */
    goto 4.2.1 TCP Server :发送处理;

    /* 使用 SEND_MAC 命令:
     refer to 错误!未找到引用源。 错误!未找到引用源。 */
    /*
      goto 4.2.1 TCP Server : Sending Process replaced Sn_CR[SEND] with
      Sn_CR[SEND_MAC];
    */
}
```

- **超时**

参考 *4.3.1 UDP 单播: 超时*

4.5 MACRAW

MACRAW 模式支持以太网 MAC 的数据通信，并且仅只能 SOCKET 0.
在 MACRAW 模式下，SOCKET 0 接收所有以太网数据包或受 Sn_MR [MR]限制的接收以太网数据包（如广播，组播和具有与 SHAR 相同目的地 MAC 地址的数据包）。 并且 Sn_MR2 也可以阻止接收广播，组播和 IPv6 数据包。
在 MACRAW 模式下，SOCKET 0 不会接收到其他 SOCKET 的任何数据包，但会接收到 ARP 请求和 PING 请求（如果 IPRAW 模式 SOCKET 不支持 ICMP）。 即使 MACRAW 模式 SOCKET 0 接收到 ARP 和 PING 请求，W5100S 也会自动发送 ARP 和 PING 响应。

图 12 显示了 MACRAW 模式的 SOCKET 0 操作流程。

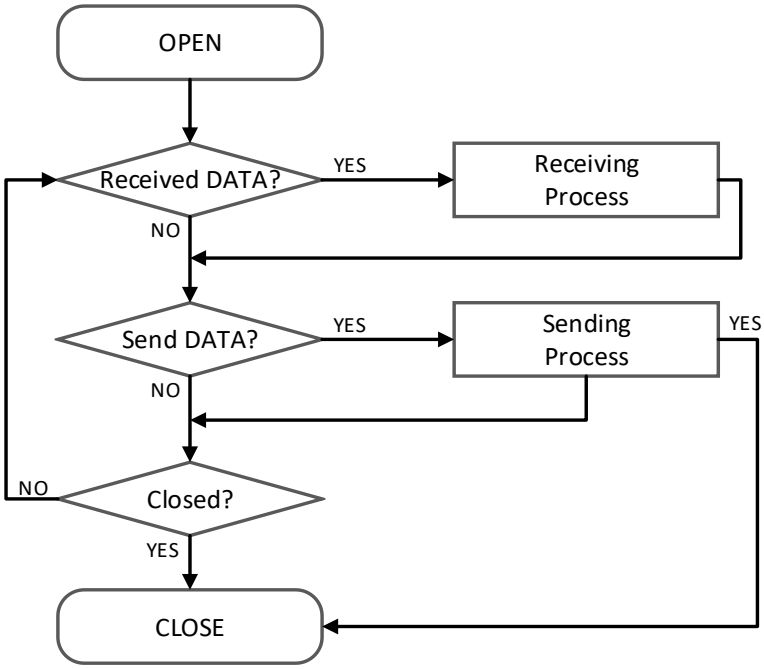


图 12 MACRAW 操作流程

- **打开**

在 MACRAW 模式下打开 SOCKET 0.

```
{
开始:
  /* 设置 MACRAW 模式*/
  S0_MR = “0100”;

  /* MACRAW SOCKET 选项*/
  /*
     S0_MR[MR] = ‘1’;          // Enable MAC Filter
```

```
        S0_MR2[MBBLK] =  '1' ;    // Broadcast Packet Block
        S0_MR2[MMBLK] =  '1' ;    // Multicast Packet Block
        S0_MR2[IPV6BLK] =  '1' ;  // IPv6 Packet Block
    */

    /* 设置 OPEN 命令*/
    S0_CR = OPEN;
    while(Sn_CR != 0x00); /* 等待 OPEN 命令被清除*/

    /*检查 SOCKET0 是否是 MACRAW 模式*/
    if(S0_SR != SOCK_MACRAW) S0_CR = CLOSE; goto START;
}
```

- **接收数据**

参考 *4.2.1 TCP 服务器: 接收数据*

- **接收处理**

在 MACRAW 模式下，SOCKET 0 可以接收来自多个目标的数据包。 MACRAW 模式下 SOCKET 0 将包含“PACKET INFO”的数据存储在 SOCKET 0 RX 缓冲区中，如图 13。所示。

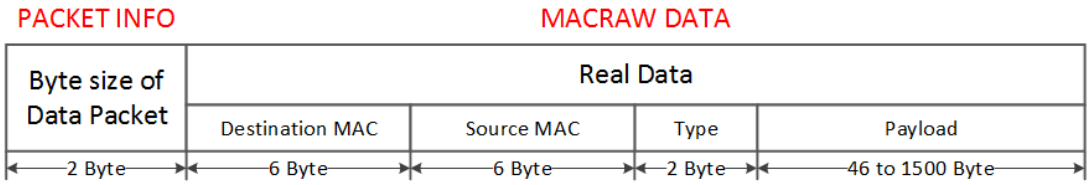


图 13 在 MACRAW 中接收到数据格式

```
{
    /* 接收 PACKINFO */
    goto 4.2.1 TCP Server :接收处理 with get_size = 2;

    /*在信息数据包中提取大小*/
    data_size = (destination_address[0] << 8) + destination_address[1];

    /* 读取 UDP 数据*/
    goto 4.2.1 TCP Server :接收处理 with get_size = data_size;
}
```

- **发送数据/发送处理**

要发送的数据不能超过 SOCKET 0 发送缓冲区的大小。 如果数据大于 MSS，则被 MSS 分片后发送。 如果发送的数据小于 60 字节，则数据为 ‘0 ‘，填充为 60 字节。(参考 *4.2.1 TCP Server：发送数据 /发送处理*)

- **关闭**

参考 *4.3.1 UDP 单播：关闭*

4.6 SOCKET-less 命令 (SLCR)

SOCKET-less 命令（SLCR）通过 SLCR [ARP] ='1'和 SLCR [PING] ='1'发送 ARP 请求和 PING 请求，无需 SOCKET OPEN 命令。

如果收到目标 ARP 和 PING 回复，则 SLIR [ARP]和 SLIR [PING]设置为'1'。

但是，如果在 SOCKET-less 的重传超时时间内没有响应各个请求，则发生超时（SLIR [TIMEOUT] ='1'）。 (参考 *4.7.1 ARP 或 PING 重传*)

图 14 显示了 SOCKET-less 的命令操作流程。

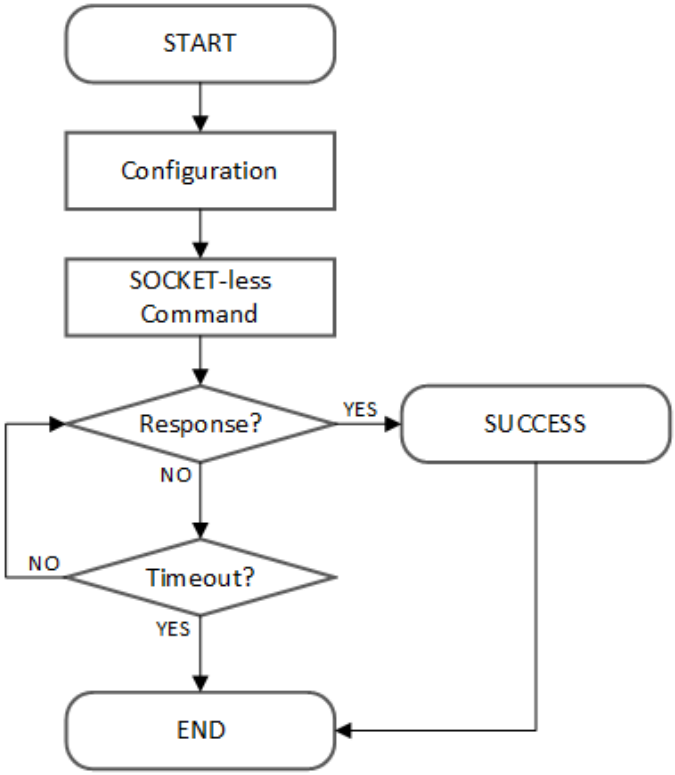


图 14 SOCKET-less 命令操作流程

4.6.1 ARP 请求(SLCR [ARP] = ‘1’)

ARP 请求包由 SLCR [ARP]命令发送数据。 在执行 SLCR [ARP]命令之前，必须在 SLPIPR 中设置目标 IP 地址。 如果收到对方的 ARP 响应，则 SLIR [ARP]设置为'1'，目标 MAC 地址写入 SLPHAR。 如果在 SOCKET-

less 的重传超时时间内没有响应 ARP 请求，则 SLIR [TIMEOUT]设置为'1'。(参考 *4.7.1 ARP 或 PING 重传*

- **配置**

配置 SOCKET-less 的重传超时时间和 ARP 中断屏蔽位和目标 IP 地址。

