W6100

全硬件双核 TCP/IP 协议栈控制器

V1.0.0

浩然电子 Hasion Electronics

https://www.hschip.com

W6100 概述

W6100 是全球第一款支持 IPv4/IPv6 双核的新一代全硬件以太网 TCP/IP 协议栈控制器。W6100 在 WIZnet 核心专利技术——全硬件 TCP/IP 协议栈 IPv4 的基础上增加了 IPv6，并且支持 TCP，UDP，IPv6，IPv4，ICMPv6，ICMPv4，IGMP，ARP 以及 PPPoE 等协议。同时其内部集成了 10/100M 以太网数据链路层（MAC）以及物理层（PHY），使用户能够更加简单快速地实现嵌入式设备的联网功能。

W6100 具有 8 个独立的硬件 SOCKET，并且支持多种 SOCKET-less 命令，这些命令通过 ARP、PINGv4 及 PINGv6 来实现 IPv6 自动配置、监查和管理网络。

W6100 具有 SPI 和并行总线两种主机接口，并自带 32KB 的高速数据缓存用于发送和接收数据。同时为了满足日益增加的低功耗需求，W6100 提供了网络唤醒（WOL）和以太网 PHY 掉电模式，方便用户在不同的应用中使用。

此外，W6100 提供 LQFP48 和 QFN48 两种无铅封装，并且与 W5100S PIN-2-PIN 兼容。

特点

- ✓ 支持 IPv4/IPv6 双协议栈
- ✓ 支持多种 TCP/IP 协议
  - TCP、UDP、IPv6、IPv4、ICMPv6、ICMPv4、IGMP、MLDv1、ARP、PPPoE
- ✓ 支持 8 个独立 SOCKET，同时具有 32K 的收发缓存
- ✓ 支持多种 SOCKET-less 命令
  - ARP4、ARP6、PING4、PING6、NS（DAD）、RS、UNA
- ✓ 支持 PHY 掉电和关闭系统时钟两种节能模式
- ✓ 支持基于 UDP 协议的网络唤醒（WOL）功能
- ✓ 支持串行和并行主机接口
  - 高速 SPI（MODE 0/3）
  - 系统总线（2 位地址总线和 8 位数据总线）
- ✓ 具有内部 32K 的收发缓存
- ✓ 集成了 10BaseT/10BaseTe/100BaseTX 的以太网 PHY
- ✓ 支持自动协商（全/半双工、10/100M）
- ✓ 支持 auto-MDIX 自动极性转换（仅在自动协商模式下）
- ✓ 不支持 IP 分片和巨型帧
- ✓ 工作电压：3.3V（I/O 耐压 5V）
- ✓ 网络指示灯引脚（全/半双工、Link、10/100M、Act）
- ✓ 48 管脚 LQFP 和 QFN 两种无铅封装(7x7mm，0.5mm 间距)
- ✓ 与 W5100S PIN-2-PIN 兼容

应用

- ✓ 智能家居网关
- ✓ 串口转以太网
- ✓ 并口转以太网：POS 机/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
- ✓ 工厂和楼宇自动化
- ✓ 医用检测设备
- ✓ 嵌入式服务器
- ✓ 物联网（IoT）设备

W6100 内部架构图

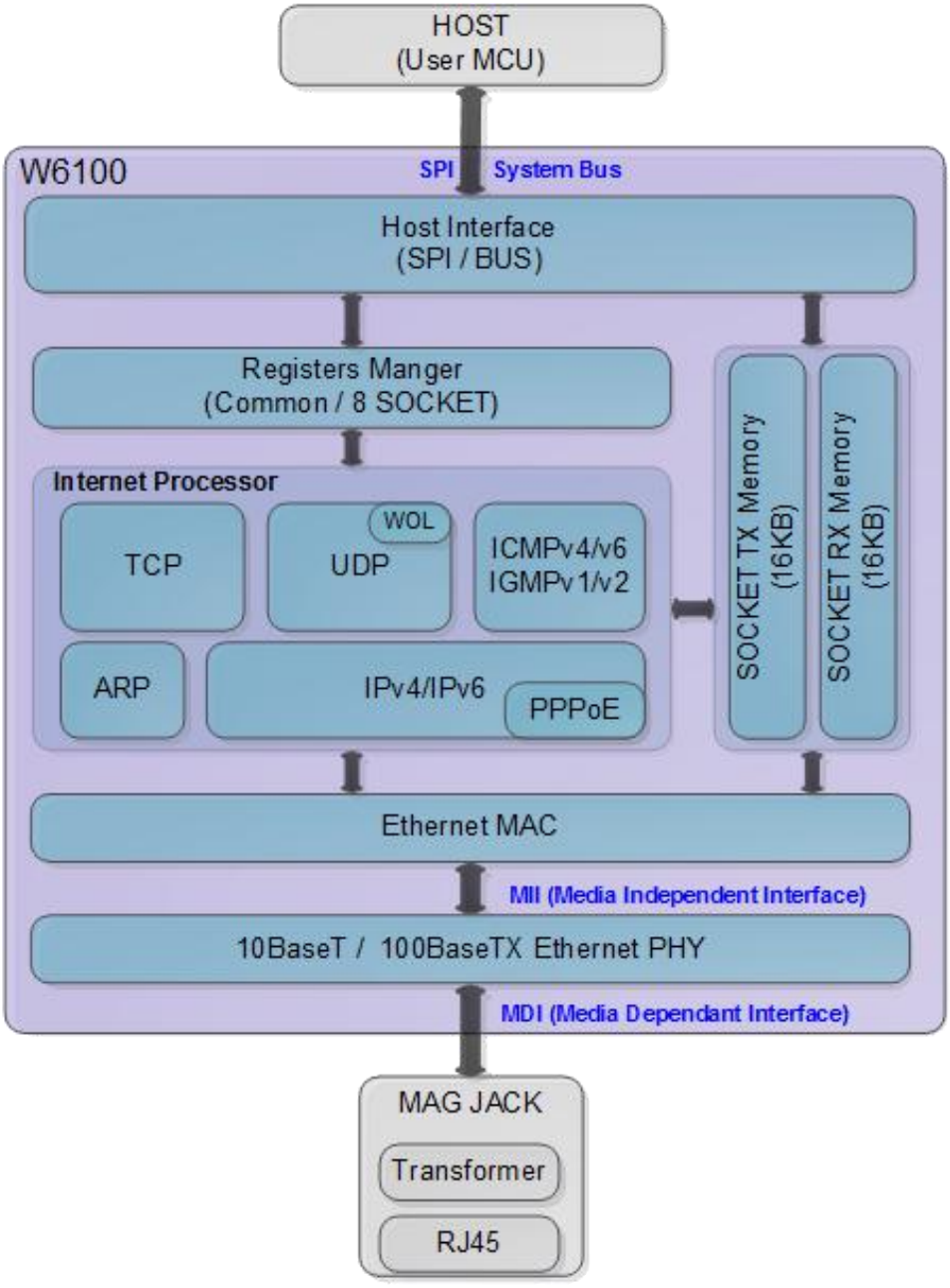


图 1 W6100 内部架构框图

目 录

1 引脚描述

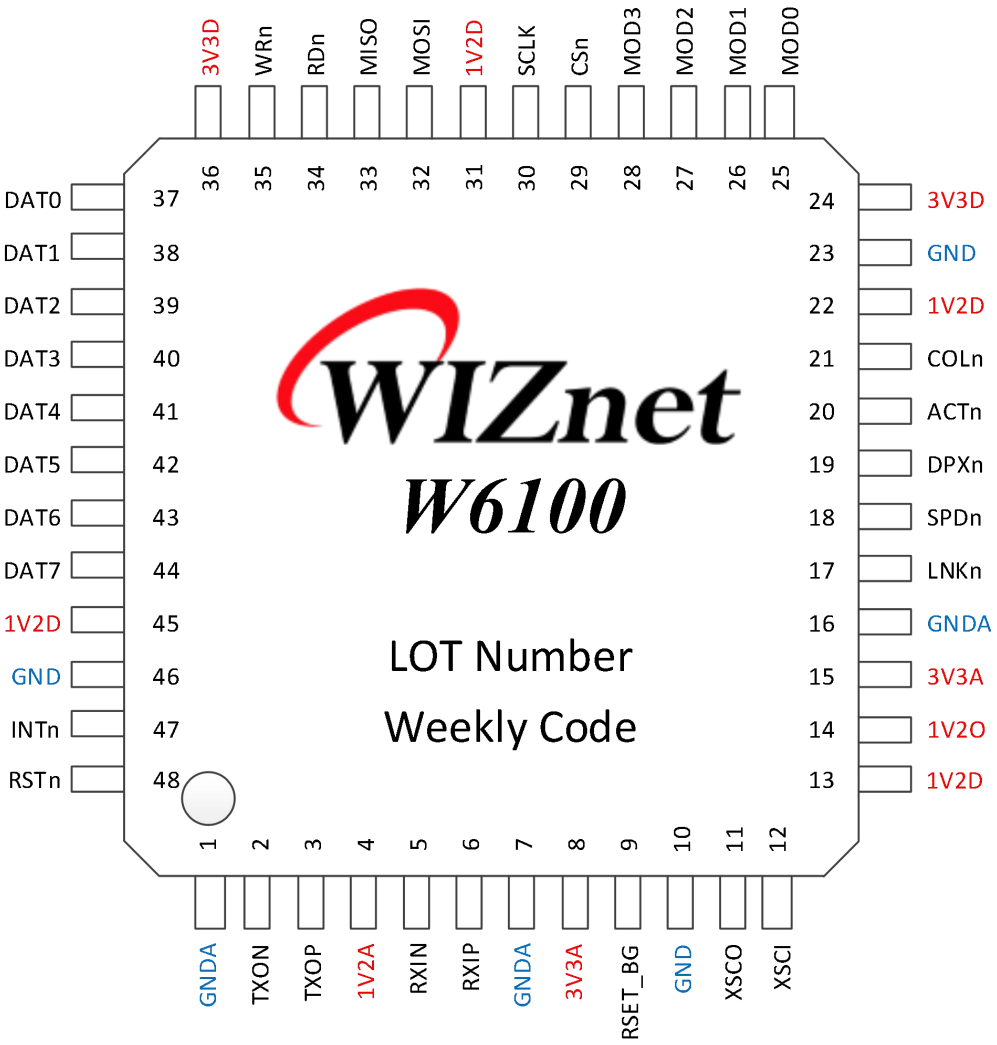


图 2 W6100 引脚分布图

表 1 引脚类型表

引脚类型	引脚描述
I	输入
O	输出
M	保留信号/多功能信号
U	内部上拉 75KΩ电阻
D	内部下拉 75KΩ电阻
A	模拟信号
P	电源和地

1.1 引脚描述

表 2 引脚描述

引脚	符号	类型	描述
1	GNDA	AP	模拟地
2	TXON	AO	TXON/ TXOP 差分发送信号对 在 MDI 模式下，通过 TXOP/TXON 差分信号对发送差分数据信号
3	TXOP	AO	
4	1V2A	AP	1.2V 模拟电源输入 必须由 Pin14 引脚输出的 1V2O 转为 1V2A 后输入
5	RXIN	AI	TXON/ TXOP 差分接收信号对 在 MDI 模式下，通过 RXIP/RXIN 差分信号对接收差分数据信号
6	RXIP	AI	
7	GNDA	AP	模拟地
8	3V3A	AP	3.3V 模拟电源输入
9	RSET_BG	AO	外接下拉电阻 必须外接 12.3KΩ、精度 1%的电阻接地
10	GND	AP	数字地
11	XSCO	AO	25MHz 时钟 外接 25MHz 无源晶振（XTAL）或有源晶振（OSC）作为内部工作时钟，W6100 可将其转换为 25MHz 或 100MHz 作为 SYS_CLK 使用。 正常模式下，SYS_CLK 为 100MHz； 低频模式下，SYS_CLK 为 25MHz 如果使用有源晶振，则必须使用 25MHz/1.2V，XSCI 必须接地，且 XSCO 脚必须悬空 参考) “时钟选择指南” http://wizwiki.net.
12	XSCI	AI	
13	1V2D	P	1.2V 数字电源输入 该引脚需要输入 1.2V 的数字电源
14	1V2O	PO	内部稳压器输出 1.2V 电压 W6100 内部稳压器通过该引脚输出 1.2V 电压及最大 150mA 的电流 通过外接 3.3uF 电容以确保其稳定输出 1V2D 和 1V2A。 当输出 1V2O 时，必须使用磁珠将 1V2D 和 1V2A 隔离 * 注意：此电源输出仅可用于 *W6100*，不能用于其他设备供电
15	3V3A	AP	3.3V 模拟电源输入
16	GNDA	AP	模拟地
17	LNKn	OU	PHY 连接 LED 指示 适用于 SPI 和总线模式。 低电平: PHY 已连接

			高电平: PHY 未连接
18	SPDn	OU	网络速度 LED 指示 适用于 SPI 和总线模式。 低电平: 100Mbps 高电平: 10Mbps
19	DPXn	OU	全/半双工 LED 指示 适用于 SPI 和总线模式。 低电平: 全双工状态 高电平: 半双工状态
20	ACTn	OU	活动状态 LED 指示 适用于 SPI 和总线模式。 低电平: 已连接无数据收发 高低电平变化: 已连接且有数据收发 高电平: 未连接
21	COLn	OU	连接冲突检测 LED 指示 适用于 SPI 和总线模式。 表示数据在传输过程中发生连接冲突。 低电平: 检测到冲突 高电平: 无冲突
22	1V2D	P	1.2V 数字电源输入 该引脚需要输入 1.2V 的数字电源
23	GND	P	数字地
24	3V3D	P	3.3V 数字电源输入
25	MOD[0]	ID	W6100 接口模式选择 接口模式由 MOD [3:0]确定: “000X”: SPI 模式 “010X”: 总线模式 Others: 保留
26	MOD[1]	ID	
27	MOD[2]	ID	
28	MOD[3]	ID	
29	CSn	IU	W6100 片选 低电平: 使能 高电平: 失能
30	SCLK	ID	SPI 时钟 在 SPI 模式时作为 SPI 的时钟输入，在总线模式下必须接地或悬空
31	1V2D	P	1.2V 数字电源输入 该引脚需要输入 1.2V 的数字电源
32	MOSI /ADDR0	IDM	SPI 接口的 MOSI 引脚 / ADDR0 MOSI: 在 SPI 模式下作为 MOSI 引脚 ADDR0: 在总线模式下作为 ADDR0
33	MISO	IOPM	SPI 接口的 MISO 引脚 / ADDR1

	/ADDR1		MISO：在 SPI 模式下作为 MISO 引脚 ADDR1：在总线模式下作为 ADDR1
34	RDn	IU	读使能 在并行总线模式下指示读操作 在 SPI 模式下，必须接 3V3D 或者悬空
35	WRn	IU	写使能 在并行总线模式下指示写操作 在 SPI 模式下，必须接 3V3D 或者悬空
36	3V3D	P	3.3V 数字电源
37	DAT0	IOU	8 位数据总线 在总线模式下，主机和 W6100 之间通过 DAT[7:0]传输数据； 在 SPI 模式下，DAT[7:0]必须悬空
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 数字电源输入 该引脚需要输入 1.2V 的数字电源
46	GND	P	数字地
47	INTn	OP	中断引脚 当以太网数据通信期间发生中断事件时，INTn 将被触发。 低电平：触发中断 高电平：无中断 请参考 SYCR1（系统配置寄存器 1）中的 IEN（中断使能），INTPTMR（中断挂起时间寄存器），IR（中断寄存器），SIR（SOCKET 中断寄存器），SLIR（SOCKET-less 中断寄存器）。
48	RSTn	IP	复位引脚 W6100 通过 RSTn 引脚的复位来初始化。 RSTn 需要拉低不小于 1.0us 才能复位，复位后 W6100 需要 60.3ms 来完成初始化过程 *请参考* 8.4.1 复位时序

2 内存映射

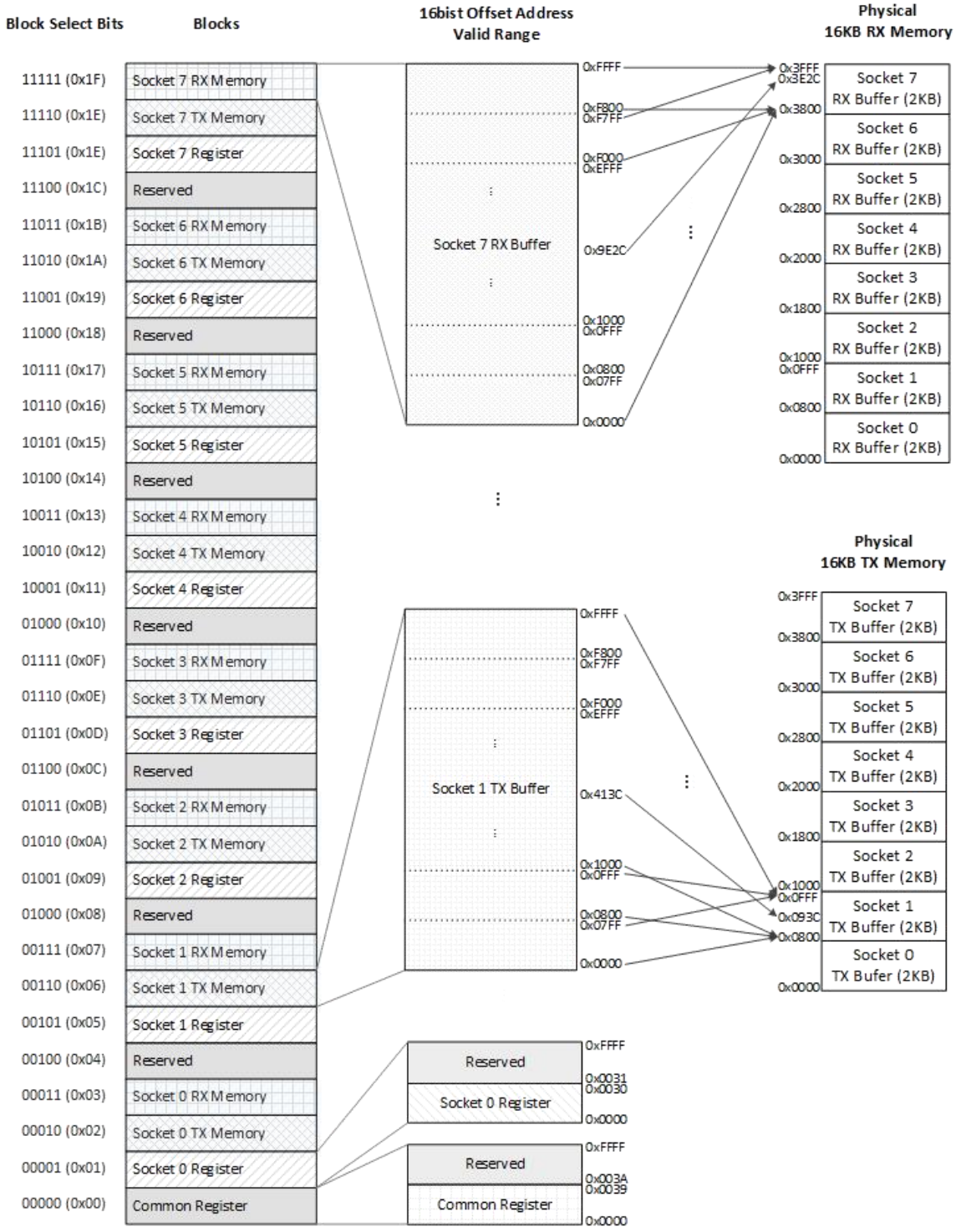


图 3 W6100 内存映射

如图 3 中，W6100 内存由以下几个区块组成：

- 1 个通用寄存器区块
- 7 个保留区块
- 8 个 SOCKET n 寄存器区块
- 8 个 SOCKET n 接收缓存区块
- 8 个 SOCKET n 发送缓存区块

用户可以通过 5 位选择位来选取要访问的储存器块，并使用 16 位偏移地址访问每个内存区块。

所有 SOCKET 发送缓存一共 16KB，每个 SOCKET 的发送缓存大小默认为 2KB。用户可以通过 Sn_TX_BSR 寄存器(4.2.23)重新分配 SOCKET 的发送缓存大小，每一个 SOCKET 的发送缓存均可配置为 0、1、2、4、8 或者 16KB，但是全部 SOCKET 的发送缓存大小之和不得超过 16KB。

所有 SOCKET 接收缓存一共 16KB，每个 SOCKET 的接收缓存大小默认为 2KB。用户可以通过 Sn_RX_BSR 寄存器(4.2.27)重新分配 SOCKET 的接收缓存大小，每一个 SOCKET 的接收缓存均可配置为 0、1、2、4、8 或者 16KB，但是全部 SOCKET 的接收缓存大小之和不得超过 16KB。

3 寄存器列表

3.1 通用寄存器

偏移地址	寄存器	类型[1]	复位值
0x0000	CIDR0（芯片 ID 寄存器 0）	RO	0x61
0x0001	CIDR1（芯片 ID 寄存器 1）	RO	0x00
0x0002	VER0（版本寄存器 0）	RO	0x46
0x0003	VER1（版本寄存器 1）	RO	0x61
0x2000	SYSR（系统状态寄存器）	RO	0xEU
0x2004	SYCR0（系统配置寄存器 0）	WO	0x80
0x2005	SYCR1（系统配置寄存器 1）	R=W	0x80
0x2016	TCNTR0（滴答计数器寄存器 0）	RO	0x00
0x2017	TCNTR1（滴答计数器寄存器 1）	RO	0x00
0x2020	TCNTCLR（滴答计数器清楚寄存器）	WO	0x00
0x2100	IR（中断寄存器）	RO	0x00
0x2101	SIR（SOCKET 中断寄存器）	RO	0x00
0x2102	SLIR（SOCKET-less 中断寄存器）	RO	0x00
0x2104	IMR（中断屏蔽寄存器）	R=W	0x00
0x2108	IRCLR（中断清除寄存器）	WO	0x00
0x2114	SIMR（SOCKE 中断屏蔽寄存器）	R=W	0x00
0x2124	SLIMR（SOCKET-less 中断屏蔽寄存器）	R=W	0x00
0x2128	SLIRCLR（SOCKET-less 中断清除寄存器）	WO	0x00
0x212C	SLPSR（SOCKET-less 首选源 IPv6 地址寄存器）	R=W	0x00
0x2130	SLCR（SOCKET-less 控制寄存器）	RW,AC	0x00
0x3000	PHYSR（PHY 状态寄存器）	RO	0x00
0x3008	PHYRAR（PHY 寄存器地址寄存器）	R=W	0x00
0x300C	PHYDIR0（PHY 数据输入寄存器 0）	R=W	0x00
0x300D	PHYDIR1（PHY 数据输入寄存器 1）	R=W	0x00
0x3010	PHYDOR0（PHY 数据输出寄存器 0）	RO	0x00
0x3011	PHYDOR1（PHY 数据输出寄存器 1）	RO	0x00
0x3014	PHYACR（PHY 访问控制寄存器）	RW,AC	0x00
0x3018	PHYDIVR（PHY 分频寄存器）	R=W	0x01
0x301C	PHYCR0（PHY 控制寄存器 0）	WO	0x00
0x301D	PHYCR1（PHY 控制寄存器 1）	R=W	0x40
0x4000	NET4MR（IPv4 网络模式寄存器）	R=W	0x00

[1] 参考) 4. 通用寄存器

0x4004	NET6MR（IPv6 网络模式寄存器）	R=W	0x00
0x4008	NETMR（网络模式寄存器）	R=W	0x00
0x4009	NETMR2 （网络模式寄存器 2）	R=W	0x00
0x4100	PTMR（PPP 链路控制协议请求定时器寄存器）	R=W	0x28
0x4104	PMNR（PPP 链路控制协议 Magic Number 寄存器）	R=W	0x00
0x4108	PHAR0（PPPoE 模式下 MAC 地址寄存器 0）	R=W	0x00
0x4109	PHAR1（PPPoE 模式下 MAC 地址寄存器 1）	R=W	0x00
0x410A	PHAR2（PPPoE 模式下 MAC 地址寄存器 2）	R=W	0x00
0x410B	PHAR3（PPPoE 模式下 MAC 地址寄存器 3）	R=W	0x00
0x410C	PHAR4（PPPoE 模式下 MAC 地址寄存器 4）	R=W	0x00
0x410D	PHAR5（PPPoE 模式下 MAC 地址寄存器 5）	R=W	0x00
0x4110	PSIDR0（PPPoE 会话 ID 寄存器 0）	R=W	0x00
0x4111	PSIDR1（PPPoE 会话 ID 寄存器 1）	R=W	0x00
0x4114	PMRUR0（PPPoE 最大接收单元寄存器 0）	R=W	0xFF
0x4115	PMRUR1（PPPoE 最大接收单元寄存器 0）	R=W	0xFF
0x4120	SHAR0（源 MAC 地址寄存器 0）	R=W	0x00
0x4121	SHAR1（源 MAC 地址寄存器 1）	R=W	0x00
0x4122	SHAR2（源 MAC 地址寄存器 2）	R=W	0x00
0x4123	SHAR3（源 MAC 地址寄存器 3）	R=W	0x00
0x4124	SHAR4（源 MAC 地址寄存器 4）	R=W	0x00
0x4125	SHAR5（源 MAC 地址寄存器 5）	R=W	0x00
0x4130	GAR0（网关 IP 地址寄存器 0）	R=W	0x00
0x4131	GAR1（网关 IP 地址寄存器 1）	R=W	0x00
0x4132	GAR2（网关 IP 地址寄存器 2）	R=W	0x00
0x4133	GAR3（网关 IP 地址寄存器 3）	R=W	0x00
0x4134	SUBR0（子网掩码寄存器 0）	R=W	0x00
0x4135	SUBR1（子网掩码寄存器 1）	R=W	0x00
0x4136	SUBR2（子网掩码寄存器 2）	R=W	0x00
0x4137	SUBR3（子网掩码寄存器 3）	R=W	0x00
0x4138	SIPR0（IPv4 源 IP 地址寄存器 0）	R=W	0x00
0x4139	SIPR1（IPv4 源 IP 地址寄存器 1）	R=W	0x00
0x413A	SIPR2（IPv4 源 IP 地址寄存器 2）	R=W	0x00
0x413B	SIPR3（IPv4 源 IP 地址寄存器 3）	R=W	0x00
0x4140	LLAR0（Link 本地地址寄存器 0）	R=W	0x00
0x4141	LLAR1（Link 本地地址寄存器 1）	R=W	0x00
0x4142	LLAR2（Link 本地地址寄存器 2）	R=W	0x00
0x4143	LLAR3（Link 本地地址寄存器 3）	R=W	0x00
0x4144	LLAR4（Link 本地地址寄存器 4）	R=W	0x00

0x4145	LLAR5（Link 本地地址寄存器 5）	R=W	0x00
0x4146	LLAR6（Link 本地地址寄存器 6）	R=W	0x00
0x4147	LLAR7（Link 本地地址寄存器 7）	R=W	0x00
0x4148	LLAR8（Link 本地地址寄存器 8）	R=W	0x00
0x4149	LLAR9（Link 本地地址寄存器 9）	R=W	0x00
0x414A	LLAR10（Link 本地地址寄存器 10）	R=W	0x00
0x414B	LLAR11（Link 本地地址寄存器 11）	R=W	0x00
0x414C	LLAR12（Link 本地地址寄存器 12）	R=W	0x00
0x414D	LLAR13（Link 本地地址寄存器 13）	R=W	0x00
0x414E	LLAR14（Link 本地地址寄存器 14）	R=W	0x00
0x414F	LLAR15（Link 本地地址寄存器 15）	R=W	0x00
0x4150	GUAR0（全局单播地址寄存器 0）	R=W	0x00
0x4151	GUAR1（全局单播地址寄存器 1）	R=W	0x00
0x4152	GUAR2（全局单播地址寄存器 2）	R=W	0x00
0x4153	GUAR3（全局单播地址寄存器 3）	R=W	0x00
0x4154	GUAR4（全局单播地址寄存器 4）	R=W	0x00
0x4155	GUAR5（全局单播地址寄存器 5）	R=W	0x00
0x4156	GUAR6（全局单播地址寄存器 6）	R=W	0x00
0x4157	GUAR7（全局单播地址寄存器 7）	R=W	0x00
0x4158	GUAR8（全局单播地址寄存器 8）	R=W	0x00
0x4159	GUAR9（全局单播地址寄存器 9）	R=W	0x00
0x415A	GUAR10（全局单播地址寄存器 10）	R=W	0x00
0x415B	GUAR11（全局单播地址寄存器 11）	R=W	0x00
0x415C	GUAR12（全局单播地址寄存器 12）	R=W	0x00
0x415D	GUAR13（全局单播地址寄存器 13）	R=W	0x00
0x415E	GUAR14（全局单播地址寄存器 14）	R=W	0x00
0x415F	GUAR15（全局单播地址寄存器 15）	R=W	0x00
0x4160	SUB6R0（IPv6 子网前缀寄存器 0）	R=W	0x00
0x4161	SUB6R1（IPv6 子网前缀寄存器 1）	R=W	0x00
0x4162	SUB6R2（IPv6 子网前缀寄存器 2）	R=W	0x00
0x4163	SUB6R3（IPv6 子网前缀寄存器 3）	R=W	0x00
0x4164	SUB6R4（IPv6 子网前缀寄存器 4）	R=W	0x00
0x4165	SUB6R5（IPv6 子网前缀寄存器 5）	R=W	0x00
0x4166	SUB6R6（IPv6 子网前缀寄存器 6）	R=W	0x00
0x4167	SUB6R7（IPv6 子网前缀寄存器 7）	R=W	0x00
0x4168	SUB6R8（IPv6 子网前缀寄存器 8）	R=W	0x00
0x4169	SUB6R9（IPv6 子网前缀寄存器 9）	R=W	0x00
0x416A	SUB6R10（IPv6 子网前缀寄存器 10）	R=W	0x00

0x416B	SUB6R11（IPv6 子网前缀寄存器 11）	R=W	0x00
0x416C	SUB6R12（IPv6 子网前缀寄存器 12）	R=W	0x00
0x416D	SUB6R13（IPv6 子网前缀寄存器 13）	R=W	0x00
0x416E	SUB6R14（IPv6 子网前缀寄存器 14）	R=W	0x00
0x416F	SUB6R15（IPv6 子网前缀寄存器 15）	R=W	0x00
0x4170	GA6R0（IPv6 网关 IP 地址寄存器 0）	R=W	0x00
0x4171	GA6R1（IPv6 网关 IP 地址寄存器 1）	R=W	0x00
0x4172	GA6R2（IPv6 网关 IP 地址寄存器 2）	R=W	0x00
0x4173	GA6R3（IPv6 网关 IP 地址寄存器 3）	R=W	0x00
0x4174	GA6R4（IPv6 网关 IP 地址寄存器 4）	R=W	0x00
0x4175	GA6R5（IPv6 网关 IP 地址寄存器 5）	R=W	0x00
0x4176	GA6R6（IPv6 网关 IP 地址寄存器 6）	R=W	0x00
0x4177	GA6R7（IPv6 网关 IP 地址寄存器 7）	R=W	0x00
0x4178	GA6R8（IPv6 网关 IP 地址寄存器 8）	R=W	0x00
0x4179	GA6R9（IPv6 网关 IP 地址寄存器 9）	R=W	0x00
0x417A	GA6R10（IPv6 网关 IP 地址寄存器 10）	R=W	0x00
0x417B	GA6R11（IPv6 网关 IP 地址寄存器 11）	R=W	0x00
0x417C	GA6R12（IPv6 网关 IP 地址寄存器 12）	R=W	0x00
0x417D	GA6R13（IPv6 网关 IP 地址寄存器 13）	R=W	0x00
0x417E	GA6R14（IPv6 网关 IP 地址寄存器 14）	R=W	0x00
0x417F	GA6R15（IPv6 网关 IP 地址寄存器 15）	R=W	0x00
0x4180	SLDIP6R0（SOCKET-less 目标 IP 地址寄存器 0）	R=W	0x00
0x4181	SLDIP6R1（SOCKET-less 目标 IP 地址寄存器 1）	R=W	0x00
0x4182	SLDIP6R2（SOCKET-less 目标 IP 地址寄存器 2）	R=W	0x00
0x4183	SLDIP6R3（SOCKET-less 目标 IP 地址寄存器 3）	R=W	0x00
0x4184	SLDIP6R4（SOCKET-less 目标 IP 地址寄存器 4）	R=W	0x00
0x4185	SLDIP6R5（SOCKET-less 目标 IP 地址寄存器 5）	R=W	0x00
0x4186	SLDIP6R6（SOCKET-less 目标 IP 地址寄存器 6）	R=W	0x00
0x4187	SLDIP6R7（SOCKET-less 目标 IP 地址寄存器 7）	R=W	0x00
0x4188	SLDIP6R8（SOCKET-less 目标 IP 地址寄存器 8）	R=W	0x00
0x4189	SLDIP6R9（SOCKET-less 目标 IP 地址寄存器 9）	R=W	0x00
0x418A	SLDIP6R10（SOCKET-less 目标 IP 地址寄存器 10）	R=W	0x00
0x418B	SLDIP6R11（SOCKET-less 目标 IP 地址寄存器 11）	R=W	0x00
0x418C	SLDIP6R12（SOCKET-less 目标 IP 地址寄存器 12）	R=W	0x00
0x418D	SLDIP6R13（SOCKET-less 目标 IP 地址寄存器 13）	R=W	0x00
0x418E	SLDIP6R14（SOCKET-less 目标 IP 地址寄存器 14）	R=W	0x00
0x418F	SLDIP6R15（SOCKET-less 目标 IP 地址寄存器 15）	R=W	0x00
0x4190	SLDHAR0（SOCKET-less 目标 MAC 地址寄存器 0）	RO	0x00

0x4191	SLDHAR1（SOCKET-less 目标 MAC 地址寄存器 1）	RO	0x00
0x4192	SLDHAR2（SOCKET-less 目标 MAC 地址寄存器 2）	RO	0x00
0x4193	SLDHAR3（SOCKET-less 目标 MAC 地址寄存器 3）	RO	0x00
0x4194	SLDHAR4（SOCKET-less 目标 MAC 地址寄存器 4）	RO	0x00
0x4195	SLDHAR5（SOCKET-less 目标 MAC 地址寄存器 5）	RO	0x00
0x4198	PINGIDR0（PING ID 寄存器 0）	R=W	0x00
0x4199	PINGIDR1（PING ID 寄存器 1）	R=W	0x00
0x419C	PINGSEQR0（PING 序列号寄存器 0）	R=W	0x00
0x419D	PINGSEQR1（PING 序列号寄存器 1）	R=W	0x00
0x41A0	UIPR0（IP 地址不可达寄存器 0）	RO	0x00
0x41A1	UIPR1（IP 地址不可达寄存器 1）	RO	0x00
0x41A2	UIPR2（IP 地址不可达寄存器 2）	RO	0x00
0x41A3	UIPR3（IP 地址不可达寄存器 3）	RO	0x00
0x41A4	UPORTR0（端口不可达寄存器 0）	RO	0x00
0x41A5	UPORTR1（端口不可达寄存器 1）	RO	0x00
0x41B0	UIP6R0（IPv6 IP 地址不可达寄存器 0）	RO	0x00
0x41B1	UIP6R1（IPv6 IP 地址不可达寄存器 1）	RO	0x00
0x41B2	UIP6R2（IPv6 IP 地址不可达寄存器 2）	RO	0x00
0x41B3	UIP6R3（IPv6 IP 地址不可达寄存器 3）	RO	0x00
0x41B4	UIP6R4（IPv6 IP 地址不可达寄存器 4）	RO	0x00
0x41B5	UIP6R5（IPv6 IP 地址不可达寄存器 5）	RO	0x00
0x41B6	UIP6R6（IPv6 IP 地址不可达寄存器 6）	RO	0x00
0x41B7	UIP6R7（IPv6 IP 地址不可达寄存器 7）	RO	0x00
0x41B8	UIP6R8（IPv6 IP 地址不可达寄存器 8）	RO	0x00
0x41B9	UIP6R9（IPv6 IP 地址不可达寄存器 9）	RO	0x00
0x41BA	UIP6R10（IPv6 IP 地址不可达寄存器 10）	RO	0x00
0x41BB	UIP6R11（IPv6 IP 地址不可达寄存器 11）	RO	0x00
0x41BC	UIP6R12（IPv6 IP 地址不可达寄存器 12）	RO	0x00
0x41BD	UIP6R13（IPv6 IP 地址不可达寄存器 13）	RO	0x00
0x41BE	UIP6R14（IPv6 IP 地址不可达寄存器 14）	RO	0x00
0x41BF	UIP6R15（IPv6 IP 地址不可达寄存器 15）	RO	0x00
0x41C0	UPORT6R0（IPv6 端口不可达寄存器 0）	RO	0x00
0x41C1	UPORT6R1（IPv6 端口不可达寄存器 1）	RO	0x00
0x41C5	INTPTMR0（中断挂起时间寄存器 0）	R=W	0x00
0x41C6	INTPTMR1（中断挂起时间寄存器 1）	R=W	0x00
0x41D0	PLR（前缀长度寄存器）	RO	0x00
0x41D4	PFR（前缀标志寄存器）	RO	0x00
0x41D8	VLTR0（有效生存时间寄存器 0）	RO	0x00