```
{
开始:
  /* 设置 SOCKET-less 重传超时时间, 100ms(0x03E8) (The unit is 100us) */
  SLRTR[0:1] = {0x03, 0xE8};

  /* 设置 SOCKET-less 重传次数, 5 */
  SLRCR = 0x05;

  /* 设置中断屏蔽位 */
  SLIMR[ARP] = '1'; // ARP 中断屏蔽位
  SLIMR[TIMEOUT] = '1'; // TIMEOUT 中断屏蔽位

  /* 设置目标 IP 地址, 192.168.0.100 */
  SLPIPR[0:3] = {0xC0, 0xA8, 0x00, 0x64};
}
```

- **SOCKET-less 命令**

ARP 请求报文由 ARP 命令发送。

```
{
  /* 设置 ARP 命令*/
  SLCR[ARP] = '1';
  while(SLCR != 0x00) ; /*等待 ARP 命令完成*/
}
```

- **应答**

ARP 中断表示是否收到 ARP 应答。

```
{
  /*检查 ARP 中断*/
  if(SLIR[ARP] == '1') /*收到 ARP 响应报文*/
  {
    /* 清除中断 */
```

```
    SLIR[ARP] =  '1' ;  goto SUCCESS;
  }
  else goto Timeout;
}
```

- **超时**

如果在重传时间内对方没有发送对的 ARP 请求数据包的响应，则发生 TIMEOUT 中断。
(参考 *4.7.1 ARP 或 PING 重传*)

```
{
  /* 检查 TIMEOUT 中断*/
  if(SLIR[TIMEOUT]  == 1)
  {
    /* 清除中断 */
    SLIR[TIMEOUT] =  '1' ;
    goto END;
  }
  else goto Response;
}
```

- **成功**

如果收到 ARP 响应，目标 MAC 地址存储在 SLPHAR 中。

```
{
  /*获取目的地 MAC 地址*/
  destination_mac[0:5] = SLPHAR[0:5];
  goto END;
}
```

4.6.2 PING 命令(SLCR [PING] = '1')

通过 SLCR [PING]命令发送 PING 请求包。 在执行 SLCR [PING]命令之前，必须先设置 SLPIPR，PINGSEQR 和 PINGIDR。 如果接收到目标发送的 PING 回复，则将 SLIR [PING]设置为'1'，并将目标 MAC 地址写入 SLPHAR 中。 如果在 SOCKET-less 的重传超时时间内没有对 PING 请求的响应，SLIR [TIMEOUT]被设置为'1' (参考 *4.7.1 ARP 或 PING 重传*)

•配置

配置 SOCKET-less 的重传超时时间和 PING 中断屏蔽位和目标 IP 地址。

```
{
开始:
  /* 设置 SOCKET-less 重传超时时间, 100ms(0x03E8) (The unit is 100us) */
  SLRTR[0:1] ={ 0x03, 0xE8};

  /* 设置 SOCKET-less 重传次数, 5 */
  SLRCR = 0x05;

  /* 设置中断屏蔽位 */
  SLIMR[PING] = ‘1’;        // PING 中断屏蔽位
  SLIMR[TIMEOUT] = ‘1’;  // TIMEOUT 中断屏蔽位

  /* 设置目标 IP 地址, 192.168.0.100 */
  SLPIPR[0:3] = {0xC0, 0xA8, 0x00, 0x64};

  /* 设置 PING 序列号, 1000(0x03E8) */
  PINGSEQR[0:1] = {0x03, 0xE8};

  /* 设置 PING ID, 256(0x0100) */
  PINGIDR[0:1] = {0x01,0x00};
}
```

• SOCKET-less 命令

PING 由 PING 命令发送请求数据包。

```
{
  /* 设置 PING 命令*/
  SLCR[PING] = ‘1’;
  while(SLCR != 0x00) ; /* 等待 PING 命令完成*/
}
```

•响应

PING 中断表示是否收到 PING 应答。

```
{
  /* 检查 PING 中断*/
  if(SLIR[PING] == ‘1’)  /*收到 PING 响应包*/
  {
```

```
  /* 清除中断 */
    SLIR[PING] = ‘1’;
    goto SUCCESS;
  }
  else goto Timeout;
}
```

- **超时/成功**

参考 *4.6.1 ARP 请求(SLCR [ARP] = ‘1’) 超时/ 成功*。

4.7 重传

4.7.1 ARP 或 PING 重传

当对 ARP 或 PING 请求包没有响应时，就要执行 ARP 或 PING 重传。
在重传过程中，每个 RTR 都会一直重传请求数据包，直到收到响应数据包。 如果重传次数超过 RCR，则请求数据包发生超时（TIMEOUT）。
下表显示重传超时(ARPTO, PINGTO)。。

$$ARP_{TO},\ PING_{TO} = (TIMEOUT_{VAL} \times 0.1ms) \times (TIMEOUT_{CNT} + 1)$$

$TIMEOUT_{VAL}$ = SLRTR or Sn_RTR
$TIMEOUT_{CNT}$ = SLRCR or Sn_RCR

例如） $TIMEOUT_{VAL}$ = 2000(0x07D0), $TIMEOUT_{CNT}$ = 8(0x0008)
ARP_{TO}= 2000 X 0.1ms X 9 = 1.8s

ARPTO 由 SLCR [ARP]，Sn_CR [SEND] 和 Sn_CR [CONNECT]所导致。 并通过 SLIR [TIMEOUT]或 Sn_IR [TIMEOUT]进行检查。
PINGTO 由 SLCR [PING]所导致，并通过 SLCR [TIMEOUT]进行检查。

4.7.2 TCP 重传

在 TCP 模式下，SOCKET 在发送 SYN，FIN 或 DATA 数据包后，未接收到对方响应的 ACK 数据包时，执行 TCP 重传。
在 TCP 重传过程中，每个 Sn_RTR 都会重传请求包，直到收到对方的确认包。 如果重传次数超过 Sn_RCR，则发生 SOCKET n 超时（TIMEOUT）。
下表显示了 TCP 重传 TIMEOUT (TCP_{TO})。

$$TCP_{TO} = \left(\sum_{N=0}^{M} (TIMEOUT_{VAL} \times 2^{N}) + ((TIMEOUT_{CNT} - M) \times TIMEOUT_{最大VAL}) \right) \times 0.1ms$$

N : Number of Retransmission, $0 \leq N \leq M$

M : Minimum value of $TIMEOUT_{VAL} \times 2^{(M+1)}$ >65535 and $0 \leq M \leq TIMEOUT_{CNT}$

$TIMEOUT_{VAL}$ = SLRTR or Sn_RTR

$TIMEOUT_{CNT}$ = SLRCR or Sn_RCR

$TIMEOUT_{最大VAL}$: $TIMEOUT_{VAL} \times 2^{M}$

例如) RTR = 2000(0x07D0), RCR = 8(0x0008)

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

TCP$_{TO}$由 Sn_CR 中的 CONNECT，SEND 和 DISCON 命令发生，并通过 Sn_IR [TIMEOUT]进行检查。

4.8 其他功能

4.8.1 系统时钟（SYS_CLK）切换

SYS_CLK 设置为 25MHz 或 100MHz。 它由 MR2 [CLKSEL]，PHYCR0 [RST]或 PHYCR1 [PWDN]切换。 在切换时钟速度时，主机必须等到 SYS_CLK 稳定后切换（参考 *7.4.1 复位时序*）

MR2[CLKSEL]	PHYCR0[RST]	PHYCR1[PWDN]	SYS_CLK(MHz)
0	0	X	25
0	1	0	100 (Default)
0	1	1	25
1	X	X	25

4.8.2 以太网 PHY 操作模式配置

在 PHY 硬件复位后，通过配置 PHYCR0 设置的 PHY 操作模式（速度，Dupl 例如）。

通过 PHYSR [5：3]进行检查 PHY 操作模式的设置，链路状态在以太网 PHY 连接（Ethernet PHY Link Up。）后在 PHYSR [2：0]上显示。

在 PHYCR0 置为 1 之前，PHYLCKR 必须是解锁状态。

例如）如何设置 PHY 操作模式