0x41D9	VLTR1（有效生存时间寄存器 1）	RO	0x00
0x41DA	VLTR2（有效生存时间寄存器 2）	RO	0x00
0x41DB	VLTR3（有效生存时间寄存器 3）	RO	0x00
0x41DC	PLTR0（首选生存时间寄存器 0）	RO	0x00
0x41DD	PLTR1（首选生存时间寄存器 1）	RO	0x00
0x41DE	PLTR2（首选生存时间寄存器 2）	RO	0x00
0x41DF	PLTR3（首选生存时间寄存器 3）	RO	0x00
0x41E0	PAR0（前缀地址寄存器 0）	RO	0x00
0x41E1	PAR1（前缀地址寄存器 1）	RO	0x00
0x41E2	PAR2（前缀地址寄存器 2）	RO	0x00
0x41E3	PAR3（前缀地址寄存器 3）	RO	0x00
0x41E4	PAR4（前缀地址寄存器 4）	RO	0x00
0x41E5	PAR5（前缀地址寄存器 5）	RO	0x00
0x41E6	PAR6（前缀地址寄存器 6）	RO	0x00
0x41E7	PAR7（前缀地址寄存器 7）	RO	0x00
0x41E8	PAR8（前缀地址寄存器 8）	RO	0x00
0x41E9	PAR9（前缀地址寄存器 9）	RO	0x00
0x41EA	PAR10（前缀地址寄存器 10）	RO	0x00
0x41EB	PAR11（前缀地址寄存器 11）	RO	0x00
0x41EC	PAR12（前缀地址寄存器 12）	RO	0x00
0x41ED	PAR13（前缀地址寄存器 13）	RO	0x00
0x41EE	PAR14（前缀地址寄存器 14）	RO	0x00
0x41EF	PAR15（前缀地址寄存器 15）	RO	0x00
0x41F0	ICMP6BLKR（ICMPv6 块寄存器）	R=W	0x00
0x41F4	CHPLCKR（芯片锁定寄存器）	WO	0x00
0x41F5	NETLCKR（网络锁定寄存器）	WO	0x00
0x41F6	PHYLCKR（PHY 锁定寄存器）	WO	0x00
0x4200	RTR0（重传时间寄存器 0）	R=W	0x07
0x4201	RTR1（重传时间寄存器 1）	R=W	0xD0
0x4204	RCR（重传次数寄存器）	R=W	0x08
0x4208	SLRTR0（SOCKET-less 重传时间寄存器 0）	R=W	0x07
0x4209	SLRTR1（SOCKET-less 重传时间寄存器 1）	R=W	0xD0
0x420C	SLRCR（SOCKET-less 重传次数寄存器）	R=W	0x00
0x420F	SLHOPR（跳数限制寄存器）	R=W	0x80

3.2 SOCKET 寄存器

偏移地址	寄存器	类型[2]	复位值
0x0000	Sn_MR（SOCKET n 模式寄存器）	R=W	0x00
0x0004	Sn_PSR（SOCKET n 首选源 IPv6 地址寄存器）	R=W	0x00
0x0010	Sn_CR（SOCKET n 控制寄存器）	RW,AC	0x00
0x0020	Sn_IR（SOCKET n 中断寄存器）	WO	0x00
0x0024	Sn_IMR（SOCKET n 中断屏蔽寄存器）	R=W	0xFF
0x0028	Sn_IRCLR（Sn_IR 中断清除寄存器）	WO	0xFF
0x0030	Sn_SR（SOCKET n 状态寄存器）	RO	0x00
0x0031	Sn_ESR（SOCKET n 扩展状态寄存器）	RO	0x00
0x0100	Sn_PNR（SOCKET n IP 协议号寄存器）	R=W	0x00
0x0104	Sn_TOSR（SOCKET n IP 服务类型寄存器）	R=W	0x00
0x0108	Sn_TTLR（SOCKET n IP 生存时间寄存器）	R=W	0x80
0x010C	Sn_FRGR0（SOCKET n IP 头分段偏移寄存器 0）	R=W	0x40
0x010D	Sn_FRGR1（SOCKET n IP 头分段偏移寄存器 1）	R=W	0x00
0x0110	Sn_MSSR0（SOCKET n 最大分段寄存器 0）	RW	0x00
0x0111	Sn_MSSR1（SOCKET n 最大分段寄存器 1）	RW	0x00
0x0114	Sn_PORTR0（SOCKET n 源端口寄存器 0）	R=W	0x00
0x0115	Sn_PORTR1（SOCKET n 源端口寄存器 1）	R=W	0x00
0x0118	Sn_DHAR0（SOCKET n 目标 MAC 地址寄存器 0）	RW	0x00
0x0119	Sn_DHAR1（SOCKET n 目标 MAC 地址寄存器 1）	RW	0x00
0x011A	Sn_DHAR2（SOCKET n 目标 MAC 地址寄存器 2）	RW	0x00
0x011B	Sn_DHAR3（SOCKET n 目标 MAC 地址寄存器 3）	RW	0x00
0x011C	Sn_DHAR4（SOCKET n 目标 MAC 地址寄存器 4）	RW	0x00
0x011D	Sn_DHAR5（SOCKET n 目标 MAC 地址寄存器 5）	RW	0x00
0x0120	Sn_DIPR0（SOCKET n IPv4 目标 IP 地址寄存器 0）	RW	0x00
0x0121	Sn_DIPR1（SOCKET n IPv4 目标 IP 地址寄存器 1）	RW	0x00
0x0122	Sn_DIPR2（SOCKET n IPv4 目标 IP 地址寄存器 2）	RW	0x00
0x0123	Sn_DIPR3（SOCKET n IPv4 目标 IP 地址寄存器 3）	RW	0x00
0x0130	Sn_DIP6R0（SOCKET n IPv6 目标 IP 地址寄存器 0）	RW	0x00
0x0131	Sn_DIP6R1（SOCKET n IPv6 目标 IP 地址寄存器 1）	RW	0x00
0x0132	Sn_DIP6R2（SOCKET n IPv6 目标 IP 地址寄存器 2）	RW	0x00
0x0133	Sn_DIP6R3（SOCKET n IPv6 目标 IP 地址寄存器 3）	RW	0x00
0x0134	Sn_DIP6R4（SOCKET n IPv6 目标 IP 地址寄存器 4）	RW	0x00
0x0135	Sn_DIP6R5（SOCKET n IPv6 目标 IP 地址寄存器 5）	RW	0x00

[2] 参考) 4. *通用寄存器*

0x0136	Sn_DIP6R6（SOCKET n IPv6 目标 IP 地址寄存器 6）	RW	0x00
0x0137	Sn_DIP6R7（SOCKET n IPv6 目标 IP 地址寄存器 7）	RW	0x00
0x0138	Sn_DIP6R8（SOCKET n IPv6 目标 IP 地址寄存器 8）	RW	0x00
0x0139	Sn_DIP6R9（SOCKET n IPv6 目标 IP 地址寄存器 9）	RW	0x00
0x013A	Sn_DIP6R10（SOCKET n IPv6 目标 IP 地址寄存器 10）	RW	0x00
0x013B	Sn_DIP6R11（SOCKET n IPv6 目标 IP 地址寄存器 11）	RW	0x00
0x013C	Sn_DIP6R12（SOCKET n IPv6 目标 IP 地址寄存器 12）	RW	0x00
0x013D	Sn_DIP6R13（SOCKET n IPv6 目标 IP 地址寄存器 13）	RW	0x00
0x013E	Sn_DIP6R14（SOCKET n IPv6 目标 IP 地址寄存器 14）	RW	0x00
0x013F	Sn_DIP6R15（SOCKET n IPv6 目标 IP 地址寄存器 15）	RW	0x00
0x0140	Sn_DPORTR0（SOCKET n 目标端口寄存器 0）	RW	0x00
0x0141	Sn_DPORTR1（SOCKET n 目标端口寄存器 1）	RW	0x00
0x0144	Sn_MR2（SOCKET n 模式寄存器 2）	R=W	0x00
0x0180	Sn_RTR0（SOCKET n 重传时间寄存器 0）	RW	0x00
0x0181	Sn_RTR1（SOCKET n 重传时间寄存器 1）	RW	0x00
0x0184	Sn_RCR（SOCKET n 重传次数寄存器）	RW	0x00
0x0188	Sn_KPALVTR（SOCKET n 心跳检测寄存器）	R=W	0x00
0x0200	Sn_TX_BSR（SOCKET n 发送缓存大小寄存器）	R=W	0x02
0x0204	Sn_TX_FSR0（SOCKET n 发送缓存空闲大小寄存器 0）	RO	0x00
0x0205	Sn_TX_FSR1（SOCKET n 发送缓存空闲大小寄存器 1）	RO	0x00
0x0208	Sn_TX_RD0（SOCKET n 发送读指针寄存器 0）	RO	0x00
0x0209	Sn_TX_RD1（SOCKET n 发送读指针寄存器 1）	RO	0x00
0x020C	Sn_TX_WR0（SOCKET n 发送写指针寄存器 0）	RW	0x00
0x020D	Sn_TX_WR1（SOCKET n 发送写指针寄存器 1）	RW	0x00
0x0220	Sn_RX_BSR（SOCKET n 接收缓存大小寄存器）	R=W	0x02
0x0224	Sn_RX_RSR0（SOCKET n 已接收数据大小寄存器 0）	RO	0x00
0x0225	Sn_RX_RSR1（SOCKET n 已接收数据大小寄存器 1）	RO	0x00
0x0228	Sn_RX_RD0（SOCKET n 接收读指针寄存器 0）	RW	0x00
0x0229	Sn_RX_RD1（SOCKET n 接收读指针寄存器 1）	RW	0x00
0x022C	Sn_RX_WR0（SOCKET n 接收写指针寄存器 0）	RO	0x00
0x022D	Sn_RX_WR1（SOCKET n 接收写指针寄存器 1）	RO	0x00

4 寄存器详细说明

寄存器描述示例：

* 寄存器符号(寄存器名称)

[寄存器类型][偏移地址][默认值]

寄存器详细说明....

7		5	4	3	2	1	0
Bit7	Bit6	Bit5	Bit	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

例如：Sn_IR[3:0] 表示：寄存器符号 [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]：寄存器类型

[RW]：可读写

[R=W]：读写值相同

[RO]：只读

[WO]：只写

[W]：只写

[WC]：通过写 1 来清空

[W0]：仅可写入 0

[W1]：仅可写入 1

[AC]：自动清空

[1]：始终读为 1

[0]：始终读为 0

[-]：无法使用

[偏移地址]：寄存器的地址偏移量

[默认值]：默认值

例 1) NETMR (网络模式寄存器) ***[R=W][0x4008][0x00]***

NETMR 设置各种块模式和网络唤醒（WOL）

7	6	5	4	3	2	1	0
-	-	ANB	M6B	-	WOL	IP6B	IP4B
		R=W	R=W		R=W	R=W	R=W

例 2) NETMR[ANB]

NETMR 寄存器的 ANB 位

例 3) NETMR[7:0]

NETMR 寄存器的第 7 位到第 0 位

4.1 通用寄存器

4.1.1 CIDR（芯片 ID 寄存器）

[RO][0x0000~0x0001] [0x6100]

W6100 芯片的 ID 是固定值：0x6100。

CIDR0(0x0000)	CIDR1(0x0001)
0x61	0x00

4.1.2 VER (版本寄存器)

[RO][0x0002~0x0003] [0x4661]

当前芯片版本为 0x4661。

VER0(0x0002)	VER1(0x0003)
0x46	0x61

4.1.3 SYSR (系统状态寄存器)

[RO][0x2000] [0xEU]

SYSR 寄存器用于显示芯片/网络/PHY 配置锁定和主机接口模式的状态。

7	6	5	4	3	2	1	0
CHPL	NETL	PHYL	-	-	-	IND	SPI
RO	RO	RO				RO	RO

位	符号	描述
7	CHPL	芯片锁定状态 芯片锁定由 CHPLCKR 寄存器设置 0: 解锁，解锁状态下 SYC0R 和 SYC1R 寄存器可以设置 1: 锁定，锁定状态下 SYC0R 和 SYC1R 寄存器无法设置
6	NETL	NET 锁定状态 NET 锁定由 NETLCKR 寄存器设置 0: 解锁，解锁状态下可配置 W6100 的 IP 地址、网关等网络信息配置寄存器 1: 锁定，锁定状态下不可配置相关的网络信息配置寄存器 例如： 以下与网络信息配置相关的寄存器可由 NETL 位锁定

IPv4	SHAR	源MAC地址寄存器
	GAR	网关IP地址寄存器
	SUBR	子网掩码寄存器
	SIPR	源IP地址寄存器
IPv6	LLAR	Link本地地址寄存器
	GUAR	全局单播地址寄存器
	SUB6R	IPv6子网前缀寄存器

		* 注意:无论 NETL 处于解锁还是锁定状态，GA6R 都可以设置
5	PHYL	PHY 锁定状态 PHY 锁定由 PHYLCKR 寄存器设置 0: 解锁，解锁状态下可以配置 PHYC0R、PHYC1R 寄存器 1: 锁定，锁定状态下不可配置 PHYC0R、PHYC1R 寄存器
[4:2]	-	保留位
1	IND	并行总线模式 0: 其他 1: PIN MODE[3:0] = “010X”
0	SPI	SPI 接口模式 0: Others 1: PIN MODE[3:0] = “000X”

4.1.4 SYCR0 (系统配置寄存器 0)

[WO][0x2004] [0x80]

SYCR0 可软复位 W6100，仅在 SYSR [CHPL] ='0'（芯片解锁）的情况下才能设置 SYCR0。

7	6	5	4	3	2	1	0
RST	-	-	-	-	-	-	-
WO							

位	符号	描述
7	RST	软件复位 W6100 S/W 复位， 所有的寄存器都将恢复默认值 0： 复位 W6100 1：Normal operation
[6:0]	-	保留位

4.1.5 SYCR1 (系统配置寄存器 1)

[R=W][0x2005] [0x80]

SYCR1 寄存器可用于设置中断使能、系统时钟选择

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

位	符号	描述
7	IEN	中断使能 启用中断 0： 关闭中断失能，INTn 拉高

		1：打开中断使能，有中断产生时，INTn 被拉低
[6:1]	-	保留位
0	CLKSEL	系统时钟选择 只有在 SYSR[CHPL] = '0'（芯片解锁）的状态下有效 0：100MHz 1：25MHz

4.1.6 TCNTR (滴答计数器寄存器)

[RO][0x2016-0x2017][0x0000]

每 100us 自增一次。

4.1.7 TCNTRCLR (TCNTR 清除寄存器)

[WO][0x2020][0x00]

通过对 TCNTCLR 寄存器的写操作来清空 TCNTR 寄存器计数的值。

4.1.8 IR (中断寄存器)

[RO] [0x2100] [0x00]

当网络唤醒（WOL）或者目标地址不可达等中断事件发生时，IR 寄存器的相应的位被置 1。

7	6	5	4	3	2	1	0
WOL			UNR6		IPCONF	UNR4	PTERM
RO			RO		RO	RO	RO

位	符号	描述
7	WOL	WOL 网络唤醒包 0：其他 1：W6100 接收到 WOL MAGIC 包时，该位置 1
[6:5]	-	保留位
4	UNR6	IPv6 的目标端口不可达 0：其他 1：W6100 接收到 ICMPv6 目标端口不可达包时，该位置 1 IPv6 不可达地址和不可达端口分别存储在 UIP6R 和 UPORT6R 寄存器中
3	-	保留位
2	IPCONF	IP 冲突 0：其他 1：IPv4 地址冲突发生时，该位置 1
1	UNR4	目标端口不可达 0: 其他 1: W6100 接收到 ICMPv4 目标端口不可达包时，该位置 1 不可达的 IP 地址和端口号分别存储在 UIPR 和 UPORTR 寄存器中
0	PTERM	PPPoE 连接关闭

		0：其他 1：PPPoE 模式下接收到 PPPT 或者 LCPT 包时，该位置 1

4.1.9 SIR (SOCKET 中断寄存器)

[RO] [0x2101] [0x00]

如果指定 SOCKET 的 Sn_IR 不等于 0，那么 SIR 相对应的第 n 位将被置 1。

例：SOCKET 2 的 Sn_IR 寄存器的值位 0x01（TCP 连接成功），那么 SIR 寄存器的 S2_INT 被置 1。

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

位	符号	描述
[7:0]	Sn_INT-	SOCKET n 中断 0：SOCKET n 的中断未触发，Sn_IR 等于 0 1：SOCKET n 的中断触发，Sn_IR 不等于 0

4.1.10 SLIR (SOCKET-less 中断寄存器)

[RO] [0x2102] [0x00]

当 SLCR 寄存器的命令正确执行后，发生超时或者从 IPv6 的网关收到 ICMPv6 RA 包等情况时，相应位被置 1。

7	6	5	4	3	2	1	0
TOUT	ARP4	PING4	ARP6	PING6	NS	RS	RA
RO	RO	RO	RO	RO	RO	RO	RO

位	符号	描述
7	TOUT	超时中断 0：其他 1：当执行 SOCKET-less 命令后发生超时，该位被置 1。
6	ARP4	ARP 中断 0：其他 1：当执行 SOCKET-less 命令后接收到 ARP 应答时，该位置 1
5	PING4	PING 中断 0：其他 1：当执行 SOCKET-less 命令后接收到 PING 的应答时，该位置 1
4	ARP6	IPv6 ARP 中断 0：其他 1：当执行 SOCKET-less 命令后接收到 ARP6 应答时，该位置 1
3	PING6	IPv6 PING 中断

		0：其他 1：当执行 SOCKET-less 命令后接收到 PING6 应答时，该位置 1
2	NS	DAD NS 中断 0：其他 1：当执行 SOCKET-less 命令后接收到 NA 报文时，该位置 1 NS 位用于 IPv6 的地址冲突检测
1	RS	自动配置 RS 中断 0：其他 1：当执行 SOCKET-less 命令后接收到 RA 时，该位置 1
0	RA	RA 接收中断 0：其他 1：从 IPv6 网关接收到 All-node RA 时，该位置 1

当 SLIR [RS] = '1' 或 SLIR [RA] = '1'时，RA 包的前缀信息将存储到以下对应的寄存器中，并可用于 IPv6 的自动配置。

- PLR (前缀长度寄存器)
- PFR (前缀标志寄存器)
- VLTR (RA 有效生存时间寄存器)
- PLTR (RA 首选生存时间寄存器)
- PAR (前缀地址寄存器)

* ***警告***: *仅当接收到的 RA 消息的第一个选项是源链路层地址（0x01）而第二个选项是前缀信息选项（0x03）时，才能正确设置上述寄存器。否则，需要使用 SOCKET 的 IPRAW6 模式接收 RA 消息并处理前缀信息。*

4.1.11 IMR (中断屏蔽寄存器)

[R=W] [0x2104] [0x00]

IMR 用于打开或者屏蔽相应的中断位。

7	6	5	4	3	2	1	0
WOL			UNR6		IPCONF	UNR4	PTERM
R=W			R=W		R=W	R=W	R=W

位	符号	描述
7	WOL	**屏蔽 WOL(网络唤醒)中断** 0：使能 WOL 中断 1：失能 WOL 中断
[6:5]	-	保留位
4	UNR6	**屏蔽 IPv6 目标端口不可达中断** 0：使能 UNREACH6 中断 1：失能 UNREACH6 中断
3	-	保留位
2	IPCONF	**屏蔽 IPv4 冲突中断**

		1： 使能冲突检测中断 0： 失能冲突检测中断
1	UNR4	**屏蔽目标端口不可达中断** 1： 使能 UNREACH 中断 0： 失能 UNREACH 中断
0	PTERM	**屏蔽 PPPoE 断开中断** 1： 使能 PPPTERM 中断 0： 失能 PPPTERM 中断

4.1.12 IRCLR (IR 清除寄存器)

[W1] [0x2108] [0x00]

当 IRCLR 寄存器与 IR 寄存器对应的位被写 1 时，IR 寄存器对应的位被清空。

7	6	5	4	3	2	1	0
WOL			UNR6		IPCONF	UNR4	PTERM
W1			W1		W1	W1	W1

4.1.13 SIMR (SOCKET 中断屏蔽寄存器)

[R=W] [0x2114] [0x00]

SIMR 屏蔽 SIR 寄存器中对应的位。

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

位	符号	描述
[7:0]	Sn_INT	**SOCKET n 中断屏蔽** 1： 打开 SOCKET n 中断 0： 关闭 SOCKET n 中断

4.1.14 SLIMR (SOCKET-less 中断屏蔽寄存器)

[R=W] [0x2124] [0x00]

SLIMR 屏蔽 SLIR 寄存器中对应的位。

7	6	5	4	3	2	1	0
TOUT	ARP4	PING4	ARP6	PING6	NS	RS	RA
R=W	R=W	R=W	R=W	R=W	R=W	R=W	R=W

位	符号	描述
7	TOUT	**屏蔽 TIMEOUT 中断** 1： 使能 TIMEOUT 中断 0： 失能 TIMEOUT 中断
6	ARP4	**屏蔽 ARP 中断**

		1： 使能 ARP4 中断 0： 失能 ARP4 中断
5	PING4	**屏蔽 PING 中断** 1： 使能 PING4 中断 0： 失能 PING4 中断
4	ARP6	**屏蔽 IPv6 ARP 中断** 1： 使能 ARPv6 中断 0： 失能 ARPv6 中断
3	PING6	**屏蔽 IPv6 PING 中断** 1： 使能 PINGv6 中断 0： 失能 PINGv6 中断
2	NS	**屏蔽 DAD NS 中断** 1： 使能 DAD NS 中断 0： 失能 DAD NS 中断
1	RS	**屏蔽 Auto configuration RS 中断** 1： 使能 AUTO RS 中断 0： 失能 AUTO RS 中断
0	RA	**屏蔽 RA Receive 中断** 1： 使能 RA RECV 中断 0： 失能 RA RECV 中断

4.1.15 SLIRCLR (SLIR 清除寄存器)

[W1] [0x2128] [0x00]

当 SLIRCLR 寄存器与 SLIR 寄存器对应的位被写 1 时， SLIR 对应的位被清空。

7	6	5	4	3	2	1	0
TOUT	ARP4	PING4	ARP6	PING6	NS	RS	RA
W1	W1	W1	W1	W1	W1	W1	W1

4.1.16 SLPSR (SOCKET-less 首选源 IPv6 地址寄存器)

[R=W] [0x212C] [0x00]

SLPSR 设置由 SLCR 寄存器发送的 IPv6 包的源地址。

值	符号	描述
0x00	AUTO	根据目标 IPv6 地址(SLDIP6R: SOCKET-less 目标 IPv6 地址寄存器) 选择源 IPv6 地址(SIP6) 如果 SLDIP6R 的值是 LLA，则 SIP6 被设置为 LLAR 如果 SLDIP6R 的值是 GUA，则 SIP6 被设置为 GUAR
0x02	LLA	SIP6 的值固定为 LLAR
0x03	GUA	SIP6 的值固定为 GUAR

4.1.17 SLCR (SOCKET-less 控制寄存器)

[RW, AC] [0x2130] [0x00]

在没有初始化 SOCKET 的情况下 SLCR 寄存器可以配置发送一个指定的报文，发送命令完成后自动清除，并且在清除上一条命令之前无法执行下一条命令。执行命令的结果可通过 SLIR (SOCKET-less 中断寄存器)查询。

7	6	5	4	3	2	1	0
-	ARP4	PING4	ARP6	PING6	NS	RS	UNA
	RW	RW	RW	RW	RW	RW	RW

位	符号	描述
7	-	保留位
6	ARP4	**ARP 请求命令** 1： 发送 ARP 请求 0： 准备状态
5	PING4	**IPv4 PING 请求命令** 1： 发送 PING 请求 0： 准备状态
4	ARP6	**NS ARP 命令** 1： 发送 NS ARP 0： 准备状态
3	PING6	**IPv6 PING 请求命令** 1： 发送 IPv6 PING 请求 0： 准备状态
2	NS	**DAD 的 NS 命令** 1：DAD 发送 NS 包 0： 准备状态
1	RS	**自动配置 RS 命令** 1： 发送 RS 包 0： 准备状态
0	UNA	**未经请求的 NA 命令** 1： 发送未经请求的 NA 包 0： 准备状态

4.1.18 PHYSR (PHY 状态寄存器)

[RO] [0x3000] [0x00]

PHYSR 通过 PHYCR0(PHY 控制寄存器 0)检查 PHY 工作模式和链路状态设置。

7	6	5	4	3	2	1	0
CAB	-	MODE2	MODE1	MODE0	DPX	SPD	LNK
RO		RO	RO	RO	RO	RO	RO

位	符号	描述
7	CAB	**网线插入状态** 1： 网线未插入 0： 网线插入
6	-	保留位
[5:3]	MODE[2:0]	**PHY OPMODE** MODE2 / MODE1 / MODE0 / 描述 0 / X / X / 自动协商 1 / 0 / 0 / 100BASE-TX 全双工 1 / 0 / 1 / 100BASE-TX 半双工 1 / 1 / 0 / 10BASE-T 全双工 1 / 1 / 1 / 10BASE-T 半双工
2	DPX	**双工标志位** 1： 半双工 0： 全双工
1	SPD	**速度标志位** 1：10Mbps 0：100Mbps
0	LNK	**Link 标志位** 1： 连接成功 0： 断开连接

4.1.19 PHYRAR (PHY 寄存器地址寄存器)

[R=W] [0x3008] [0x00]

PHYRAR 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
[7:5]	-	保留位
[4:0]	ADDR[4:0]	**PHY 寄存器地址** 设置 PHY 寄存器的地址

4.1.20 PHYDIR (PHY 数据输入寄存器)

[R=W] [0x300C-0x300D] [0x0000]

PHYDIR 寄存器的作用是将值写入到由 PHYRAR 寄存器指定的 PHY 寄存器中。

例如： PHYDIR = 0x1234

PHYDIR0(0x300C)	PHYDIR1(0x300D)
0x34	0x12

4.1.21 PHYDOR (PHY 数据输出寄存器)

[RO] [0x3010-0x3011] [0x0000]

PHYDOR 寄存器的作用是获取由 PHYRAR 寄存器指定的 PHY 寄存器中的值。

例如： PHYDOR = 0x1234

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

4.1.22 PHYACR (PHY 访问控制寄存器)

[RW, AC] [0x3014] [0x00]

PHYACR 寄存器的作用是配置由 PHYRAR 寄存器指定的 PHY 寄存器的访问类型，配置完成后，PHYACR 寄存器会自动清除。

访问类型	值	寄存器
Write	0x01	PHYDIR
Read	0x02	PHYDOR

4.1.23 PHYDIVR (PHY 分频寄存器)

[R=W] [0x3018] [0x01]

PHYDIVR 是 PHY 的 MDC 时钟分频寄存器 (注意不要超过 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

4.1.24 PHYCR0 (PHY 控制寄存器 0)

[WO] [0x301C] [0x00]

PHYCR0 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 PHYCR0 之前，PHYLCKR（PHY 锁定寄存器）必须处于解锁模式。

PHYCR0 的配置可通过 PHYSR [5:3]位来查询。

7	6	5	4	3	2	1	0
-	-	-	-	-	MODE2	MODE1	MODE0
					WO	WO	WO

位	符号	描述
[7:3]	-	保留位

[2:0]	MODE	

MODE2	MODE1	MODE0	描述
0	x	x	自动协商
1	0	0	100BASE-TX 全双工
1	0	1	100BASE-TX 半双工
1	1	0	10BASE-TX 全双工
1	1	1	10BASE-TX 半双工

4.1.25 PHYCR1 (PHY 控制寄存器 1)

[R=W] [0x301D] [0x40]

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 PHYCR1 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	-	TE	-	-	RST
-	-	R=W	-	R=W	-	-	AC

位	符号	描述
7	-	保留位
6	-	始终写‘1’
5	PWDN	**PHY 掉电模式** 0： 关闭 PHY 的掉电模式，SYS_CLK 由 SYCR1[CLKSEL]来决定 1： 打开 PHY 的掉电模式，SYS_CLK 切换到 25MHz ***参考)***8.4.1 复位时序
4	-	保留位
3	TE	10BASE-Te 模式 仅在 PHYSR [MODE2：MODE0] = '000'的情况下有效。 0： 关闭 10BASE-Te 模式 1： 打开 10BASE-Te 模式
[2:1]	-	保留位
0	RST	**PHY 复位** 在 PHY 硬件复位时，SYS_CLK 切换到 25MHz。 复位完成后该位自动清零，SYS_CLK 恢复为先前设置的是时钟。 ***参考)*** 8.4.1 复位时序 0 :正常运行 1：PHY 硬件复位

SYCR1[CLKSEL]	SYS_CLK
0	100 MHz
1	25 MHz

4.1.26 NET4MR (IPv4 网络模式寄存器)

[R=W] [0x4000] [0x00]

NET4MR 寄存器为 IPv4 提供了一些特殊功能的设置。

7	6	5	4	3	2	1	0
-	-	-	-	UNRB	PARP	RSTB	PB
-	-	-	-	*R=W*	*R=W*	*R=W*	*R=W*

位	符号	描述
[7:4]	-	保留位
3	UNRB	**UDP4 目标端口不可达包** 当 UDP4 数据包传输到处于关闭状态的 SOCKET 时，发送目标端口不可达包。该状态下可能成为 UDP 端口扫描攻击的目标，为了防止这种情况，可以阻塞目标端口不可达包的传输 0： 不阻塞 1： 阻塞
2	PARP	**ARPv4 应答** 可设置在 PINGv4 应答之前发送 ARPv4 0： 关闭 1： 打开
1	RSTB	**TCP4 RST 包** 当发送 SYN 包至一个非监听状态的 SOCKET 时，芯片会向对方发送一个 RST 包，该状态下可能成为 TCP 端口扫描攻击的目标，为了防止这种情况，可以阻塞传输 RST 包 0： 不阻塞 1： 阻塞
0	PB	**PINGv4 回复** 可设置为不回复 PINGv4 的请求 0： 回复 PINGv4 请求 1： 不回复 PINGv4 请求

4.1.27 NET6MR (IPv6 网络模式寄存器)

[R=W] [0x4004] [0x00]

NET6MR 寄存器为 IPv6 提供了一些特殊功能的设置。

7	6	5	4	3	2	1	0
-	-	-	-	UNRB	PARP	RSTB	PB
-	-	-	-	*R=W*	*R=W*	*R=W*	*R=W*

位	符号	描述
[7:4]	-	保留位

3	UNRB	**UDP6 目标端口不可达包** 当 UDP6 数据包传输到处于关闭状态的 SOCKET 时，发送目标端口不可达包。该状态下可能成为 UDP 端口扫描攻击的目标，为了防止这种情况，可以阻塞目标端口不可达包的传输。 0：不阻塞 1：阻塞
2	PARP	**ARPv6 应答** 可设置在 PINGv6 应答之前发送 ARPv6 0：关闭 1：打开
1	RSTB	**TCP6 RST Packet Block** 当发送 SYN 包至一个非监听状态的 SOCKET 时，芯片会向对方发送一个 RST 包，该状态下可能成为 TCP 端口扫描攻击的目标，为了防止这种情况，可以阻塞传输 RST 包 0：不阻塞 1：阻塞
0	PB	**PINGv6 响应** 可设置为不回复 PINGv6 的请求 0：不阻塞 1：阻塞

4.1.28 NETMR (网络模式寄存器)

[R=W] [0x4008] [0x00]

NETMR 设置阻塞模式和网络唤醒（WOL）。

7	6	5	4	3	2	1	0
-	-	ANB	M6B	-	WOL	IP6B	IP4B
-	-	*R=W*	*R=W*	-	*R=W*	*R=W*	*R=W*

位	符号	描述
[7:6]	-	保留位，始终为低电平
5	ANB	**IPv6 全部阻塞** 全节点多播地址阻塞 PING6 请求 0：开启 1：关闭
4	M6B	**IPv6 组播阻塞** 阻塞 PING6 的多播组地址请求 0：关闭 1：打开
3	-	保留位，始终为低电平

2	WOL	**WOL(网络唤醒)** 0： 关闭 1： 打开
1	IP6B	**IPv6 包阻塞** 0： 不阻塞 1： 阻塞，此时 ANB 和 M6B 位可以忽略
0	IP4B	**IPv4 包阻塞** 0： 不阻塞 1： 阻塞

4.1.29 NETMR2 (网络模式寄存器 2)

[R=W] [0x4009] [0x00]

NETMR2 设置 PPPoE 模式。

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

位	符号	描述
7	DHAS	**ARP/ND 过程中，目标硬件地址选择** 0： 选择以太网 MAC 层数据帧 1： 选择 ARP 目标 MAC 地址
[6:1]	-	保留位
0	PPPoE	**PPPoE 模式** 0 :关闭 PPP 模式 1 :打开 PPP 模式

4.1.30 PTMR (PPP 链路控制协议请求时间寄存器)

[R=W] [0x4100] [0x28]

PTMR 设置发送 LCP 响应请求的时间。

单位是 25ms，并且仅在 PPPoE 模式下有效。

例如： PTMR = 200 (0xC8),

200 * 25ms = 5s

4.1.31 PMNR (PPP 链路控制协议 Magic Number 寄存器)

[R=W] [0x4104] [0x00]

PMNR 设置 4 字节的 Magic Number 用于 LCP 协商。

PMNR 仅在 PPPoE 模式下有效。

例如： PMNR = 0x01

PMNR(0x4104)
0x01

LCP Magic Number = 0x01010101

4.1.32 PHAR (PPPoE 服务器硬件地址寄存器)

[R=W] [0x4108-0x410D] [0x0000]

PHAR 设置 PPPoE 目标硬件地址。

PHAR 仅在 PPPoE 模式下有效。

*例如：*PHAR = “11:22:33:AA:BB:CC”

PHAR0(0x4108)	PHAR1(0x4109)	PHAR2(0x410A)
0x11	0x22	0x33
PHAR3(0x410B)	PHAR4(0x410C)	PHAR5(0x410D)
0xAA	0xBB	0xCC

4.1.33 PSIDR (PPPoE 会话 ID 寄存器)

[R=W] [0x4110-0x4111] [0x0000]

PSIDR 设置 PPPoE 会话 ID。

PSIDR 仅在 PPPoE 模式下有效。

*例如：*PSIDR = 0x1234

PSIDR0(0x4110)	PSIDR1(0x4111)
0x12	0x34

4.1.34 PMRUR (PPPoE 最大接收单元寄存器)

[R=W] [0x4114-0x4115] [0xFFFF]

PMRUR 设置 PPPoE 模式下的 MRU (最大接收单元) 。如果 PMRUR 设置的值大于 1472，则自动设置成 1472，PMRUR 必须在创建 SOCKET 之前设置(Sn_CR [OPEN] = '1')。

PMRUR 仅在 PPPoE 模式下有效。

*例如：*PMUR = 1000 (0x03E8)

PMUR0(0x4114)	PMUR1(0x4115)
0x03	0xE8

4.1.35 SHAR (源硬件地址寄存器)

[R=W] [0x4120-0x4125] [0x00000_0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器) 未锁定状态下)时，SHAR 设置源硬件地址。

*例如：*SHAR = “11:22:33:AA:BB:CC”

SHAR0(0x4120)	SHAR1(0x4121)	SHAR2(0x4122)
0x11	0x22	0x33
SHAR3(0x4123)	SHAR4(0x4124)	SHAR5(0x4125)
0xAA	0xBB	0xCC

4.1.36 GAR (网关 IP 地址寄存器)

[R=W] [0x4130-0x4133] [0x0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器) 未锁定状态下)，GAR 设置源网关地址。

例如：GAR = “192.168.0.1”

GAR0(0x4130)	GAR1(0x4131)	GAR2(0x4132)	GAR3(0x4133)
192 (0xC0)	168 (0xA8)	0 (0x00)	1 (0x01)

4.1.37 SUBR (子网掩码寄存器)

[R=W] [0x4134-0x4137] [0x0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器)未锁定状态下)，SUBR 设置子网掩码。

例如：SUBR = “255.255.255.255”

SUBR0(0x4134)	SUBR0(0x4135)	SUBR0(0x4136)	SUBR0(0x4137)
255 (0xFF)	255 (0xFF)	255 (0xFF)	255 (0xFF)

4.1.38 SIPR (IPv4 源地址寄存器)

[R=W] [0x4138-0x413B] [0x0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器) 未锁定状态下)，SIPR 设置源 IP 地址寄存器。

例如：SIPR = “192.168.0.100”

SIPR0(x4138)	SIPR1(0x4139)	SIPR2(0x413A)	SIPR3(0x413B)
192 (0xC0)	168 (0xA8)	0 (0x00)	100(0x64)

4.1.39 LLAR (Link 本地地址寄存器)

[R=W] [0x4140-0x414F]
[0x0000_0000_0000_0000_0000_0000_0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器) 未锁定状态下)，LLAR 设置 link 本地地址。

例如：LLAR = “FE80::AB:CDEF”

LLAR0(0x4140)	LLAR1(0x4141)	LLAR2(0x4142)	LLAR3(0x4143)
0xFE	0x80	0x00	0x00
LLAR4(0x4144)	LLAR5(0x4145)	LLAR6(0x4146)	LLAR7(0x4147)
0x00	0x00	0x00	0x00
LLAR8(0x4148)	LLAR9(0x4149)	LLAR10(0x414A)	LLAR11(0x414B)
0x00	0x00	0x00	0x00
LLAR12(0x414C)	LLAR13(0x414D)	LLAR14(0x414E)	LLAR15(0x414F)
0x00	0xAB	0xCD	0xEF

4.1.40 GUAR (全局单播地址寄存器)

[R=W] [0x4150-0x415F]
[0x0000_0000_0000_0000_0000_0000_0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器) 未锁定状态下)，GUAR 设置全球单播地址寄存器。

例如： GUAR = “2001::AB:CDEF”

GUAR0(0x4150)	GUAR1(0x4151)	GUAR2(0x4152)	GUAR3(0x4153)
0x20	0x01	0x00	0x00
GUAR4(0x4154)	GUAR5(0x4155)	GUAR6(0x4156)	GUAR7(0x4157)
0x00	0x00	0x00	0x00
GUAR8(0x4158)	GUAR9(0x4159)	GUAR10(0x415A)	GUAR11(0x415B)
0x00	0x00	0x00	0x00
GUAR12(0x415C)	GUAR13(0x415D)	GUAR14(0x415E)	GUAR15(0x415F)
0x00	0xAB	0xCD	0xEF

4.1.41 SUB6R (IPv6 子网前缀寄存器)

[R=W] [0x4160-0x416F]
[0x0000_0000_0000_0000_0000_0000_0000_0000]

当 SYSR [NETL] = '0' (NETLCKR (网络锁定寄存器)未锁定状态下)，SUB6R 设置前缀掩码。

例如： SUB6R = “FFFF:FFFF:FFFF:FFFF::”