```
PHY_10FDX :
{
  /* PHYCR0 & PHYCR1 解锁*/
  PHYLCKR = 0x53;

  /* set PHYCR0 10Mbps/Full-Duplex */
  PHYCR0[DPX] = ‘0’ ;     // 0 – FDX, 1 - HDX;
  PHYCR0[SPD] = ‘1’ ;     // 0 - 100Mbps, 1 – 10Mbps;
  PHYCR0[AUTO] = ‘1’ ;    // 0 – Auto-negotiation, 1 – Manual;

  /* PHY 重置过程*/
  PHYCR1[RST] = ‘0’ ;
  Wait  TPRST;    // 参考 7.4.1 复位时序

  /*等到 PHY 链路启动*/
```

```
  while(PHYSR[LINK] !=  '0' );

  /* 读取 PHYSR */
  If( (PHYSR[DPX] ==  '0' ) & ( PHYSR[SPD] ==  '1' ) & (PHYSR[AUTO] ==  '1' ) )
SUCCESS;
  else FAIL;

  /* PHYCR0 & PHYCR1 锁定*/
  PHYLCKR = 0x00;    // for Lock, write any value
}
```

4.8.3 以太网 PHY 并行检测

当链接伙伴不支持自动协商模式时，内部以太网 PHY 通过并行检测形成链路

如果双工模式不同（如 10F / 10H），则会出现网络性能下降的问题。

PHY \ Link Partner	自动	10H	10F	100H	100F
自动	100F 100F	10H 10H	10F 10H	100H 100H	100F 100H
Manual 10H	10H 10H	10H 10H	10F 10H		
Manual 10F	10H 10F	10H 10F	10F 10F		
Manual 100H	100H 100H			100H 100H	100F 100F
Manual 100F	100H 100F			100H 100F	100F 100F

4.8.4 以太网 PHY Auto-MDIX

以太网 PHY 支持自动协商功能，并由 PHYCR0 [AUTO]设置。 如果设置了自动协商（PHYCR0 [AUTO] ='0'），则支持 Auto-MDIX，并操作对称变压器（图 30 变压器类型）。

如果没有设置自动协商（PHYCR0 [AUTO] ='1'）功能，则由于不支持 Auto-MDIX 功能，因此必须使用交叉 UTP 电缆。

参考）如果任一链接节点支持 Auto-MDIX 功能，则可以使用直通和跨接 UTP 电缆。

4.8.5 以太网 PHY 省电模式

当 PHYCR1 [PWDN]设置为 1 时，以太网 PHY 变为省电模式，SYS_CLK 被设置为 25MHz。

另一方面，如果 PHYCR1 [PWDN]设置为'0'，则以太网 PHY 变为正常模式，SYS_CLK 由 MR2 [CLKSEL]置 1. (参考 *3.1.19 MR2 (模式寄存器 2)*)

```
进入省电模式:
{
  /* PHYCR0 & PHYCR1 解锁*/
  PHYLCKR = 0x53;

  /*启用省电模式 */
  PHYCR1[PWDN]  =  '1' ;

  /* PHYCR0 & PHYCR1 锁定*/
  PHYLCKR = 0x00; // for Lock, write any value 对于锁定，写入任何值

  /*等到时钟稳定切换*/
  Wait TPRST; //参考 7.4.1 复位时序
}

退出掉电功能 :
{
  /* PHYCR0 & PHYCR1 解锁*/
  PHYLCKR = 0x53;

  /* 启用省电功能*/
  PHYCR1[PWDN]  =  '0' ;

  /* PHYCR0 & PHYCR1 锁定*/
  PHYLCKR = 0x00; // for Lock, write any value

  /*等到时钟稳定切换*/
  Wait TPRST; // 参考 7.4.1 复位时序
```

```
  /*等到时钟切换 25 到 100MHz */
  Wait TLF; // 参考 7.4.1 复位时序
}
```

4.8.6 以太网 PHY 的控制寄存器

以太网 PHY 中的寄存器由 MDC / MDIO（管理数据时钟/输入输出）接口访问。 W5100S 具有 MDC / MDIO 控制器，它由 PHYDIVR，PHYRAR，PHYDOR，PHYDIR 和 PHYACR 控制。

例如） BMCR 写入

```
{
  PHYRAR = 0x00;   // BMCR Address 0x00
  PHYDIR = 0x80;   // BMCR[15] =  '1'  , PHY SW Reset

  /* 写入*/
  PHYACR = 0x01;
  while(PHYACR != 0); //等到 MDC / MDIO 控制完成
}
```

例如） BMSR 读取