PRFXR0(0x4160)	PRFXR1(0x4161)	PRFXR2(0x4162)	PRFXR3(0x4163)
0xFF	0xFF	0xFF	0xFF
PRFXR4(0x4164)	PRFXR5(0x4165)	PRFXR6(0x4166)	PRFXR7(0x4167)
0xFF	0xFF	0xFF	0xFF
PRFXR8(0x4168)	PRFXR9(0x4169)	PRFXR10(0x416A)	PRFXR11(0x416B)
0x00	0x00	0x00	0x00
PRFXR12(0x416C)	PRFXR13(0x416D)	PRFXR14(0x416E)	PRFXR15(0x416F)
0x00	0x00	0x00	0x00

4.1.42 GA6R (IPv6 网关 IP 地址寄存器)

[R=W] [0x4170-0x417F]
[0x0000_0000_0000_0000_0000_0000_0000_0000]

GA6R 设置 IPv6 的网关 IP 地址。

例如： GA6R = “FE80::FE:DCBA”

GA6R0(0x4170)	GA6R1(0x4171)	GA6R2(0x4172)	GA6R3(0x4173)

0xFE	0x80	0x00	0x00
GA6R4(0x4174)	GA6R5(0x4175)	GA6R6(0x4176)	GA6R7(0x4177)
0x00	0x00	0x00	0x00
GA6R8(0x4178)	GA6R9(0x4179)	GA6R10(0x417A)	GA6R11(0x417B)
0x00	0x00	0x00	0x00
GA6R12(0x417C)	GA6R13(0x417D)	GA6R14(0x417E)	GA6R15(0x417F)
0x00	0xFE	0xDC	0xBA

4.1.43 SLDIP6R (SOCKET-less IPv6 目标地址寄存器)

[R=W] [0x4180-0x418F] [0x0000_0000_0000_0000_0000_0000_0000_0000]

SLDIPR 为通过 SLCR 传输的数据包设置 IPv6 的目标 IP 地址。

*例如：*SLDIPR = “FE80::AB:CDEF”

SLDIP6R0(0x4180)	SLDIP6R1(0x4181)	SLDIP6R2(0x4182)	SLDIP6R3(0x4183)
0xFE	0x80	0x00	0x00
SLDIP6R4(0x4184)	SLDIP6R5(0x4185)	SLDIP6R6(0x4186)	SLDIP6R7(0x4187)
0x00	0x00	0x00	0x00
SLDIP6R8(0x4188)	SLDIP6R9(0x4189)	SLDIP6R10(0x418A)	SLDIP6R11(0x418B)
0x00	0x00	0x00	0x00
SLDIP6R12(0x418C)	SLDIP6R13(0x418D)	SLDIP6R14(0x418E)	SLDIP6R15(0x418F)
0x00	0xAB	0xCD	0xEF

4.1.44 SLDIPR (SOCKET-less IPv4 目标 IP 地址寄存器)

[R=W] [0x418C-0x418F] 0x00000000]

SLDIPR 为通过 SLCR 传输的数据包设置 IPv4 的目标 IP 地址。

*例如：*SLDIPR = “192.169.0.21”

SLDIPR0 / SLDIP6R12(0x418C)	SLDIPR1 / SLDIP6R13(0x418D)	SLDIPR2 / SLDIP6R14(0x418E)	SLDIPR3 / SLDIP6R15(0x418F)
192(0xC0)	168(0xA8)	0(0x00)	21(0x15)

4.1.45 SLDHAR (SOCKET-less 目标硬件地址寄存器)

[RO] [0x4190-0x4195] [0x0000_0000_0000]

当收到 SLCR[ARP4] 或者 SLCR[ARP6] 响应包时，SLDHAR 设置目标硬件地址。

*例如：*SLDHAR = “11:22:33:AA:BB:CC”

SLDHAR0(0x4190)	SLDHAR1(0x4191)	SLDHAR2(0x4192)
0x11	0x22	0x33
SLDHAR3(0x4193)	SLDHAR4(0x4194)	SLDHAR5(0x4195)
0xAA	0xBB	0xCC

4.1.46 PINGIDR (PING ID 寄存器)

[R=W] [0x4198-0x4199] [0x0000]

PINGIDR 设置由 SLCR [PING4] 或者 SLCR [PING6]发送的 ping 请求包的 ID。

例如： PINGIDR = 256 (0x0100)

PINGIDR0(0x4198)	PINGIDR1(0x4199)
0x61	0x00

4.1.47 PINGSEQR (PING 序列号寄存器)

[R=W] [0x419C-0x419D] [0x0000]

PINGSEQR 设置由 SLCR [PING4]或者 SLCR [PING6]发送 PING 请求包的序列号，并且不会自动增加。

例如： PINGSEQR = 1000 (0x03E8)

PINGSEQR0(0x419C)	PINGSEQR1(0x419D)
0x03	0xE8

4.1.48 UIPR (不可达 IP 地址寄存器)

[RO] [0x41A0-0x41A3] [0x0000_0000]

当接收到 ICMPv4 不可达包 (IR[UNR4] = '1')时，UIPR 被设置为接收到数据包的 IPv4 目标不可达 IP 地址。

例如： 不可达 IP 地址= “192.169.10.10”

UIPR0(0x41A0)	UIPR1(0x41A1)	UIPR2(0x41A2)	UIPR3(0x41A3)
192(0xC0)	168(0xA8)	10(0x0A)	10(0x0A)

4.1.49 UPORTR (不可达端口寄存器)

[RO] [0x41A4-0x41A5] [0x0000]

当接收到 ICMPv4 的不可达报文(IR[UNR4] = '1')时，UPORTR 被设置为接收数据包的目标不可达端口号。

例如： 不可达端口号= “3000” (0x0BB8)

UPORTR0(0x41A4)	UPORTR1(0x41A5)
0x0B	0xB8

4.1.50 UIP6R (IPv6 不可达地址寄存器)

[RO] [0x41B0-0x41BF] [0x0000_0000_0000_0000_0000_0000_0000_0000]

当接收到 ICMPv6 不可达数据包（IR [UNR6] ='1'）时，UIP6R 被设置为接收数据包的目标 IPv6 不可达地址。

例如： 不可达 IP 地址：“FE80::AB:CDEF”

UIP6R0(0x41B0)	UIP6R1(0x41B1)	UIP6R2(0x41B2)	UIP6R3(0x41B3)
0xFE	0x80	0x00	0x00
UIP6R4(0x41B4)	UIP6R5(0x41B5)	UIP6R6(0x41B6)	UIP6R7(0x41B7)

0x00	0x00	0x00	0x00
UIP6R8(0x41B8)	UIP6R9(0x41B9)	UIP6R10(0x41BA)	UIP6R11(0x41BB)
0x00	0x00	0x00	0x00
UIP6R12(0x41BC)	UIP6R13(0x41BD)	UIP6R14(0x41BE)	UIP6R15(0x41BF)
0x00	0xAB	0xCD	0xEF

4.1.51 UPORT6R (IPv6 不可达端口号寄存器)

[RO] [0x41C0-0x41C1] [0x0000]

当接收 ICMPv6 无法到达数据包（IR [UNR6] ='1'）时，UPORT6R 被设置为接收数据包的目标端口。

*例如：*不可达端口号= “3000” (0x0BB8)

UPORT6R0(0x41C0)	UPORT6R1(0x41C1)
0x0B	0xB8

4.1.52 INTPTMR (中断挂起时间寄存器)

[RW][0x41C5-0x41C6][0x0000]

INTPTMR 用于设置内部中断挂起计时器计数，当 INTn 被置为高电平时，定时器计数由 INTPTMR 中的值初始化，并且在中断发生时从 SYS_CLK 的 4 个时钟周期开始递减 1，直到变为 0。

当发生中断且相应的中断屏蔽被使能且 INTPTMR = 0 时，INTn 被置为低电平。

*例如：*INTPTMR = 1000(0x03EB)

INTPTMR0(0x41C5)	INTPTMR1(0x41C6)
0x03	0xEB

4.1.53 PLR (前缀长度寄存器)

[RO] [0x41D0] [0x00]

PLR 设置为路由器接收到的 RA(路由器通告)消息前缀信息选项中的前缀长度字段(SLIR[RS] = 1 或 SLIR[RA] = ' 1 ')。

*例如：*RA 前缀长度= 0x10

PLR(0x41D0)
0x10

4.1.54 PFR (前缀标志寄存器)

[RO] [0x41D4] [0x00]

PFR 被设置为从路由器接收的 RA（路由器通告）消息的前缀信息选项中的前缀标志字段（SLIR [RS] = 1 或 SLIR [RA] ='1'）。

*例如：*Flag = 0xC0

PFR(0x41D4)
0xC0

4.1.55 VLTR (有效生存时间寄存器)

[RO] [0x41D8-0x41DB] [0x0000_0000]

VLTR 设置为路由器接收到的 RA(路由器通告)消息前缀信息选项中的有效生命时间字段(SLIR[RS] = 1 或 SLIR[RA] = ' 1 ')。

*例如：*有效生存时间 = 2592000

VLTR0(0x41D8)	VLTR1(0x41D9)	VLTR2(0x41DA)	VLTR3(0x41DB)
0x00	0x27	0x8D	0x00

4.1.56 PLTR (首选生存时间寄存器)

[RO] [0x41DC-0x41DF] [0x0000_0000]

PLTR 在路由器接收到的 RA(路由器通告)消息的前缀信息选项中设置为首选生存时间字段(SLIR[RS] = 1 或 SLIR[RA] = ' 1 ')。

*例如：*首选生存时间= 604800

PLTR0(0x41DC)	PLTR1(0x41DD)	PLTR2(0x41DE)	PLTR3(0x41DF)
0x00	0x09	0x3A	0x80

4.1.57 PAR (前缀地址寄存器)

[RO] [0x41E0-0x41EF] [0x0000_0000_0000_0000_0000_0000_0000_0000]

PAR 设置为路由器接收到的 RA(路由器通告)消息前缀信息选项中的首选地址字段(SLIR[RS] = 1 或 SLIR[RA] = ' 1 ')。

*例如：*前缀地址是 “2001:2b8:10:1::”

PAR0(0x41E0)	PAR1(0x41E1)	PAR2(0x41E2)	PAR3(0x41E3)
0x20	0x01	0x02	0xb8
PAR4(0x41E4)	PAR5(0x41E5)	PAR6(0x41E6)	PAR7(0x41E7)
0x00	0x10	0x00	0x01
PAR8(0x41E8)	PAR9(0x41E9)	PAR10(0x41EA)	PAR11(0x41EB)
0x00	0x00	0x00	0x00
PAR12(0x41EC)	PAR13(0x41ED)	PAR14(0x41EE)	PAR15(0x41EF)
0x00	0x00	0x00	0x00

4.1.58 ICMP6BLKR (ICMPv6 阻塞寄存器)

[R=W] [0x41F0] [0x00]

ICMP6BLKR 选择性地阻塞 ICMPv6 包，例如 PING6、MLD 查询包、路由器通告 (RA)、邻节点通告(NA)和邻节点请求 (NS)。被阻塞包可以通过 IPRAW6 的 SOCKET 来接收。

7	6	5	4	3	2	1	0
			PING6	MLD	RA	NA	NS

			R=W	R=W	R=W	R=W	R=W

位	符号	描述
[7:5]	-	保留位
4	PING6	**ICMPv6 阻塞请求应答包** 1： 阻塞请求应答包 0： 不阻塞
3	MLD	**ICMPv6 阻塞 MLD 查询包** 1： 阻塞 MLD 查询包 0： 不阻塞
2	RA	**ICMPv6 阻塞路由器通告** 1： 阻塞路由器通告包 0： 不阻塞
1	NA	**ICMPv6 阻塞邻节点通告** 1： 阻塞邻节点通告包 0： 不阻塞
0	NS	**ICMPv6 阻塞邻节点请求** 1： 阻塞邻节点请求包 0： 不阻塞

4.1.59 CHPLCKR (芯片锁定寄存器)

[WO] [0x41F4] [0x00]

CHPLCKR 设置 SYSR[CHPL]，如果 SYSR [CHPL] 是解锁状态，SYCR0 和 SYCR1 可以设置。

解锁	锁定
0xCE	其他值

4.1.60 NETLCKR (网络锁定寄存器)

[WO] [0x41F5] [0x00]

NETLCKR 设置 SYSR[NETL]。

如果 SYSR [NETL] 处于解锁状态，网络配置寄存器(SHAR, GAR, SUBR, SIPR, LLAR, GUAR, SUB6R)可以被设置。

解锁	锁定
0x3A	0xC5

4.1.61 PHYLCKR (PHY 锁定寄存器)

[WO] [0x41F6] [0x00]

PHYLCKR 设置 SYSR[PHYL]。

如果 SYSR[PHYL]处于解锁状态，PHYCR0 和 PHYCR1 可以被设置。

解锁	锁定

0x53	其他值

4.1.62 RTR (重传时间寄存器)

[R=W] [0x4200-0x4201] [0x07D0]

RTR 设置 Sn_RTR (SOCKET n 重传时间寄存器)的初始值，单位是 100us。该寄存器与 RCR (重传次数寄存器)配合使用来重传 ARP/ND、TCP 等报文。

请参阅 *6.7 重传*。

例如：RTR = 5000 (0x1388)

5000*100us = 0.5s

RTR0(0x4200)	RTR1(0x4201)
0x13	0x88

4.1.63 RCR (重传次数寄存器)

[R=W] [0x4204] [0x08]

RCR 设置 Sn_RCR (SOCKET n 重传次数寄存器)的初始值，该寄存器与 RTR (重传时间寄存器)配合使用来重传 ARP/ND、TCP 等报文。

请参阅 *6.7 重传*。

4.1.64 SLRTR (SOCKET-less 重传时间寄存器)

[R=W] [0x4208-0x4209] [0x07D0]

SLRTR 设置 SLCR 寄存器的重传时间，单位是 100us。

在重传时间内，如果没有接受到对 SLCR 发送的请求报文的响应，会触发重传机制。如果重传次数超过 SLRCR (SOCKET-less 重传次数寄存器)的值，则会触发超时(SLIR [TOUT] = '1')。

请参阅 *6.7 重传*。

例如：SLRTR = 5000 (0x1388),

5000 * 100us = 0.5s

SLRTR0(0x4208)	SLRTR1(0x4209)
0x013	0x88

4.1.65 SLRCR (SOCKET-less 重传次数寄存器)

[R=W] [0x420C] [0x00]

SLRCR 设置 SLCR 寄存器的重传次数，如果重传次数超过了 SLRCR 的值，则 SLIR [TOUT] 变为'1'。

请参阅 *6.7 重传*。

4.1.66 SLHOPR (跳数限制寄存器)

[RW] [0x420F] [0x80]

设置 SLCR 发送的 ND 消息(NS、NA)的跳数。

例如：SLHOPR = 128

SLHOPR(0x420F)

0x80 (128)

4.2 SOCKET 寄存器

4.2.1 Sn_MR (SOCKET n 模式寄存器)

[R=W] [0x0000] [0x00]

Sn_MR 设置 SOCKET 模式和选项，必须在 SOCKET OPEN (Sn_CR[OPEN] = '1')之前设置。

7	6	5	4	3	2	1	0
MULTI/ MF	BRDB/ FPSH	ND/ MC/ SMB/ MMB	UNIB/ MMB6	P3	P2	P1	P0
R=W	R=W	R=W	R=W	R=W	R=W	R=W	R=W

位	符号	描述
7	MULTI/MF	**MULTI： 使能组播模式** 当 Sn_MR[3:0] 为 UDP4、UDP6 或者 UDPD 时有效 *参考）6.3.3 udp 组播* 0： 关闭 UDP 组播 1： 打开 UDP 组播 **MF： 使能 MAC 过滤** 当 Sn_MR[3:0]为 MACRAW 模式时有效 0： 关闭 MAC 过滤(接收所有的包) 1： 打开 MAC 过滤(仅接收多播、广播和源 MAC(SHAR) 地址的包)
6	BRDB/FPSH	**BRDB： 广播阻塞** 仅适用于 Sn_MR[3:0]在 UDP4、UDP6、UDPD 或者 MACRAW 模式下 *参考） 6.3.2 udp 广播* 0： 关闭 UDP 广播阻塞 1： 打开 UDP 广播阻塞 **FPSH： 强制 Push 标志** 当 Sn_MR[3:0] 为 TCP4、TCP6 或者 TCPD 模式时，强制设置数据包中的 PSH 标志。 0： 关闭强制 PSH 标志(仅在 SEND 命令发送的最后一个数据包中设置 PSH 标志) 1： 打开强制 PSH 标志
5	ND/MC/ SMB/MMB	**ND： 无延时 ACK** 仅适用于 Sn_MR[3:0] 在 TCP4、TCP6 或者 TCPD 模式下 0： 关闭无延时 ACK (在 Sn_RTR 设置的时间后发送 ACK 包)

		1： 打开无延时 ACK (在接收到数据包后立即发送 ACK 包) ***注：在 Sn_CR[RECV]后，如果 SOCKET 窗口大小小于 MSS（最大传输单元），则立即发送 ACK 包，此时与 ND 无关。*** **MC： 组播 IGMP 版本** 当 Sn_MR[3:0] 为 UDP4 并且 Sn_MR[MULTI] 为‘1’时 MC 位有效 0： 使用 IGMPv2 1： 使用 IGMPv1 **SMB：UDP6 组播阻塞** 当 Sn_MR[3:0] 为 UDP6 或者 UDPD 时，使用 SMB 位。 它阻止接收发送到 W6100 中的请求组播地址的数据包 0： 关闭组播阻塞 1： 打开组播阻塞 **MMB：MACRAW 模式下 UDP4 的组播阻塞** 当 Sn_MR [3:0]为 MACRAW 模式并且 Sn_MR [MF] 为‘1’时有效 0： 不阻塞 1：IPv4 组播阻塞
4	UNIB/ MMB6	**UNIB： 单播阻塞** 当 Sn_MR[3:0] 为 UDP4、UDP6 或 UDPD 时有效 ***参考***）6.3.5.2 UDP 阻塞 0： 关闭 UDP 单播阻塞 1： 打开 UDP 单播阻塞 **MMB6：MACRAW 模式下 UDP6 的多播阻塞** 当 Sn_MR [3:0] 为 MACRAW 模式并且 Sn_MR [MF] 等于‘1’的时候有效 0： 不阻塞 1：IPv6 组播阻塞
[3:0]	P[3:0]	**P[3:0]协议模式选择** 设置 SOCKET 的工作模式， MACRAW 模式只能在 SOCKET 0 时使用

P[3:0]	协议模式
0000	SOCKET Closed
0001	TCP4
0010	UDP4
0011	IPRAW4
0111	MACRAW
1001	TCP6
1010	UDP6

			1011	IPRAW6
			1101	TCP Dual (TCPD)
			1110	UDP Dual (UDPD)

4.2.2 Sn_PSR (SOCKET n 首选源 IPv6 地址寄存器)

[RW] [0x0004] [0x00]

Sn_PSR 设置 SOCKET n 的源 IPv6 地址(SIP6) 。

值	符号	描述
0x00	AUTO	根据目标的 IPv6 地址（DIP6)、源 IPv6 地址（SIP6)自动设置。 如果 DIP6 是 LLA，则 SIP6 设置为 LLA。 如果 DIP6 是 GUA，则 SIP6 设置为 GUA
0x02	LLA	SIP6 设置为 LLA
0x03	GUA	SIP6 设置为 GUA

4.2.3 Sn_CR (SOCKET n 控制寄存器)

[RW,AC] [0x0010] [0x00]

Sn_CR 设置 SOCKET 的命令位，在命令执行完以后，命令位自动清零，在前一个命令清除之前不能执行下一个命令。

<table>
<tr><th>值</th><th>符号</th><th>描述</th></tr>
<tr><td>0x01</td><td>OPEN</td><td>OPEN 命令
在 SOCKET OPEN 命令之前，必须由 Sn_MR 设置 SOCKET 模式。在 OPEN 命令之后，Sn_SR 描述 SOCKET 状态如下：
<table>
<tr><th>Sn_MR (P[3:0])</th><th>Sn_SR</th></tr>
<tr><td>Sn_MR_CLOSE(‘0000’)</td><td>SOCK_CLOSED(0x00)</td></tr>
<tr><td>Sn_MR_TCP(‘0001’)</td><td>SOCK_INIT(0x13)</td></tr>
<tr><td>Sn_MR_UDP(‘0010’)</td><td>SOCK_UDP(0x22)</td></tr>
<tr><td>Sn_MR_IPRAW(‘0011’)</td><td>SOCK_IPRAW(0x32)</td></tr>
<tr><td>S0_MR_MACRAW(‘ 100’)</td><td>SOCK_MACRAW(0x42)</td></tr>
<tr><td>Sn_MR_TCP6(‘1001’)</td><td>SOCK_INIT(0x13)</td></tr>
<tr><td>Sn_MR_UDP6(‘1010’)</td><td>SOCK_UDP(0x22)</td></tr>
<tr><td>Sn_MR_IPRAW6(‘1011’)</td><td>SOCK_IPRAW6(0x33)</td></tr>
<tr><td>Sn_MR_TCPD(‘1101’)</td><td>SOCK_INIT(0x13)</td></tr>
<tr><td>Sn_MR_UDPD(‘1110’)</td><td>SOCK_UDP(0x22)</td></tr>
</table>
</td></tr>
</table>

0x02	LISTEN	**LISTEN 命令** 如果 SOCKET 模式为 TCP4，TCP6 或 TCPD 且 Sn_SR 为 SOCK_INIT，则在 LISTEN 命令后等待 TCP Client 连接 ***参考 6.2.1 TCP 服务器 ： 监听***
0x04	CONNECT	**TCP CONNECT 命令** 如果 SOCKET 模式是 TCP4 或 TCPD，并且 Sn_SR 是 SOCK_INIT，则在 CONNECT 命令后会向 TCP Server 发起连接请求 ***参考 6.2.2 TCP 客户端 ： 连接***
0x84	CONNECT6	**TCP6 CONNECT 命令** 如果 SOCKET 模式是 TCP6 或 TCPD，并且 Sn_SR 是 SOCK_INIT，则在 CONNECT 命令后会向 TCP Server 发起连接请求 ***参考 6.2.2 TCP 客户端: 连接***
0x08	DISCON	**TCP DISCON 命令** 如果 SOCKET 模式是 TCP4，TCP6 或 TCPD 并且 Sn_SR 是 SOCK_ ESTABLISHED 或 SOCK_CLOSE_WAIT 状态，则主动发出断开连接请求包（FIN 包） ***参考 6.2.1 TCP 服务器: 断开连接（主动断开）***
0x10	CLOSE	**CLOSE 命令** 执行 CLOSE 命令后，SOCKET 将立即关闭，无论之前处于什么状态，Sn_SR 都会更改为 SOCK_CLOSED ** 警告: 在 TCP4，TCP6 和 TCPD 模式下执行该命令，不会发送 FIN 包，SOCKET 将直接关闭!*
0x20	SEND	**SEND 命令** SOCKET 在 TCP4, TCP6, TCPD, UDP4, UDPD, IPRAW4 和 MACRAW 模式下发送数据包
0xA0	SEND6	**IPv6 SEND 命令** SOCKET 在 UDP6, UDPD 和 IPRAW6 模式下发送数据包
0x22	SEND_KEEP	**TCP SEND_KEEP 命令** SEND_KEEP 命令仅在 TCP4、TCP6 和 TCPD 模式下有效，并且在发送 SEND_KEEP 命令之前主机至少发送 1 字节数据后生效。 SEND_KEEP 命令将 Keep Alive（KA）包发送到对端，用来检查 TCP 连接是否有效。如果未接收到 KA 包的 ACK 包，则会触发 Sn_IR[TIMEOUT]，并在配置的重传输时间后将 Sn_SR 设置为 SOCK_CLOSED ***参考 0 6.2.4.2 Keep*** Alive
0x40	RECV	**SOCKET RECV 命令** 通过 RECV 命令，主机能够读取 SOCKET n 的 RX 缓存区接收到的数据，必须根据读取数据的大小增加 Sn_RX_RD(SOCKET n 读指针寄存器) ***参考 4.2.28 Sn_RX_RSR (SOCKET n 接收大小寄存器), 4.2.30 Sn_RX_WR (SOCKET n 接收写指针寄存器), 4.2.29 Sn_RX_RD (SOCKET n 接收读指针寄存***

		器)

* 通过执行 SEND 或 SEND6 命令，SOCKET 将数据发出，其数据长度由 Sn_TX_WR 和 Sn_TX_RD 确定。 发送的数据长度不得超过 Sn_TX_FSR，并且在检查 Sn_IR [SENDOK] ='1'后，方能执行下一个 SEND 命令
* 在 TCP4、TCP6、TCPD、UDP4、UDP6 和 UDPD 模式下，若发送的数据长度超过 MSS（最大传输单元），则数据需要根据 MSS 拆分发送
* 在 IPRAW4、IPRAW6 和 MACRAW 模式中，数据必须根据 MSS 拆分
* 在 TCP4，TCP6 和 TCPD 模式下，如果 SOCKET 发送数据包失败（未收到 ACK 包），SOCKET 将关闭，并且将会触发 Sn_IR [TIMEOUT]和 Sn_SR [SOCK_CLOSED]
* 在 TCP4、TCP6、TCPD、UDP4、UDP6、UDPD、IPRAW4、IPRAW6 和 MACRAW 模式下，Sn_TX_FSR 的值将在 Sn_IR [SENDOK] ='1'后自动增加

4.2.4 Sn_IR (SOCKET n 中断寄存器)

[RO] [0x0020] [0x00]

Sn_IR 指示了 SOCKET n 的状态或 Sn_CR 的结果。如果 Sn_IR 寄存器中的事件被触发，并且 Sn_IMR 中相应的屏蔽位被置位，则 SIR[Sn_INT]被置成‘1’。

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

位	符号	描述
[7:5]	-	保留位
4	SENDOK	**SEND OK 中断** Sn_CR[SEND]命令完成后该位置 1
3	TIMEOUT	**TIMEOUT 中断** 当重传次数超过 ARP/ND 或 TCP 通信中的 Sn_RCR (SOCKERT n 重传次数寄存器)设置的值时，该位置 1
2	RECV	**RECEIVED 中断** 当 SOCKET 接收到数据，或执行 Sn_CR[RECV]后 SOCKET n 的缓存区仍有数据时，该位置 1
1	DISCON	DISCONNECTED 中断 当收到 FIN 包或 RST 包，或者收到执行 Sn_CR[DISCON]发送的 FIN 包后收到的 ACK 包时，该位置 1
0	CON	**CONNECTED 中断** 当执行 Sn_CR[CONNECT]或 Sn_CR[CONNECT6]命令成功建立 TCP 连接，或者接收到目的 IP 地址发来的 SYN 包时，该位置 1

4.2.5 Sn_IMR (SOCKET n 中断屏蔽寄存器)

[R=W] [0x0024] [0xFF]

Sn_IMR 用来屏蔽 Sn_IR 寄存器的相应位。

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]	-	保留位
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

4.2.6 Sn_IRCLR (Sn_IR 清除寄存器)

[WO] [0x0028] [0xFF]

Sn_IRCLR 用来清除 Sn_IR 寄存器的相应位。

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

4.2.7 Sn_SR (SOCKET n 状态寄存器)

[RO] [0x0030] [0x00]

Sn_SR 指示了 SOCKET n 的状态，SOCKET n 的状态根据 SOCKET n 命令或者数据的收/发而变化。

值	符号	描述
0x00	SOCK_CLOSED	SOCKET n 关闭
0x13	SOCK_INIT	TCP 模式下打开 SOCKET n
0x14	SOCK_LISTEN	TCP Server 模式下 SOCKET n 等待对端的连接请求
0x17	SOCK_ESTABLISHED	SOCKET n 的 TC 连接建立
0x1C	SOCK_CLOSE_WAIT	TCP 模式下 SOCKET n 接收到对端的 FIN 包
0x22	SOCK_UDP	UDP 模式下打开 SOCKET n
0x32	SOCK_IPRAW	IPRAW 模式下打开 SOCKET n
0x33	SOCK_IPRAW6	IPRAW6 模式下打开 SOCKET n
0x42	SOCK_MACRAW	MACRAW 模式下打开 SOCKET n

下表列出了 Socket n 的所有临时状态。

值	符号	描述
0x15	SOCK_SYNSENT	SOCKET 发送连接请求时的临时状态
0x16	SOCK_SYNRECV	SOCKET 收到连接请求时的临时状态
0x18	SOCK_FIN_WAIT	SOCKET n 关闭过程中的临时状态
0X1B	SOCK_TIME_WAIT	
0X1D	SOCK_LAST_ACK	

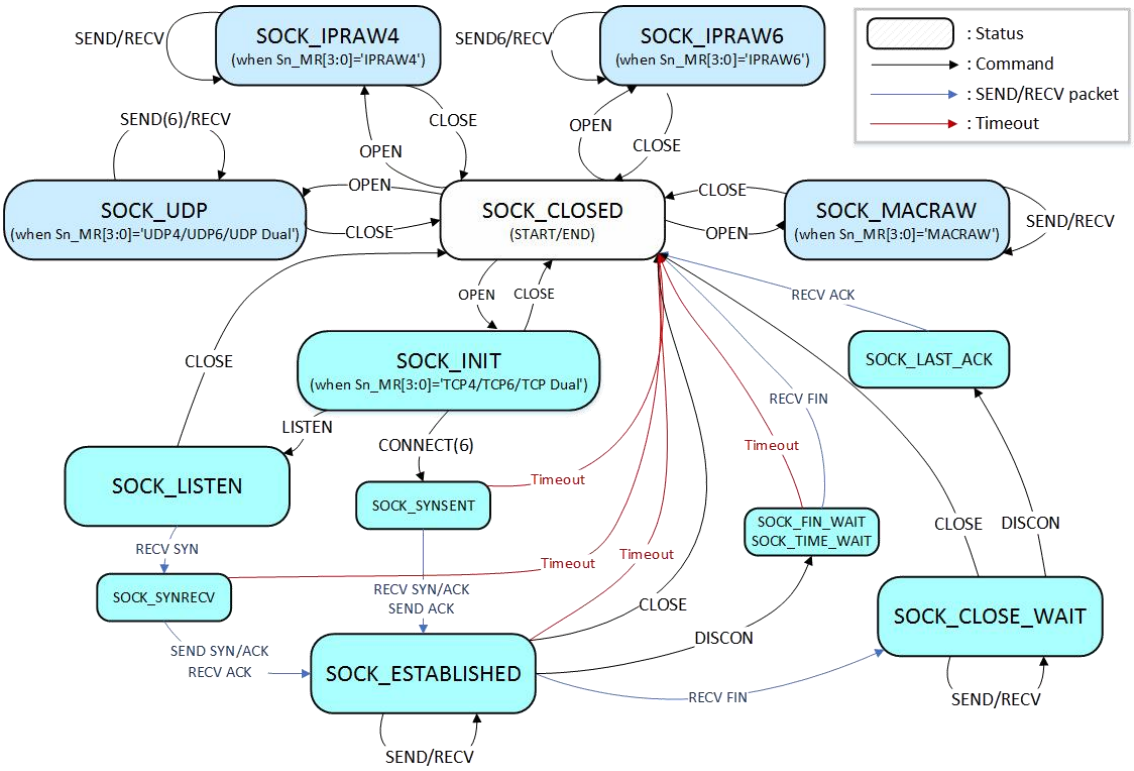


图 4 W6100 SOCKET 状态转化图

4.2.8 Sn_ESR (SOCKET n 扩展状态寄存器)

[RO] [0x0031] [0x00]

Sn_ESR 为 TCP4、TCP6 和 TCPD 模式下的 SOCKET n 扩展状态。

7	6	5	4	3	2	1	0
-	-	-	-	-	TCPM	SVR	GUA
-	-	-	-	-	RO	RO	RO

位	符号	描述
[7:3]	-	保留位
2	TCPM	TCP 协议版本 指示了在 TCPD 模式下与目标 IP 地址建立连接所使用的 TCP 协议版本 0：TCP4 1：TCP6
1	TCPOP	TCP 工作模式

		0：TCP Client 1：TCP Server
0	IP6T	IPv6 地址类型 指示了在 TCP6 和 TCPD 模式下发送数据包的源 IPv6 地址类型 0：LLA 1：GUA

4.2.9 Sn_PNR (SOCKET n IP 协议号寄存器)

[R=W] [0x0100] [0x0000]

在 IPRAW4 和 IPRAW6 模式下，Sn_PNR 设置 IPv4 的上层协议号或 IPv6 的下一个头。

请参考表 7 和 IANA_Protocol Numbers 来设置 Sn_PNR。

注：不能设置为 TCP（0x06）和 UDP（0x11）。

4.2.10 Sn_TOSR (SOCKET n IP 服务类型寄存器)

[R=W] [0x0104] [0x00]

Sn_TOSR 设置 IPv4 头部中的 TOS (服务类型)。

参考）IANA_IP Parameters

** 警告 W6100 不支持 IPv6 报头中的 Traffic Class 和 Flow Label Field，这两部分被设置为 '0'。*

4.2.11 Sn_TTLR (SOCKET n IP 生存时间寄存器)

[R=W] [0x0108] [0x80]

Sn_TTLR 设置 IPv4 头的 TTL(生存时间)，或 IPv6 头部中的跳数。

参考）IANA_IP_Parameters

4.2.12 Sn_FRGR (SOCKET n IP 头 Fragment 偏移量寄存器)

[R=W] [0x010C-0x010D] [0x4000]

Sn_FRGR 设置 IP 头中的 Fragment 偏移量。

**** 警告 Fragment 字段可以设置为任何值，但是 W6100 的 SOCKET 不执行分段也不处理任何接收到的分段数据包。***

例如： S0_FRGR0 = 0x0000 (不分片)

S0_FRGR0(0x010C)	S0_FRGR1(0x010D)
0x00	0x00

4.2.13 Sn_MSSR (SOCKET n 最大分段寄存器)

[R=W] [0x0110-0x0111] [0xFFFF]

Sn_MSSR 设置 SOCKET n 的 MSS (最大传输单元) ，并且必须在 Sn_CR[OPEN]之前设置。每一个 SOCKET 的工作模式都有一个 MSS 范围，如果由 Sn_MSSR 设置的 SOCKET n 传输单元值超过了 MSS 的范围，则自动设置为当前 MSS 范围内最大传输单元值。

Sn_MR[3:0]	正常模式范围 (NETMR2[PPPoE]='0')	PPPoE 模式范围 (NETMR2[PPPoE]='1')

TCP	1~1460	1~1452
TCP6	1~1440	1~1432
UDP	1~1472	1~1464
UDP6	1~1452	1~1444
IPRAW	1~1480	1~1472
IPRAW6	1~1460	1~1452
MACRAW	1~1514	

例如： S0_MSSR = 1460 (0x05B4)

S0_MSSR0(0x0110)	S0_MSSR1(0x0111)
0x05	0xB4

4.2.14 Sn_PORTR (SOCKET n 源端口寄存器)

[R=W] [0x0114-0x0115] [0x0000]

Sn_PORTR 设置 SOCKET n 的源端口号。

例如： S0_PORTR = 5000 (0x1388)

S0_PORTR0(0x0114)	S0_PORTR1(0x0115)
0x013	0x88

4.2.15 Sn_DHAR (SOCKET n 目的 MAC 地址寄存器)

[RW] [0x0118-0x11D] [0x0000_0000_0000]

Sn_DHAR 设置 TCP 连接建立成功后目的 MAC 地址，此时目标设备应该处于 TCP4、TCP6 或 TCPD 模式。

当 Sn_MR [3:0]为 UDP4 或 UDP6 且 Sn_MR [MULTI]为 “1”时，Sn_DHAR 被设置为组播组 MAC 地址。

参考） 6.3.3 UDP 组播

例如： S0_DHAR = “11:22:33:AA:BB:CC”

S0_DHAR0(0x0118)	S0_DHAR1(0x0119)	S0_DHAR2(0x011A)
0x11	0x22	0x33
S0_DHAR3(0x011B)	S0_DHAR4(0x011C)	S0_DHAR5(0x011D)
0xAA	0xBB	0xCC

4.2.16 Sn_DIPR (SOCKET n IPv4 目的 IP 地址寄存器)

[RW] [0x0120-0x0123] [0x0000_0000]

Sn_DIPR 设置 IPv4 的目的 IP 地址，该值取决于 Sn_MR[3:0]中的协议类型。

Sn_MR[3:0]	Sn_MR[MULTI]	Sn_DIPR
TCP4	-	设置或获取 IPv4 目的 IP 地址

TCPD	-	
UDP4	0	设置 IPv4 目标 IP 地址
UDPD	-	
IPRAW4	-	
UDP4	1	设置 IPv4 组播组 IP 地址

参考) 6.3.3 UDP 组播

例如： S0_DIPR = “192.168.0.11”

S0_DIPR0(0x0120)	S0_DIPR1(0x0121)	S0_DIPR2(0x0122)	S0_DIPR3(0x0123)
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

4.2.17 Sn_DIP6R (SOCKET n IPv6 目的 IP 地址寄存器)

[RW] [0x0130-0x013F] [0x0000_0000_0000_0000_0000_0000_0000_0000]

Sn_DIPR 设置 IPv6 目的 IP 地址，该值取决于 Sn_MR[3:0]中的协议类型。

Sn_MR[3:0]	Sn_MR[MULTI]	Sn_DIP6R
TCP6	-	设置或获取 IPv6 目的 IP 地址
TCPD	-	
UDP6	0	设置 IPv6 目的 IP 地址
UDPD	-	
IPRAW6	-	
UDP6	1	设置 IPv6 组播组的 IP 地址

参考) 6.3.3 UDP 组播

例如： 目的 IP 地址是“FE80::AB:CDEF”

S0_DIP6R0(0x0130)	S0_DIP6R1(0x0131)	S0_DIP6R2(0x0132)	S0_DIP6R3(0x0133)
0xFE	0x80	0x00	0x00
S0_DIP6R4(0x0134)	S0_DIP6R5(0x0135)	S0_DIP6R6(0x0136)	S0_DIP6R7(0x0137)
0x00	0x00	0x00	0x00
S0_DIP6R8(0x0138)	S0_DIP6R9(0x0139)	S0_DIP6R10(0x013A)	S0_DIP6R11(0x013B)
0x00	0x00	0x00	0x00
S0_DIP6R12(0x013C)	S0_DIP6R13(0x013D)	S0_DIP6R14(0x013E)	S0_DIP6R15(0x013F)
0x00	0xAB	0xCD	0xEF