```
{
  PHYRAR = 0x01; // BMSR Address 0x01

  /* 读取*/
  PHYACR = 0x02;
  while(PHYACR != 0); //等到 MDC / MDIO 控制完成

  if( PHYDOR & 0x0004)
  {
    // LINK UP – BMSR[2] =  '1'
  }
}
```

5 主机接口模式

5.1 SPI 模式

当 MOD [3：0]为“0000”时，W5100S 通过 SPI 接口连接到主机，如图 15 所示。

W5100S 支持 SPI 模式 0 和模式 3，如图 16 所示。
MOSI 在上升沿进行采样，MISO 在下降沿进行切换。 对于每个 SCLK，MOSI 和 MISO 从 MSB 到 LSB 顺序发送和接收。

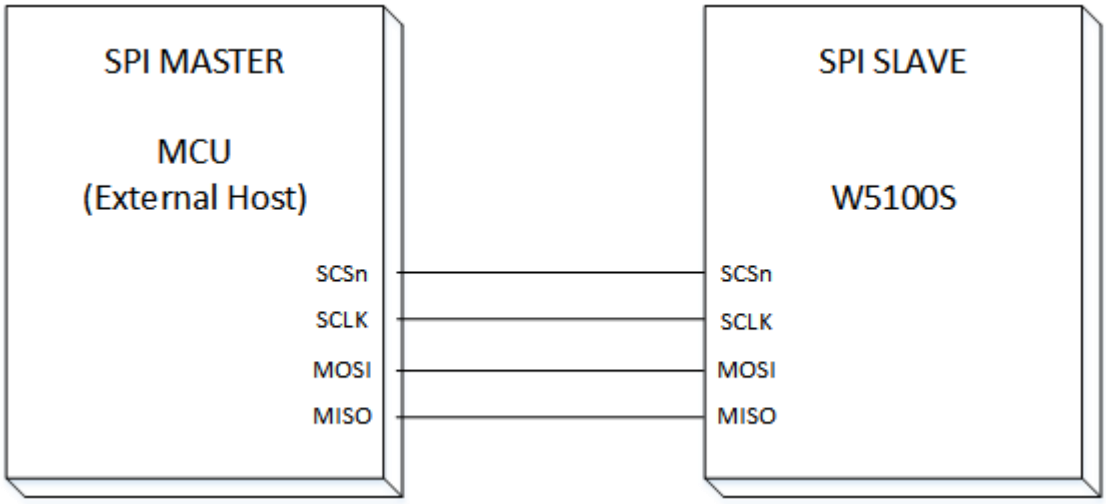


图 15 SCSn 由主机控制

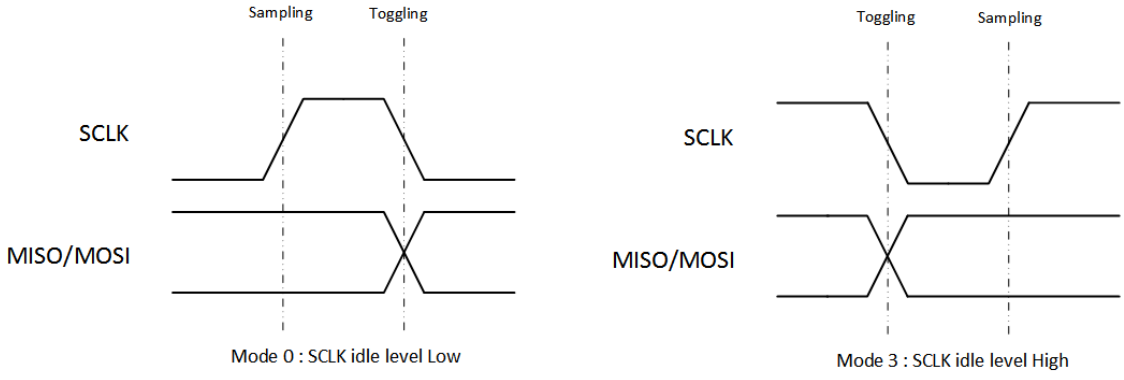


图 16 SPI 模式 0 和 模式 3

5.1.1 SPI 帧

W5100S 由主机发送的 SPI 帧（图 17 SPI 帧）控制。 SPI 帧由控制段，地址段和数据段组成。CSn 引脚必须由主机的 SPI 单元控制。

W5100 仅支持 1 字节数据读/写，但 W5100S 支持顺序 N 字节（N = 1,2,3，。。。）数据读取/写入。

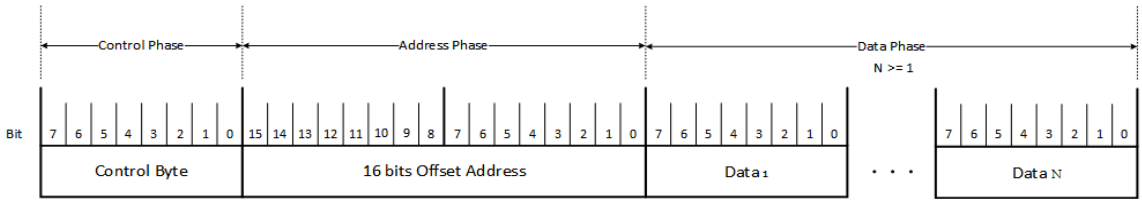


图 17 SPI 帧

- **控制段**

控制段由 8 位组成，用于设置读/写访问类型。
主机将 CSn 管脚由高电平置为低电平，并在 2 个 SYS_CLK 周期后发送控制段。

表 6 W5100S 读写类型

Access 典型 e	值
写	0xF0
读	0x0F

- **地址段**

地址段由 16 位组成，包含设置 W5100S 的寄存器地址或包含 TX / RX 存储器 16 位起始偏移地址。下一次数据访问时，起始偏移地址会自动递增 1.

- **数据段**

数据段由 N-bytes 的数据组成。 主机完成数据访问后，必须在 2 个 SYS_CLK 周期后，将 CSn 从低电平置为高电平。

5.1.2 SPI 写入操作

多字节数据写入帧，如图 18 所示

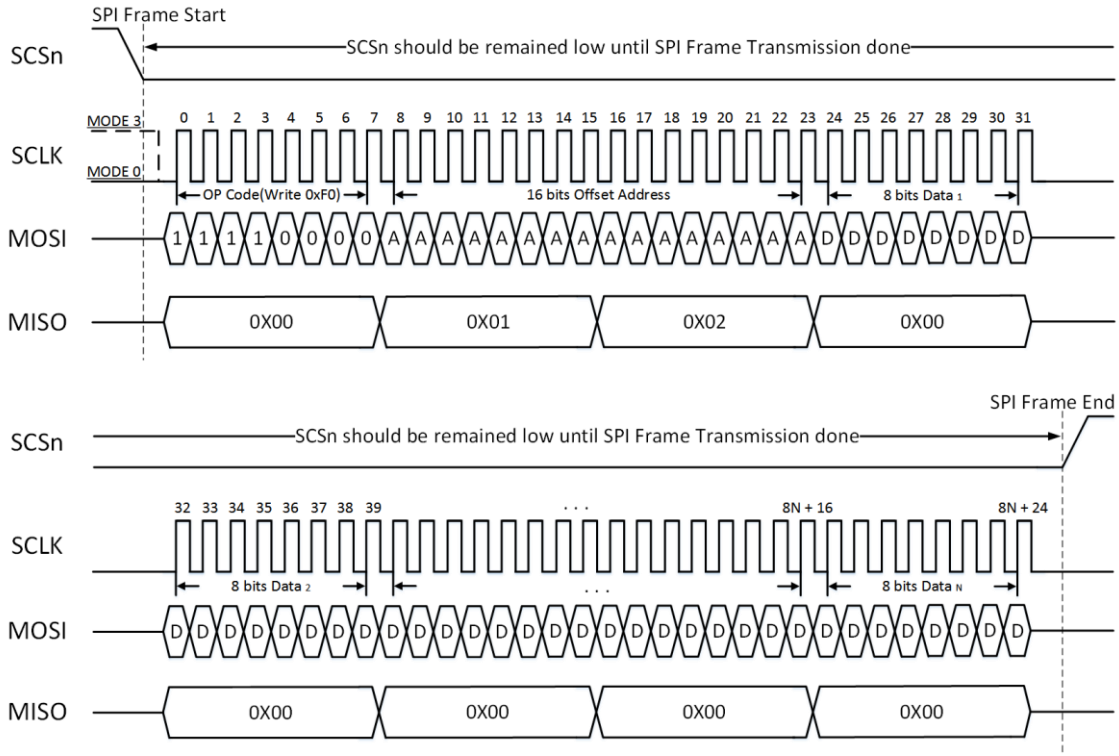


图 18 W5100S 写入 SPI 帧

5.1.3 SPI 读取操作

多字节数据读取帧，如图 19 所示

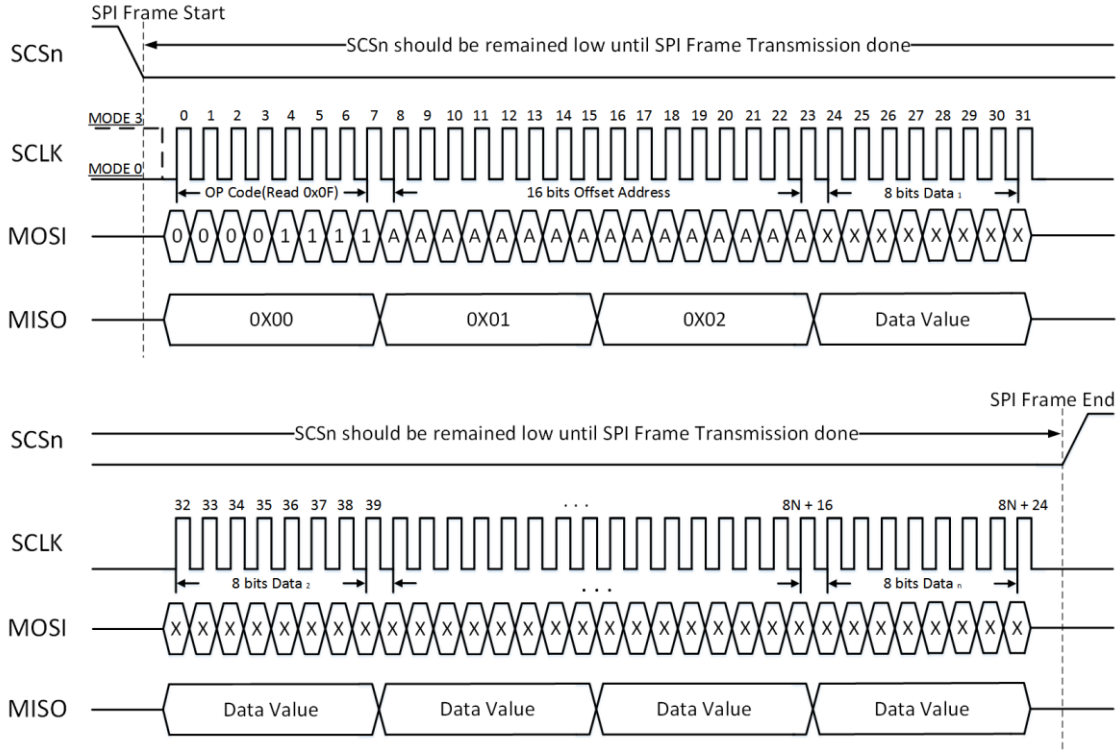


图 19 W5100S 读取 SPI 帧

5.2 并行总线模式

当 MOD [3：0]为“010X”时，主机通过并行总线接口连接 W5100S，如图 20 所示。

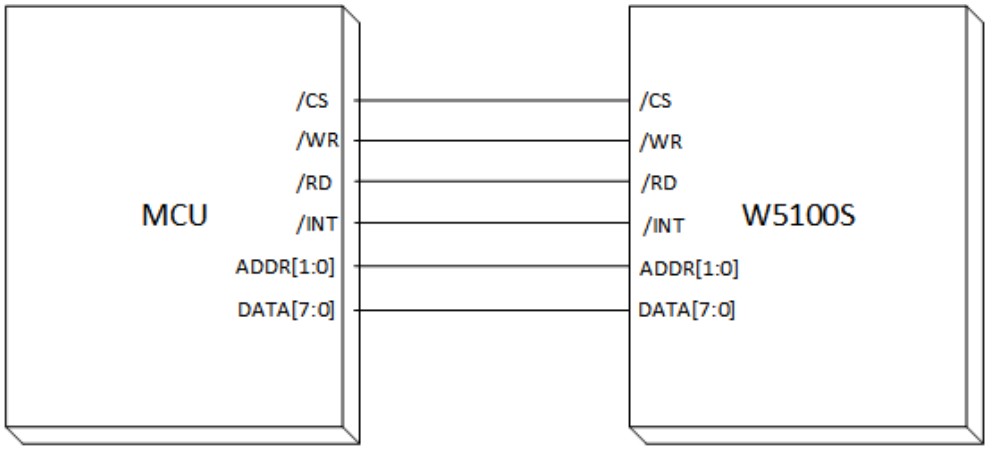


图 20 并行总线直接和间接控制模式

并行接口通过表 7 中的寄存器，访问通用寄存器/SOCKET 寄存器，TX/RX 数据缓冲区。
并支持多字节数据的顺序读写

表 7 间接模式地址值

ADDR[1:0]	符号	描述
00	MR	通用寄存器 MR
01	IDM_ARH	高 8 位偏移地址寄存器
10	IDM_ARL	低 8 位偏移地址寄存器
11	IDM_DR	8 位数据寄存器

5.2.1 并行总线数据写入

多字节数据写入帧，如图 21 所示。

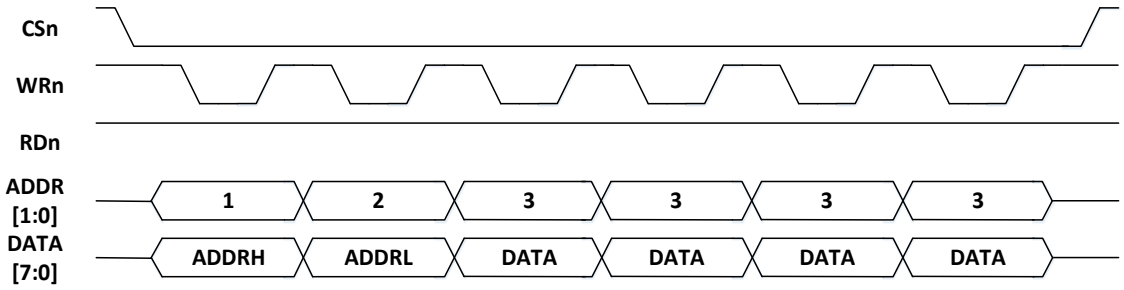


图 21 并行总线连续写入

5.2.2 并行总线数据读取

多字节数据读取帧，如图 22 所示。

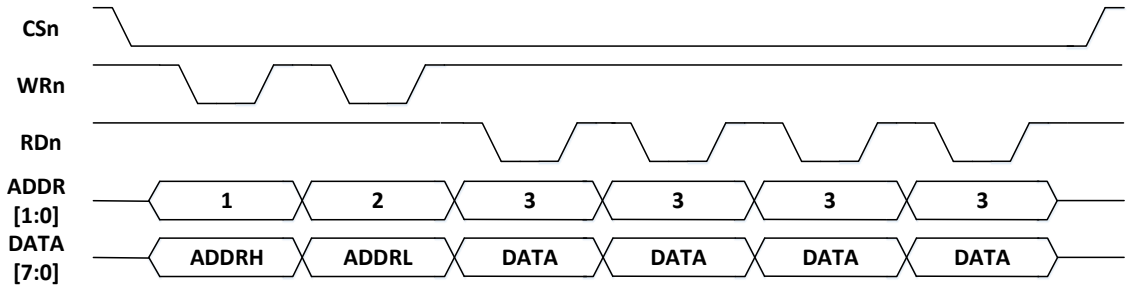


图 22 间接模式连续读取

6 时钟源和变压器要求

6.1 无源晶振特性

表 8 无源晶振特性

参数	条件 / 描述	最小	典型	最大	单位
频率(F)			25		MHz
频率误差	25°C 时	-50		+50	ppm
频率稳定性	1 年。	-50		+50	ppm
负载电容(C_L)	ESR = 30 Ω		12		pF
反馈电阻(R_F)	外部电阻		1M		Ω
启动时间	W5100S 复位			60	ms
Trans-conductance(g_m)			8.43		mA/V
gain margin ($gain_{margin}$)	$gain_{margin} = g_m / g_{mcrit}$	6.99			dB

C_0(1) : The Packaging Parasitic Shunt Capacitance。
C_L(1) : Load Capacitance。 eq) $C_L = (C_{L1} X C_{L2}) / (C_{L1} X C_{L2}) + C_s$
C_{L1}, C_{L2} : External Capacitances of the circuit connected to the crystal (Typically, $C_{L1} = C_{L2}$)
C_s : Stray Capacitance of printed circuit board and connections。
g_{mcrit} : Oscillator loop critical gain。 eq) $g_{mcrit} = 4 \times (ESR + R_{Ext}) \times (2\pi F)^2 \times (C_0 + C_L)^2$
ESR(1) : Maximal equivalent series resistance。 eq) $ESR = R_m X (1 + C_0/C_L)^2$
R_{Ext} : Resistor for limiting the drive level(DL) of the crystal。
DL(1) : The power dissipated in the crystal。 Excess power can destroy the crystal。
R_F(2) : Feedback resistor。

- *C0, CL, ESR and DL are provided by the crystal manufacturer。*
- *The W5100S has no feedback resistor。 Therefore, it must be inserted outside。*

* 无源晶振电路模型如图 23 所示。

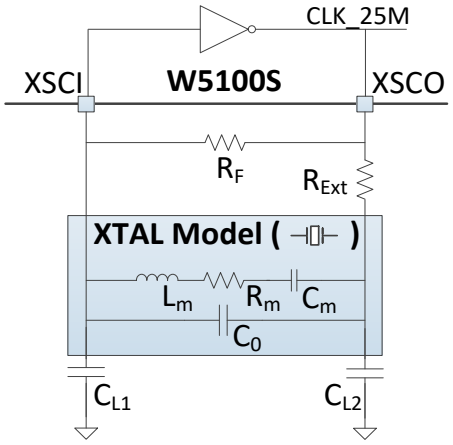


图 23 无源晶振电路模型图

表 9 无源晶振特性

参数	范围
频率	25 MHz
频率误差（25℃时）	±30 ppm
并联电容	最大 7pF
驱动功率	500uW
负载电容	12pF
老化速度（25℃时）	最大±3ppm / 年

6.2 有源晶振特性

表 10 有源晶振特性

参数	条件	最小	典型	最大	单位
频率		25			MHz
频率误差	25°C 时	-50		+50	ppm
频率稳定性	1 年老化，25°C	-50		+50	ppm
Clock duty	50%的波形	45	50	55	%
输入高电压		-	0.97	-	V
输入低电压		-	0.13	-	V
上升/下降时间	从 10% 到 90% 波形			8ns	
启动时间		-	-	10ms	
工作电压		1.08V	1.2V	1.32V	
老化速度（25℃时）		最大±3ppm / 年			ppm

6.3 变压器特性

表 11 变压器特性

参数	发送端	接收端
匝数比	1:1	1:1
电感	350 uH	350 uH