4.2.18 Sn_DPORTR (SOCKET n 目的端口寄存器)

[R=W] [0x0140-0x0141] [0x0000]

Sn_DPORTR 设置目标端口号，该值取决于 Sn_MR[3:0]中的协议类型。

Sn_MR[3:0]	Sn_MR[MULTI]	Sn_DPORTR
TCP4	-	设置或者获取目的端口号
TCP6	-	
TCPD	-	

UDP4	0	设置目的端口号
UDP6	0	
UDPD	-	
IPRAW4	-	
IPRAW6	-	
UDP4	1	设置组播组端口号
UDP6	1	

在 TCP4、TCP6 和 TCPD 模式下，Sn_DPORTR 设置为目标端口或获取已连接的目标端口。

在 UDP4、UDP6、UDPD 和 IPRAW6 模式下，Sn_DPORTR 设置为对等方的目标端口。

在 UDP4 和 UDP6 组播模式下，Sn_DPORTR 设置为组播组端口。

参考) 6.3.3 UDP 组播

例如： S0_DPORTR = 5000 (0x1388)

S0_DPORTR0(0x0140)	S0_DPORTR1(0x0141)
0x13	0x88

4.2.19 Sn_MR2 (SOCKET n 模式寄存器 2)

[R=W] [0x0144] [0x00]

Sn_MR2 设置 SOCKET n 的其他选项。

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

位	符号	描述
[7:2]	-	保留位
1	DHAM	**目的 MAC 地址** 当 Sn_MR[3:0]不是 MACRAW 模式时，它设置要发送的以太网帧的目标 MAC 地址。 0：目的 MAC 地址设置为通过 ARP 获得的地址 1：目的 MAC 地址设置为 Sn_DHAR 中的值
0	FARP	**强制 ARP** 当 Sn_MR 为 UDP4，UDP6，UDPD，IPRAW4 和 IPRAW6 模式时，SOCKET n 对每个 Sn_CR [SEND]或 Sn_CR [SEND6]执行 ARP/ND 处理。 当 Sn_MR 为 TCP4，TCP6，TCPD 模式且处于“TCP SERVER”时，在发送 SYN/ ACK 数据包之前执行 ARP/ND 处理。 ARP-process 获取的目的 MAC 地址用作待发送报文的目的 MAC 地址。 0：关闭 1：打开 * ***警告*** *在 DHAM ='1'的情况下，即使执行 ARP/ND，目的 MAC 地址也设置为 Sn_DHAR*

4.2.20 Sn_RTR (SOCKET n 重传时间寄存器)

[R=W] [0x0180-0x0181] [0x0000]

Sn_RTR 设置 SOCKET n 重传时间，单位为 100us。如果 Sn_RTR 为'0'，则由 Sn_CR [OPEN] ='1'初始化，其值为 RTR。

请参考 *6.7 重传*

例如： S0_RTR = 5000 (0x1388)

5000 * 100us = 0.5s

S0_RTR0(x0180)	S0_RTR1(0x0181)
0x013	0x88

4.2.21 Sn_RCR (SOCKET n 重传次数寄存器)

[R=W] [0x0184] [0x00]

Sn_RCR 设置 SOCKET n 重传次数。如果 Sn_RCR 为'0'，则由 Sn_CR [OPEN] = '1'初始化，其值为 RCR。

请参考 *6.7 重传*

4.2.22 Sn_KPALVTR (SOCKET n 心跳检测)

[R=W] [0x0188] [0x00]

Sn_KPALVTR 设置 Keep Alive（KA）数据包传输时间间隔，单位为 5 秒。TCP 模式下 SOCKET 每隔 Sn_KPALVTR 就会发送 KA 数据包。

在发送 KA 数据包之前，SOCKET 必须至少发送一次（超过 1 字节）数据包。通过 Sn_CR [SENDKEEP]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_KPALVTR(0x0188)
0x0A

4.2.23 Sn_TX_BSR (SOCKET n 发送缓存大小寄存器)

[R=W] [0x0200] [0x02]

Sn_TX_BSR 配置 SOCKET n 的发送缓存大小，可以配置为 0、1、2、4、8 或 16KB。

每个寄存器的发送缓存按从 SOCKET 0 到 SOCKET 7 顺序分配，8 个 SOCKET 发送缓存的大小总和不能超过 16KB，否则可能导致缓存区的读写访问出现故障。

Value (Dec)	0	1	2	4	8	16
Buffer size	0KB	1KB	2KB	4KB	8KB	16KB

例如： S0_TX_BSR= 4 Kbytes

S0_TX_BSR(0x0200)

0x04

4.2.24 Sn_TX_FSR (SOCKET n 空闲发送缓存寄存器)

[RO] [0x0204-0x205] [0x0000]

Sn_TX_FSR 指示了 SOCKET n 发送缓存空闲大小。

在UDP、IPRAW和MACRAW模式下：

Sn_TX_FSR = Sn_TX_BSR - | Sn_TX_WR(1) - Sn_TX_RD(2) |

在TCP模式下：

Sn_TX_FSR = Sn_TX_BSR - | Sn_TX_WR - Internal Pointer(3) |

(1) SOCKET n TX写指针寄存器
(2) SOCKET n TX读指针寄存器
(3) 由W6100管理的TCP ACK指针

必须确保要发送的数据大小不超过 Sn_TX_FSR 的值。

例如： S0_TX_FSR = 1024 (0x0400)

S0_TX_FSR0(0x0204)	S0_TX_FSR1(0x0205)
0x04	0x00

4.2.25 Sn_TX_RD (SOCKET n 发送读指针寄存器)

[RO] [0x0208-0x0209] [0x0000]

Sn_TX_RD 由 Sn_CR[OPEN]初始化。通过 Sn_CR[SEND]命令，SOCKET 将发送缓存中处于 Sn_TX_RD 和 Sn_TX_WR 之间的数据发送出去。数据发送完成后，Sn_IR[SENDOK]置位并且 Sn_TX_RD 自增（自增大小为发送数据的大小）。当 Sn_TX_RD 增加的值超出最大值 0xFFFF（大于 0x10000 并产生进位），Sn_TX_RD 会忽略进位，仅使用低 16 位的值。

例如： S0_TX_RD = 0xd4b3

S0_TX_RD0(0x0208)	S0_TX_RD1(0x0209)
0xd4	0xb3

4.2.26 Sn_TX_WR (SOCKET n 发送写指针寄存器)

[RW] [0x020C-0x20D] [0x0000]

Sn_TX_WR 由 Sn_CR[OPEN]命令初始化。需要将按以下步骤配置 Sn_TX_WR 才能发送数据：

(1) 主机读取发送缓存中将要保存传输数据的首地址；
(2) 从 SOCKET n 的发送缓存对应的首地址开始保存需要传输的数据；
(3) 主机根据发送数据的大小增加 Sn_TX_WR，如果 Sn_TX_RD 超出最大值 0xFFFF（大于 0x10000 并产生进位），Sn_TX_RD 会忽略进位，仅使用低 16 位的值；
(4) 主机通过 Sn_CR[SEND]命令发送保存在 SOCKET n 发送缓存区的数据

例如： S0_TX_WR = 0x0800

S0_TX_WR0(0x020C)	S0_TX_WR1(0x020D)
0x08	0x00

4.2.27 Sn_RX_BSR (SOCKET n 接收缓存大小寄存器)

[R=W] [0x0220] [0x02]

Sn_ RX_BSR 配置 SOCKET n 的接收缓存大小，可以配置为 0、1、2、4、8 或 16KB。

每个寄存器的接收缓存按从 SOCKET 0 到 SOCKET 7 顺序分配，8 个 SOCKET 接收缓存的大小总和不能超过 16KB，否则可能导致缓存区的读写访问出现故障。

Value (Dec)	0	1	2	4	8	16
Buffer size	0KB	1KB	2KB	4KB	8KB	16KB

例如： S0_RX_BSR = 8 Kbytes

S0_RX_BSR(0x0220)
0x08

4.2.28 Sn_RX_RSR (SOCKET n 接收大小寄存器)

[RO] [0x0224-0x0225] [0x0000]

Sn_RX_RSR 表示在 SOCKET n 接收缓存区中已接收的数据大小。

在TCP、UDP、IPRAW和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(0x0224)	S0_RX_RSR1(0x0225)
0x08	0x00

4.2.29 Sn_RX_RD (SOCKET n 接收读指针寄存器)

[RW] [0x0228-0x229] [0x0000]

Sn_Rx_RD 可通过 Sn_CR[OPEN]初始化。SOCKET n 接收缓存中接收到的数据按照以下步骤读取或更新：

(1) 主机读取保存在接收缓存中数据的首地址；
(2) 从保存在 SOCKET n 接收缓存中的数据的首地址开始读取数据；
(3) 主机根据接收数据的大小更新 Sn_RX_RD 的值。如果 Sn_RX_RD 超出最大值 0xFFFF（大于 0x10000 并产生进位），Sn_RX_RD 会忽略进位，仅使用低 16 位的值；
(4) 主机设置 Sn_CR[RECV]并释放 SOCKET n 接收缓存

例如： S0_RX_RD =1536(0x0600)

S0_RX_RD0(0x0228)	S0_RX_RD1(0x0229)
0x06	0x00

4.2.30 Sn_RX_WR (SOCKET n 接收写指针寄存器)

[RO] [0x022C-0x022D] [0x0000]

Sn_RX_WR 表示 SOCKET n TX 缓存区中接收数据的最后地址，Sn_RX_WR 由 Sn_CR[OPEN] 命令初始化并根据接收数据的大小自动增加。如果递增的 Sn_RX_WR 超过 16 位偏移地址的最大值 0xFFFF（大于 0x10000 并产生进位），则忽略进位并自动设置为较低的 16 位值。

*例如：*S0_RX_WR = 1536(0x0600)

S0_RW_WR0(0x022C)	S0_RW_WR1(0x022D)
0x06	0x00

5 主机接口模式

通过设置 MODE[3:0]，W6100 可以选择与主机进行 SPI 或并行总线通信。SPI 接口由 CSn、SCLK、MOSI 和 MISO 组成，并行总线接口由控制信号线(CSn、WRn、RDn、INTn)、地址线(2 位)和数据线(8 位)组成。

5.1 SPI 工作模式

当 MODE[3:0]设置为“000X”，启用 SPI 模式，W6100 作为 SPI 从机运行。W6100 与主机通过 SPI 接口通信有 2 种接线方式，如图 5 和图 6 所示。图 5 的方式 W6100 可以与其他 SPI 从机共用 SPI 接口，即 VDM（可变长度模式），该模式 CSn 由主机控制。图 6 的方式 W6100 不与其他 SPI 从机共用 SPI 接口，即 FDM（固定长度模式），该模式 CSn 接地。

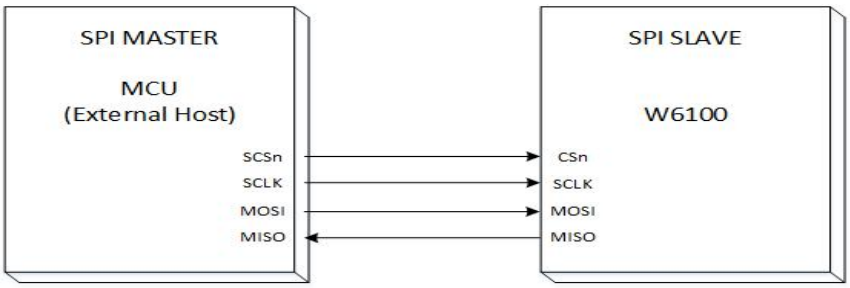


图 5 VDM 可变数据长度模式

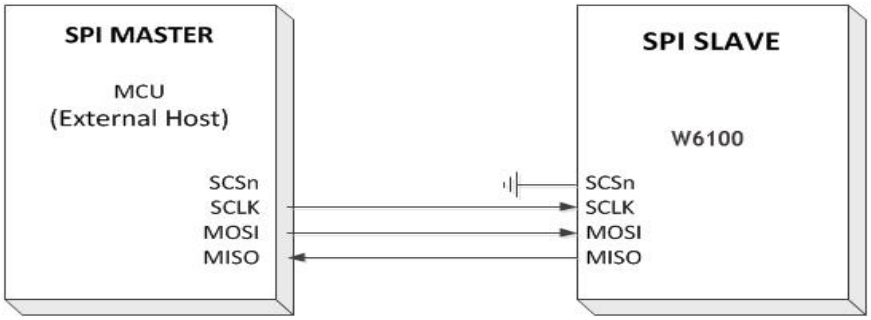


图 6 FDM 固定数据长度模式

W6100 支持 SPI 模式 0/3，如图 1 所示。数据始终在 SCLK 上升沿采样，并在 SCLK 的下降沿切换。MOSI 和 MISO 信号始终按每个 SCLK 从 MSB 到 LSB 依次发送或接收。

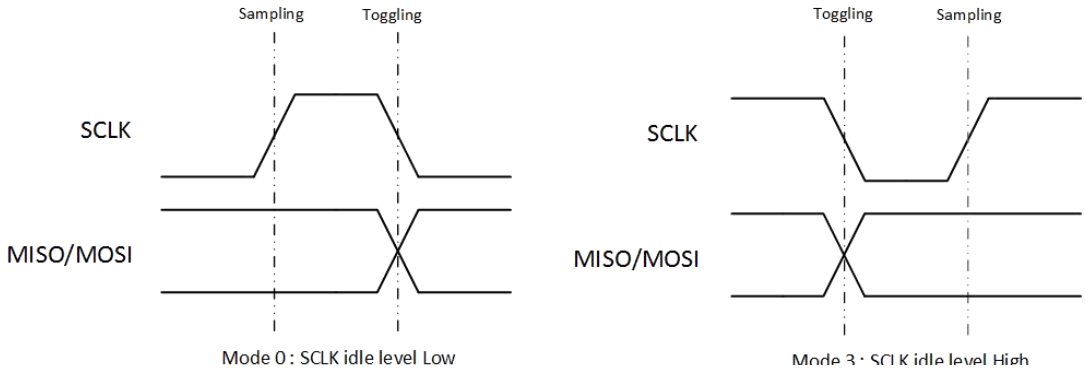


图 7 SPI 模式 0 和模式 3

5.1.1 SPI 数据帧

W6100 与主机的 SPI 通讯按照 SPI 数据帧进行。SPI 帧由地址段、控制段和数据段组成，如下图 8 所示。

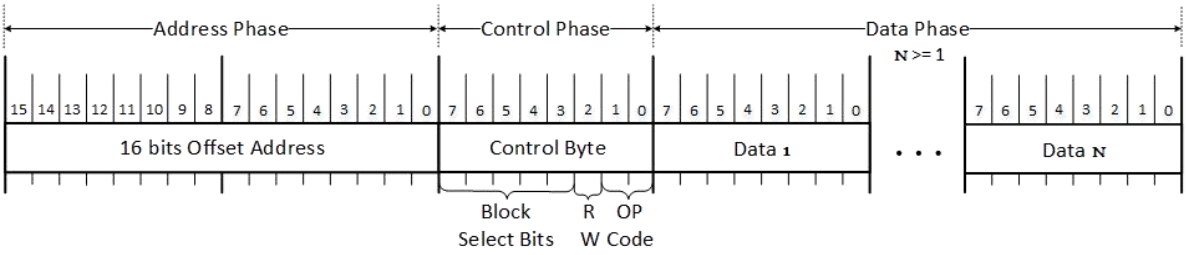


图 8 SPI 数据帧格式

地址段表示 W6100 寄存器或 TX/RX 缓冲区的 16 位偏移地址。控制段指示了访问哪个区块、R/W 访问模式和 SPI 工作模式（VDM、FDM）。数据段可以设定为任意长度（N 表示字节长度，1≤N）或者是固定的长度：1 字节、2 字节或 4 字节。

当 SPI 工作模式为可变数据长度模式（VDM）时，CSn 信号必须由主机控制。在 VDM 中，SPI 帧由主机的 CSn 从高电平到低电平开始，从低电平到高电平结束。简而言之，VDM 由 CSn 控制，长度不受限制。

在 FDM 中，CSn 固定为 0，每帧只能发送 1,2,4 字节数据。

5.1.1.1 地址段

地址段表示 W6100 通用寄存器、SOCKET 寄存器和 SOCKET n TX/RX 缓冲区的 16 位偏移地址。16 位偏移地址按顺序从 MSB 传输到 LSB。

W6100 SPI 接口支持顺序数据读/写，自动对地址进行偏移，每读取或写入 1 字节后增加 1。

6.4.1.1 控制段

控制段包括了地址段中的偏移地址所属的区块、R/W 访问模式（RWB）和 SPI 工作模式（OM[1:0]）。

7	6	5	4	3	2	1	0
BSB4	BSB3	BSB2	BSB1	BSB0	RWB	OM1	OM0

<table>
<tr><th>位</th><th>符号</th><th>描述</th></tr>
<tr><td>7~3</td><td>BSB</td><td>区块选择位
W6100 每个 SOCKET 有一个通用寄存器、8 个 SOCKET n 寄存器和 TX/RX 缓冲区，该区域由 BSB[4:0]选择，如下表所示。
最高位的三位表示从 0 到 7 的 SOCKET。
最低位的两位表示 SOCKET 区块中的子区块类型
<table>
<tr><th>BSB[4:2]</th><th>BSB[1:0]</th><th>区块描述</th></tr>
<tr><td rowspan="4">000</td><td>00</td><td>通用寄存器区块</td></tr>
<tr><td>01</td><td>SOCKET 0 寄存器区块</td></tr>
<tr><td>10</td><td>SOCKET 0 TX 缓冲区区块</td></tr>
<tr><td>11</td><td>SOCKET 0 RX 缓冲区区块</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td rowspan="4">001</td><td>00</td><td>保留</td></tr>
<tr><td>01</td><td>SOCKET 1 寄存器区块</td></tr>
<tr><td>10</td><td>SOCKET 1 TX 缓冲区区块</td></tr>
<tr><td>11</td><td>SOCKET 1 RX 缓冲区区块</td></tr>
<tr><td rowspan="4">010</td><td>00</td><td>保留</td></tr>
<tr><td>01</td><td>SOCKET 2 寄存器区块</td></tr>
<tr><td>10</td><td>SOCKET 2 TX 缓冲区区块</td></tr>
<tr><td>11</td><td>SOCKET 2 RX 缓冲区区块</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td rowspan="4">111</td><td>00</td><td>保留</td></tr>
<tr><td>01</td><td>SOCKET 7 寄存器区块</td></tr>
<tr><td>10</td><td>SOCKET 7 TX 缓冲区区块</td></tr>
<tr><td>11</td><td>SOCKET 7 RX 缓冲区区块</td></tr>
</table>
例如，如果指定 SOCKET 2 寄存器，则 BSB[4:2]=’010’和 BSB[1:0]=’01’
</td></tr>
<tr><td>2</td><td>RWB</td><td>读/写访问模式选择位
该位设置 SPI 访问模式
0： 读
1： 写</td></tr>
<tr><td>1~0</td><td>OM[1:0]</td><td>SPI 工作模式选择位
<table>
<tr><th>OM[1:0]</th><th>模式</th></tr>
<tr><td>00</td><td>VDM, N 字节数据段 (1 ≤ N)</td></tr>
<tr><td>01</td><td>FDM, 1 字节数据段</td></tr>
<tr><td>10</td><td>FDM, 2 字节数据段</td></tr>
<tr><td>11</td><td>FDM, 4 字节数据段</td></tr>
</table>
可变数据长度模式（VDM）
数据长度 N 通过 CSn 电平控制。
当主机将 CSn 拉低（从高电平到低电平），将立即通知 W6100 SPI 数据帧地址段的起始地址，此时控制段中的 OM[1:0] = ' 00 '。传输 N 字节数据后，主机将 CSn 拉高（从低到高），通知 W6100 SPI 数据帧结束