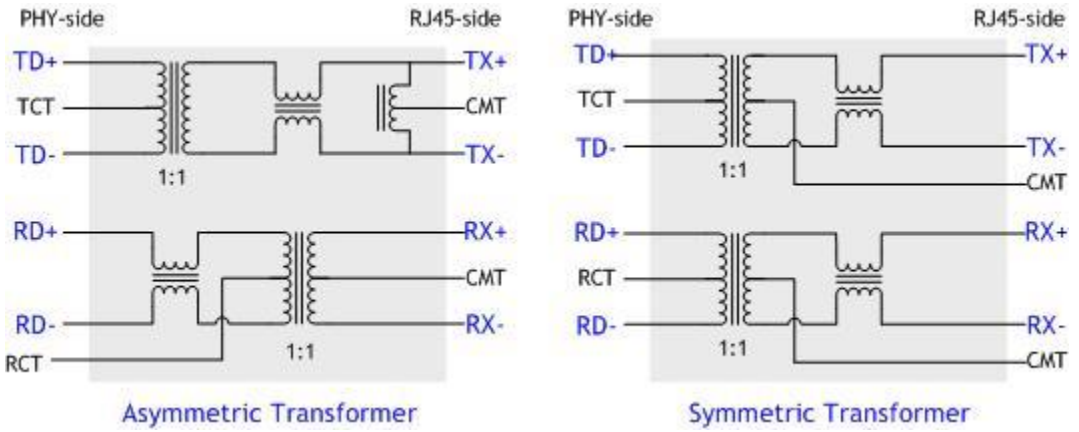


图 24 变压器特性

7 电气规格

7.1 额定值

表 12 额定值

符号	参数	额定范围	单位
V_{DD}	直流电源电压	-0.5 ~ 4.6	V
V_{IN}	直流输入电压	-0.5 ~ 4.6	V
V_{OUT}	直流输出电压	-0.5 ~ 3.63	V
I_{IN}	直流输入电流	20	mA
T_{OP}	工作温度	-40 ~ +85	°C
T_{STG}	储存温度	-65 ~ +150	°C

***注**: 超过额定值的操作可能会引起元件永久性损坏*

7.2 极限值（ESD）

表 13 极限值 (ESD)

符号	参数	测试环境	类	最大值 (1)	单位
V_{ESD} HBM	静电放电电压（人体模型)	TA = +25 °C 时，符合 MIL-STD 883F Method 3015.7	2	2000	V
V_{ESD} MM	静电放电电压（人机模型)	TA = +25 °C 时，符合 JEDEC EIA/JESD22 A115-A	B	200	V
V_{ESD} CDM	静电放电电压（充电装置模型)	TA = +25 °C 时，符合 JEDEC JESD22 C101-C	III	500	V

表 14 闩锁测试

测试环境	类	最大值	单位
TA = +25 °C 时， 符合 JESD78	Current	≥ ±100	mA
	Voltage	≥ 1.5*V_{DD}	V

7.3 DC 特性

表 15 直流特性

(测试环境: Ta = –40 ~ 85°C)

符号	参数	测试环境	最小	典型	最大	单位
V_{DD}	电源电压	供应 VDD, AVDD	2.97	3.3	3.63	V
V_{IH}	高电平输入电压		2.0	-	-	V
V_{IL}	低电平输入电压		-		0.8	V
V_{T+}	施密特触发器从低到高阈值点	除模拟管脚之外的所有输入	0.8	1.1	-	V
V_{T-}	施密特触发器从高到低阈值点	除模拟管脚之外的所有输入	-	1.6	2.0	V
T_J	结温		-40	25	125	°C
I_L	输入漏电流			±1	±10	μA
R_{PU}	上拉电阻		40	75	190	Kohm
R_{PD}	下拉电阻	RSVD (管脚 23, 38 ~ 42)	40	75	190	Kohm
V_{OL}	低电平输出电压	IOL = 4.0mA~16mA, 除 XO 以外的所有输出			0.4	V
V_{OH}	高电平输出电压	IOH = 4.0mA~6mA, 除 XO 以外的所有输出	2.4			V
I_{DD1}	电源电流（正常操作模式）	VDD=3.3V, AVDD=3.3V, Ta = 25°C		132		mA
I_{DD2}	电源电流（掉电模式）	PHY 掉电模式, VDD=3.3V, AVDD=3.3V, Ta = 25°C		13		mA

7.4 AC 特性

7.4.1 复位时序

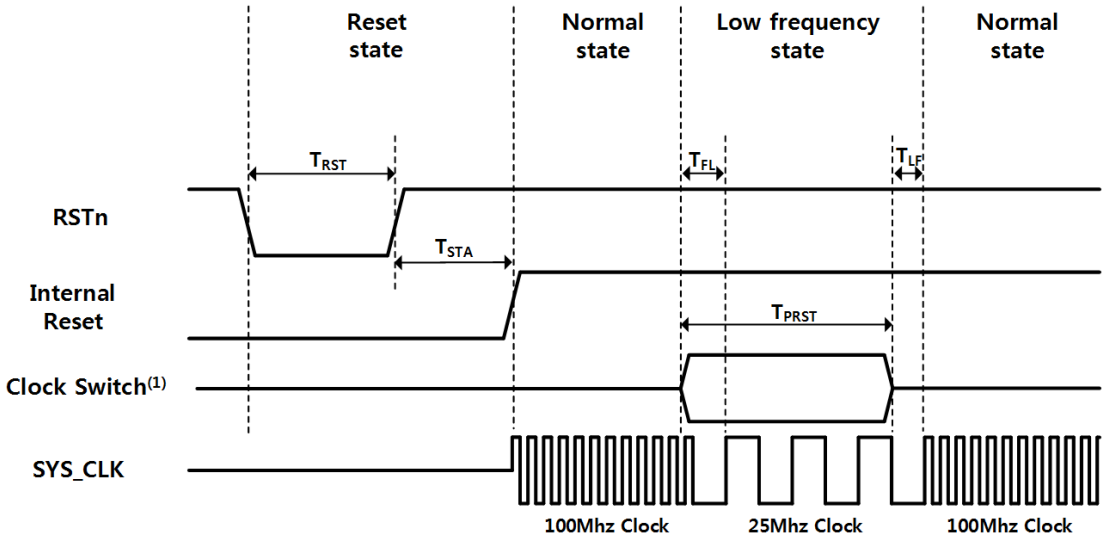


图 25 复位时序

表 16 复位时序表

符号	描述	最小	典型	最大
T_{RST}	复位时间	210 ns	330 ns	560 ns-
T_{STA}	稳定时间	-		60.3 ms
T_{FL}	时钟从高频率到低频率的时间（由 MR2[CLKSEL]触发）	100 ns		-
T_{FL}	时钟从高频率到低频率的时间（由 [Reset]或[PWDN]触发）	300 ns		
T_{PRST}	PHY 自动复位时间	0.6 ms		-
T_{PRST}	PHY 掉电时间	200 us		
T_{PRST}	时钟切换时间	200 ns		
T_{LF}	时钟从低频率到高频率的时间（由 MR2[CLKSEL]触发）	100 ns		-
T_{LF}	时钟从低频率到高频率的时间（由 [Reset]或[PWDN]触发）	100 ns		

注:PHY 掉电模式有 TFI 和 TLF（PHY 掉电模式），SYS_CLK 切换到低时钟。在 TFI 之后，用户可以禁用 PHY 掉电模式。

警告：用户不能同时设置 PHY 自动复位和 PHY 电源关闭模式

7.4.2 总线访问时序

7.4.2.1 总线读取时序

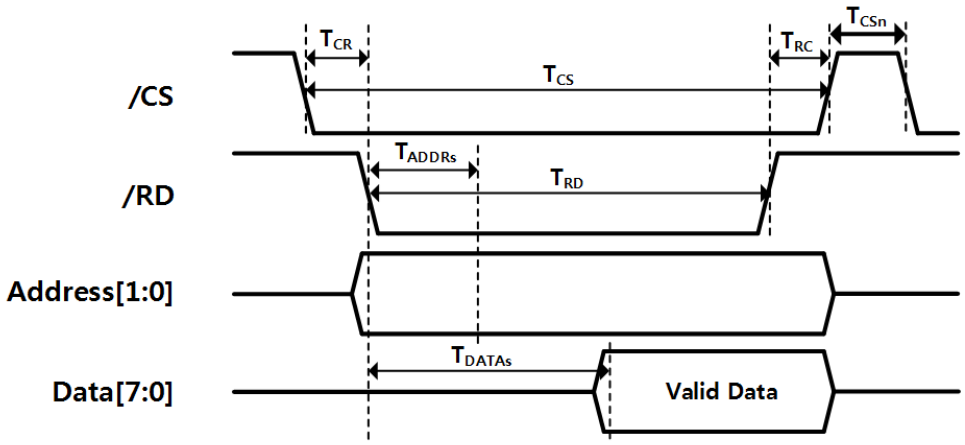


图 26 总线读取时序

表 17 总线读取时序

符号	描述	最小	最大
T_{ADDRs}	地址建立时间	SYS_CLK	
T_{CR}	/CS 拉低到 /RD 拉低的间隔	0 ns	
T_{cs}	/CS 拉低的时间	4 SYS_CLK	
T_{RC}	/RD 拉高到 /CS 拉高的间隔	0 ns	
T_{csn}	/CS 拉高到拉低的间隔	2 SYS_CLK	
T_{RD}	/RD 拉低的时间	4 SYS_CLK	
T_{DATAs}	数据建立时间		3 SYS_CLK+5ns

7.4.2.2 总线写入时序

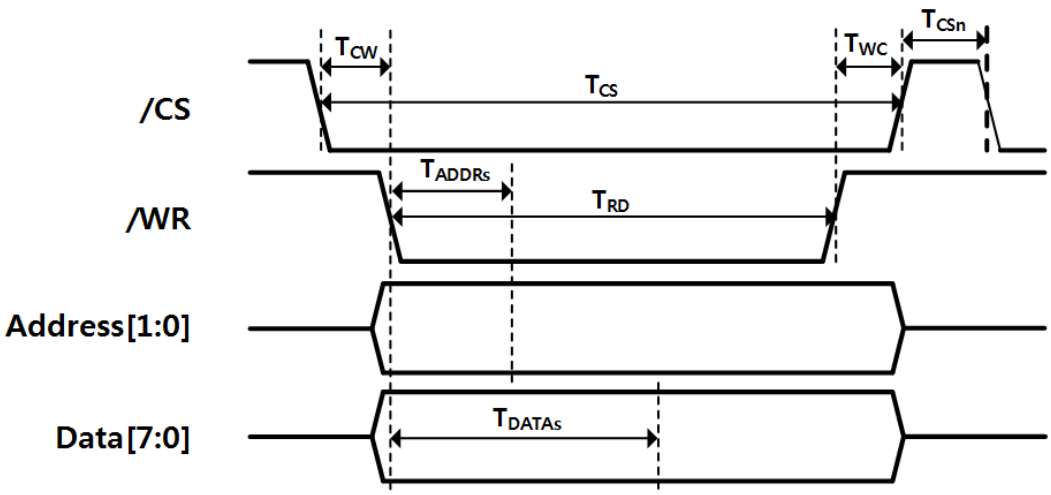


图 27 总线写入时序

表 18 总线写入时序

符号	描述	最小	最大
T_{ADDRs}	地址建立时间	SYS_CLK	
T_{CW}	/CS 拉低到 /WR 拉低的间隔	0 ns	
T_{cs}	/CS 拉低的时间	4 SYS_CLK	
T_{WC}	/WR 拉高到 /CS 拉高的间隔	0 ns	
T_{csn}	/CS 拉高到拉低的间隔	2 SYS_CLK	
T_{WR}	/WR 拉低的时间	4 SYS_CLK	
T_{DATAs}	数据建立时间	2 SYS_CLK	

7.4.3 SPI 访问时序

7.4.3.1 SPI 读取时序

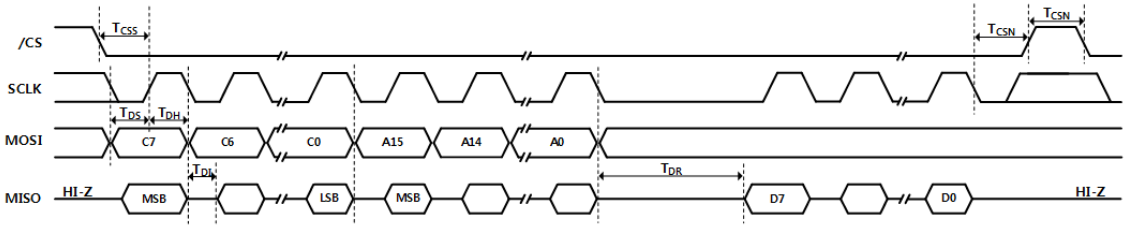


图 28 SPI 读取时序

表 19 SPI 读取时序

符号	描述	最小	最大	单位 s
F_{SCK}	SCK Clock Frequency		70	MHz
T_{CSS}	SCSn Setup Time	3 SYS_CLK		ns
T_{CSN}	SCSn Next Time	2 SYS_CLK		ns
T_{DS}	Data In Setup Time	3		ns
T_{DH}	Data In Hold Time	3		ns
T_{DI}	Data Invalid Time	7		ns
T_{DR}	Data Ready Time	6 SYS_CLK + 30		ns

7.4.3.2 SPI 写入时序

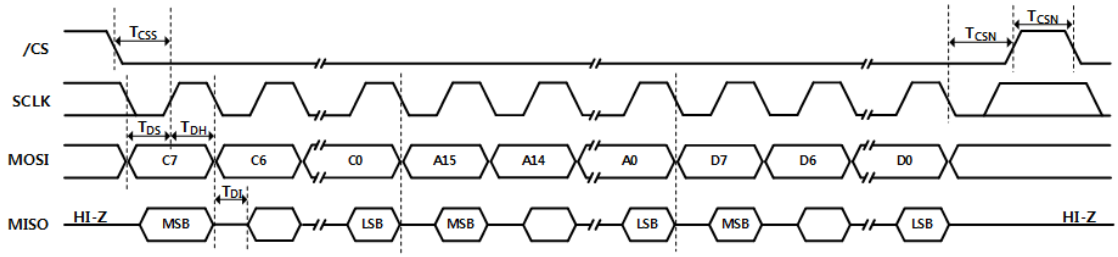


图 29 SPI 写入时序

表 20 SPI 写入时序

符号	描述	最小	最大	单位 s
F_{SCK}	SCK Clock Frequency		70	MHz
T_{CSS}	SCSn Setup Time	3 SYS_CLK		ns
T_{CSN}	SCSn Next Time	2 SYS_CLK		ns
T_{DS}	Data In Setup Time	3		ns
T_{DH}	Data In Hold Time	3		ns
T_{DI}	Data Invalid Time	7		ns

7.4.4 变压器特性

表 21 变压器特性

参数	发送端	接收端
匝数比	1:1	1:1
电感	350 uH	350 uH

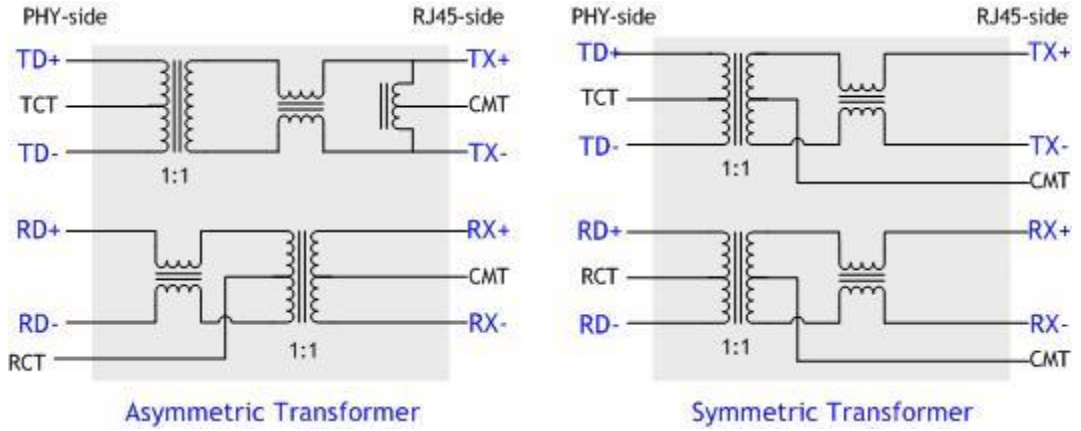


图 30 变压器类型

7.4.5 MDIX

W5100S 只在自动协商模式下支持 Auto-MDIX 功能。

7.5 功耗

表 22 功耗

(测试环境: VDD=3.3V, AVDD=3.3V, Ta = 25°C)

状态	最小	典型	最大	单位
100M Link	-	93	110	mA
10M Link	-	150	170	mA
100M Unlink(Actual Measurement)	-	45		mA
10M Unlink(Actual Measurement)	-	17		mA
Un-Link (Auto-negotiation mode) (Actual Measurement)	-	43	-	mA
Power Down mode	-	17	-	mA

8 封装

8.1 LQFP48

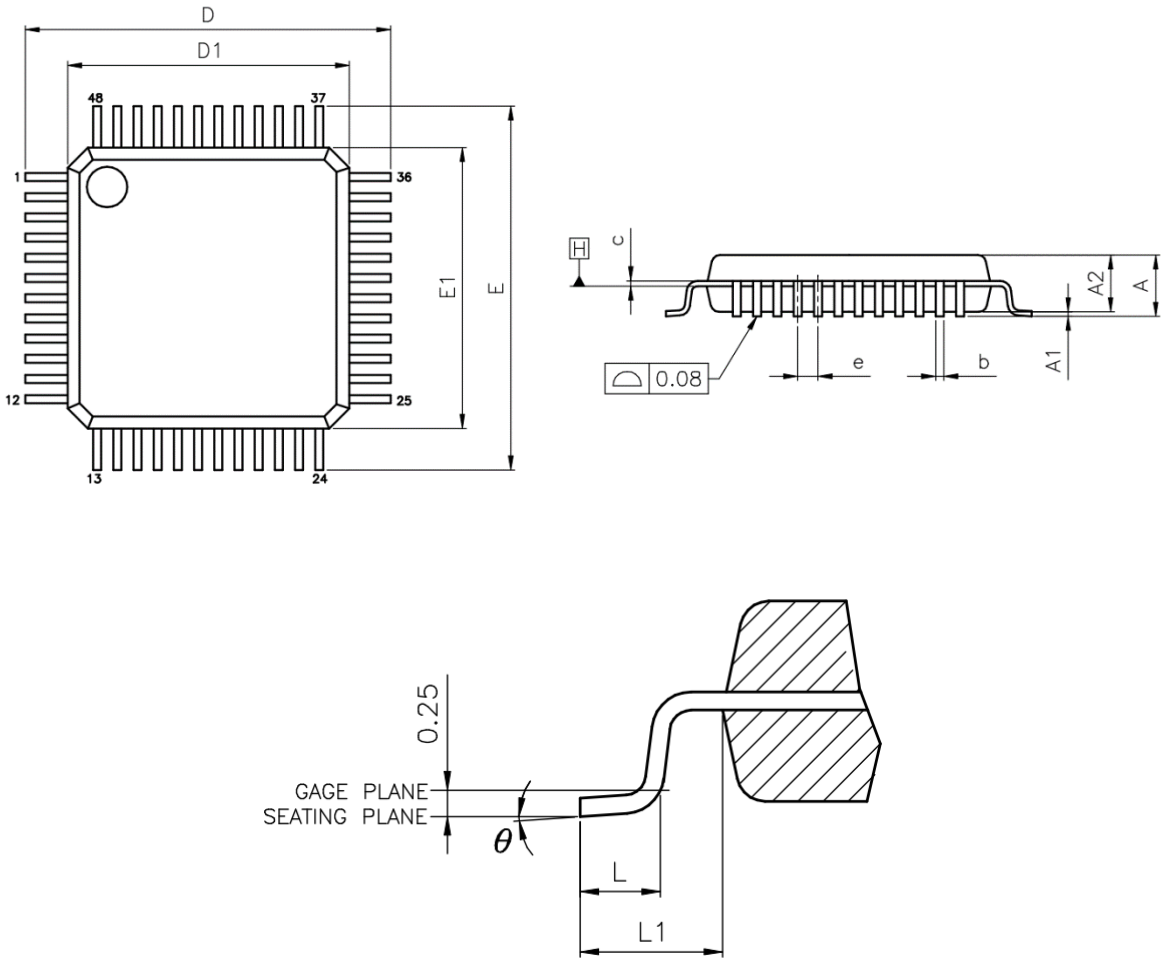


表 23 LQFP48 (所有的尺寸均以毫米：mm 表示)

符号	最小	正常	最大
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.22	0.27
c	0.09	--	0.20
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.50 BSC		

L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°

NOTES:

1. JEDEC OUTLINE:
   MS-026 BBC
   MS-026 BBC-HD (THERMALLY ENHANCED VARIATIONS ONLY)
2. DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY
3. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION。 ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE。 DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H。
4. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION。

8.2 QFN48

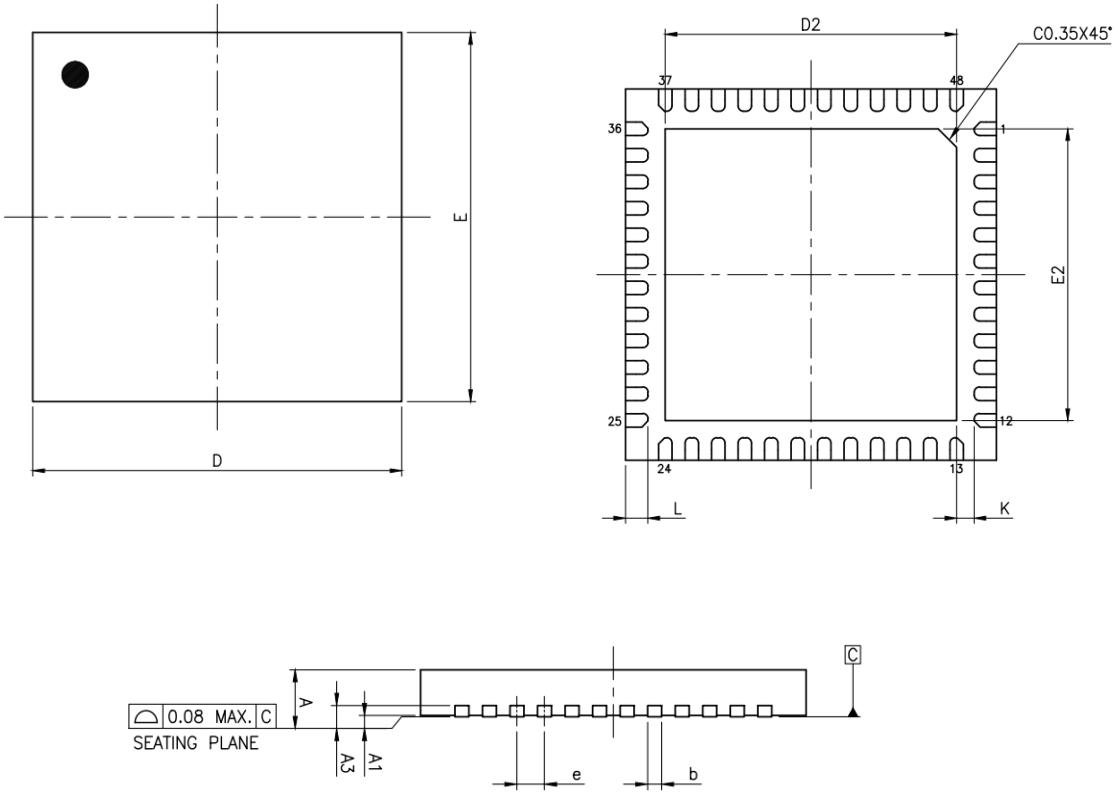


表 24 QFN48 封装(所有的尺寸均以毫米：mm 表示)

符号	最小	典型	最大
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF		
b	0.20	0.25	0.30
D	7.00 BSC		
E	7.00 BSC		
e	0.50 BSC		
D2	5.25	5.30	5.35
E2	5.25	5.30	5.35
L	0.35	0.40	0.45
K	0.20	--	--
LEAD FINISH	Pure Tin	V	PPF	X
JEDEC CODE	N/A		

NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS
2. DEMENSION B APPLIES TO METALLIZED TERMINALS AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINALS TIP。 IF THE TERMINALS HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINALS, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA。
3. BILATERAL COPLANARITY ZONE APPLIES TO THE EXPOSED HEAT SINK SLUG AS WELL AS THE TERMINALS。

9 文档修订历史

版本	日期	描述
Ver. 1.0	2018-04-01	初始版本
Ver. 1.1	2019-05-21	修改文档描述