固定数据长度模式（FDM）
在 FDM 模式下，CSn 管脚接 GND，数据长度由 OM[1:0]指定</td></tr>
</table>

6.4.1.2 数据段

数据段的长度由控制段的 SPI 工作模式选择位(OM[1:0])设置，数据通过 MOSI 或 MISO 信号依次从 MSB 传输到 LSB。

5.1.2 可变数据长度模式（VDM）

在该模式下，SPI 数据帧的数据段长度由主机控制的 CSn 决定。对于 VDM 模式，控制段的 OM [1:0]必须设置为“00”。

5.1.2.1 写访问—VDM 模式

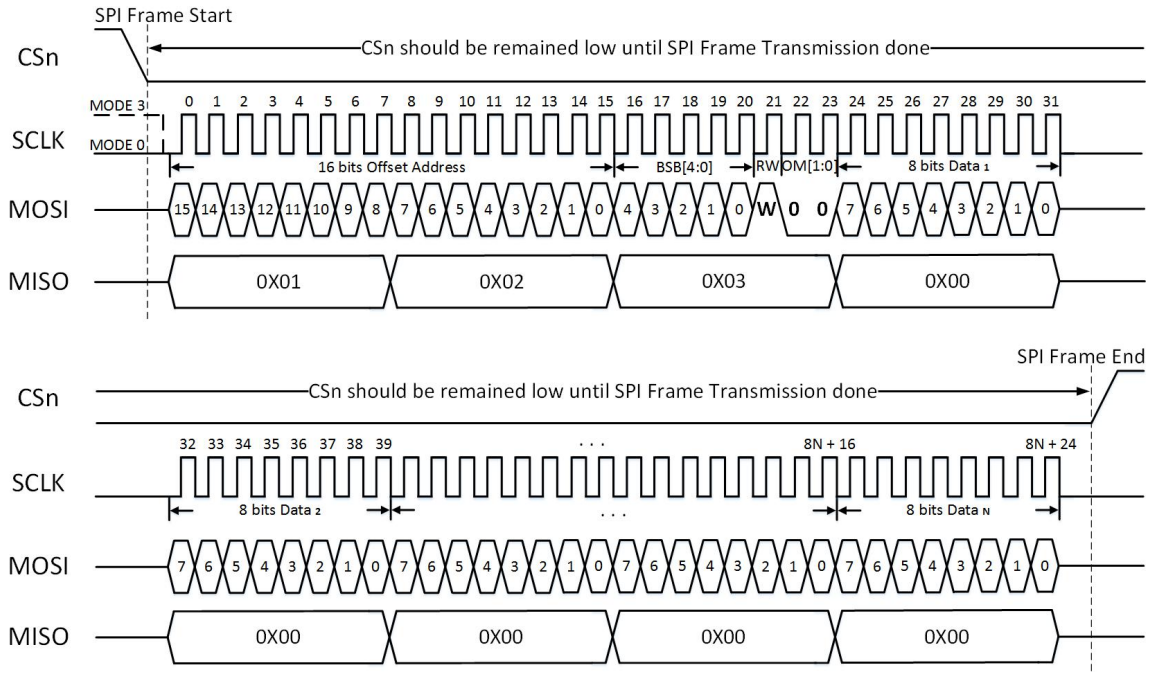


图 9 VDM 模式下的写访问

图 9 显示了写访问中的 SPI 数据帧及时钟。在 VDM 模式下，主机控制 CSn 从高到低通知 SPI 数据帧的开始，CSn 从低到高通知 SPI 数据帧的结束。控制段中，RWB 为“1”表示写访问，OM[1:0]为“00”表示 VDM 模式。通过 MOSI 传输的数据位与 SCLK（下降沿）同步。如果连续传输多个字节的数据，则支持顺序数据写入。

5.1.2.2 读访问—VDW 模式

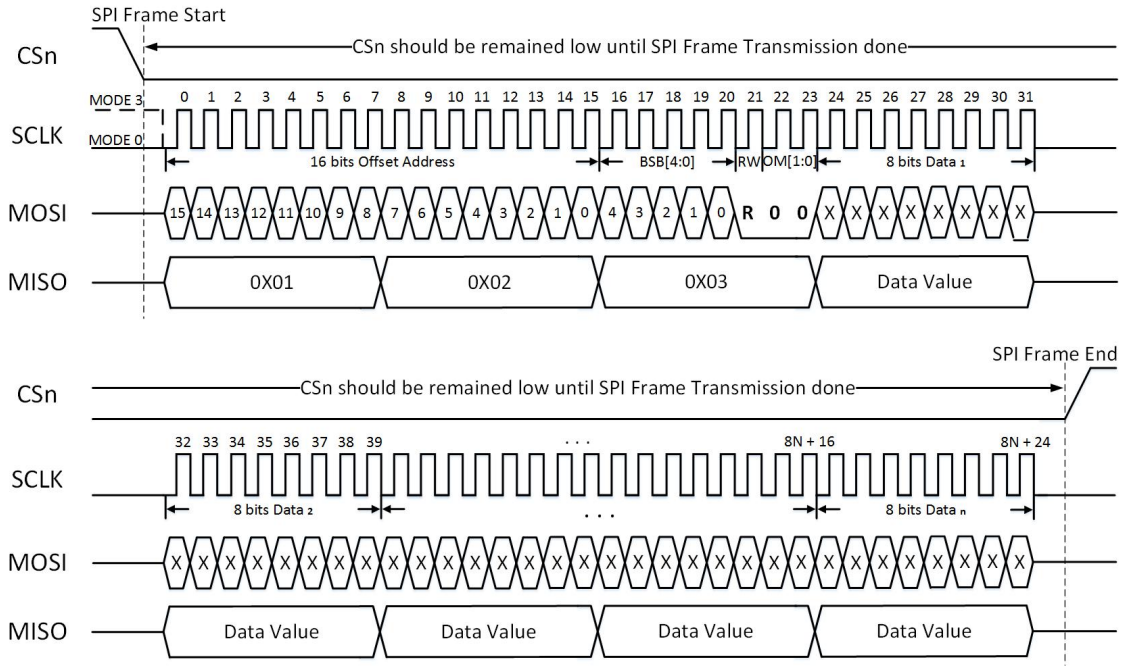


图 10 VDM 模式下的读访问

图 10 显示了读访问中的 SPI 数据帧及时钟。在 VDM 中，主机控制 CSn 从高到低通知 SPI 数据帧开始，CSn 从低到高通知 SPI 帧结束。控制段中，RWB 为“0”表示读访问，OM[1:0]为“00”表示 VDM 模式。通过 MISO 接收的数据位与 SCLK（下降沿）同步。如果连续传输多个字节的数据，则支持顺序数据写入。

5.1.3 固定数据长度模式（FDM）

FDM 模式下，CSn 接 GND，控制段的数据长度由 SPI 操作模式位(OM[1:0])选择，当 OM[1:0]和传输数据长度不同时，W6100 可能会出现异常。FMD 波形参考：5.1.2 可变数据长度模式

5.1.2.3 写访问—FDM 模式

1 字节写访问

Address Phase																Control Phase								Data Phase							
																BSB					RWB	OM		1st Data							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	R/W	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**1**	**0**	**1**	x	x	x	x	x	x	x	x

图 11 FDM 模式下，1 字节写访问的 SPI 数据帧

2 字节写访问

Address Phase																Control Phase								Data Phase							
																BSB					RWB	OM		1st Data							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	R/W	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**1**	**1**	**0**	x	x	x	x	x	x	x	x

Data Phase							
2nd Data							
7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

图 12 FDM 模式下，2 字节写访问的 SPI 数据帧

4 字节写访问

Address Phase																Control Phase								Data Phase							
																BSB					RWB	OM		1st Data							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	R/W	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**1**	**1**	**1**	x	x	x	x	x	x	x	x

Data Phase								Data Phase								Data Phase							
2nd Data								3rd Data								4th Data							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

图 13 FDM 模式下，4 字节写访问的 SPI 数据帧

5.1.2.4 读访问—FDM 模式

1 字节读访问

Address Phase																Control Phase								Data Phase							
																BSB					RWB	OM		1st Data							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	R/W	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**0**	**0**	**1**								
																								x	x	x	x	x	x	x	x

图 2 在 FDM 模式下，1 字节访问 SPI 数据帧

2 字节读访问

Address Phase																Control Phase								Data Phase							
																BSB					RWB	OM		1st Data							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	R/W	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**0**	**1**	**0**								
																								x	x	x	x	x	x	x	x

Data Phase							
2nd Data							
7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

图 3 在 FDM 模式下，2 字节读访问 SPI 数据帧

4 字节读访问

Address Phase																Control Phase								Data Phase							
																BSB					RWB	OM		1st Data							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	R/W	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	**0**	**1**	**1**								
																								x	x	x	x	x	x	x	x

Data Phase								Data Phase								Data Phase							
2nd Data								3rd Data								4th Data							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

图 4 在 FDM 模式下，4 字节读访问 SPI 数据帧

5.2 并行总线模式

如果引脚 MODE[3:0]设置为“010X”，启用并行总线模式。主机和 W6100 的接线如下图 17 所示。

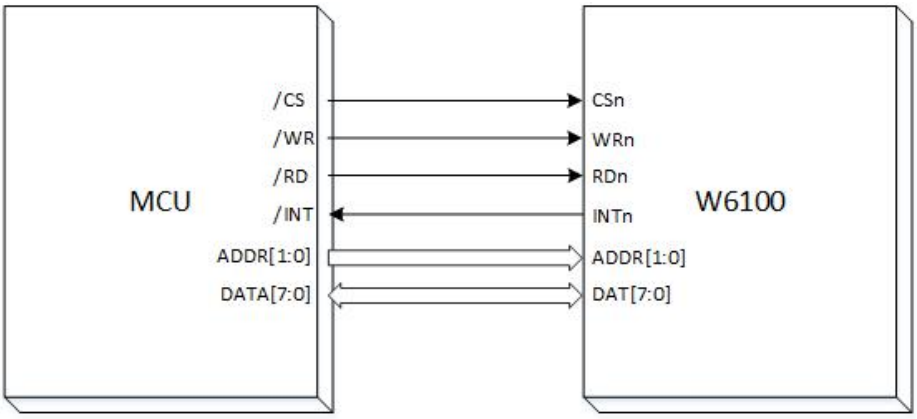


图 14 并行总线模式接线示意图

在并行总线模式下，主机可以通过 ADDR[1:0]、DAT[7:0]、CSn、RDn、WRn 等总线控制信号访问以下寄存器。与 SPI 帧一样，主机可以通过这些寄存器间接读/写 W6100 寄存器。

表 1 并行模式地址值

<table>
<tr><th>ADDR[1:0]</th><th>符号</th><th>描述</th></tr>
<tr><td>00</td><td>IDM_ARH</td><td>间接模式高地址寄存器
该位是 16 位偏移地址的最高标志位</td></tr>
<tr><td>01</td><td>IDM_ARL</td><td>间接模式低地址寄存器
该位是 16 位偏移地址的最低标志位</td></tr>
<tr><td>10</td><td>IDM_BSR</td><td>间接模式区域选择寄存器
它可以选择下面的区域
<table>
<tr><th>[7:5]</th><th>[4:3]</th><th>[2:0]</th><th>描述</th></tr>
<tr><td rowspan="4">000</td><td>00</td><td rowspan="12">保留</td><td>通用寄存器</td></tr>
<tr><td>01</td><td>SOCKET 0 寄存器</td></tr>
<tr><td>10</td><td>SOCKET 0 TX 缓冲区</td></tr>
<tr><td>11</td><td>SOCKET 0 RX 缓冲区</td></tr>
<tr><td rowspan="4">001</td><td>00</td><td>保留</td></tr>
<tr><td>01</td><td>SOCKET 1 寄存器</td></tr>
<tr><td>10</td><td>SOCKET 1 TX 缓冲区</td></tr>
<tr><td>11</td><td>SOCKET 1 RX 缓冲区</td></tr>
<tr><td rowspan="4">010</td><td>00</td><td>保留</td></tr>
<tr><td>01</td><td>SOCKET 2 寄存器</td></tr>
<tr><td>10</td><td>SOCKET 2 TX 缓冲区</td></tr>
<tr><td>11</td><td>SOCKET 2 RX 缓冲区</td></tr>
</table></td></tr>
</table>

		... 111 / 00 / 保留 / 保留 111 / 01 / 保留 / SOCKET 7 寄存器 111 / 10 / 保留 / SOCKET 7 TX 缓冲区 111 / 11 / 保留 / SOCKET 7 RX 缓冲区
11	IDM_DR	**间接模式数据寄存器** 数据

5.2.1 并行总线数据写入

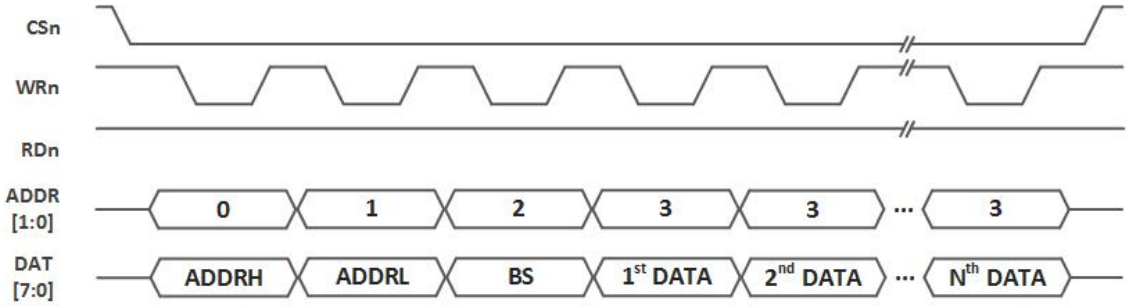


图 15 并行总线 N 字节数据写访问

图 15 显示了通过并行总线写入的 N 字节数据。在 N 字节数据传输期间，主机将 CSn 置为低电平，如果完成，则主机将 CSn 置为高电平。在写访问中，主机应该在总线转换时切换 WRn。主机在 ADDR[1:0]上发送‘00’时，表示 ADDRH 在 DAT[7:0]上；当 ADDR[1:0]发送‘01’时，表示 ADDRL 在 DAT[7:0]上；当 ADDR[1:0]发送‘10’时，表示 BS 在 DAT[7:0]上；当 ADDR[1:0]发送‘11’时，表示 DATA 在 DAT[7:0]上。

如果有多个字节数据，可以连续跟随 ADDR [1:0]上的 “11”和 DAT [7:0]上的数据。

5.2.2 并行总线数据读取

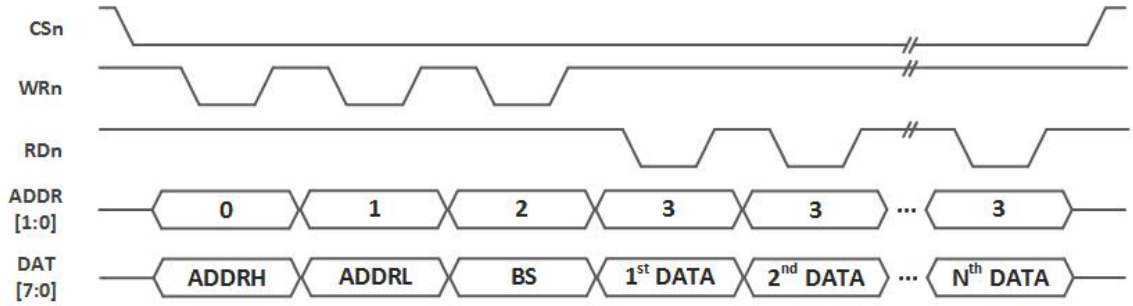


图 16 并行模式连续读访问

图 16 显示了通过并行总线读取的 N 字节数据。在 N 字节数据读取期间，主机将 CSn 置为低电平，如果完成，则主机将 CSn 置为高电平。

在读访问中，每次总线转换时，主机应该切换 WRn 来控制总线，切换 RDn 来读取数据。

主机在 ADDR[1:0]上发送‘00’时，表示 ADDRH 在 DAT[7:0]上；当 ADDR[1:0]发送‘01’时，表示 ADDRL 在 DAT[7:0]上；当 ADDR[1:0]发送‘10’时，表示 BS 在 DAT[7:0]上。

在发送三个字节后，主机在 ADDR [1:0]上发送'11'并在 DATA[7:0]上读取数据。如果要读取多个字节数据，主机可以在 ADDR [1:0]上连续发送'11'，并在 DATA[7:0]上读取数据。

6 功能描述

W6100 可以通过配置寄存器可以实现网络通信。本节主要介绍在使用 TCP、UDP、IPRAW 和 MACRAW 等协议和模式时如何初始化 W6100 相关的寄存器。

6.1 初始化

在 W6100 初始化过程中需要配置网络参数和设置 SOCKET n TX/RX 缓冲区。

6.1.1 网络信息配置

配置 IPv4/IPv6 基本网络信息。

网络配置解锁：

```
{
    /* 配置网络信息前解锁网络 */
    NETLCKR = 0x3A;
}
```

源 MAC 地址：

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

IPv4 网络信息：

```
{
    /* 网关 IP 地址, 192.168.0.1 */
    GAR[0:3] = { 0xC0, 0xA8, 0x00, 0x01 };
    /* 子网掩码, 255.255.255.0 */
    SUBR[0:3] = { 0xFF, 0xFF,, 0xFF, 0x00};
    /* IP 地址, 192.168.0.100 */
    SIPR[0:3] = {0xC0, 0xA8,0x00, 0x64};
}
```

IPv6 网络信息：

```
{
    /* 链路本地地址, FE80::1322:33FF:FEAA:BBCC */
    LLAR[0:15] = { 0xFE, 0x80, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,
    0x13, 0x22, 0x33, 0xFF, 0xFE, 0xAA, 0xBB, 0xCC };
    /* 全球单播地址, 2001:0DB8:E001::1222:33FF:FEAA:BBCC */
    GUAR[0:15] = { 0x20, 0x01, 0x0D, 0xB8, 0xE0, 0x01, 0x00, 0x00,
    0x13, 0x22, 0x33, 0xFF, 0xFE, 0xAA, 0xBB, 0xCC };
    /* IPv6 子网掩码, FFFF:FFFF:: */
    SUB6R[0:15] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
```

```
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
    /* IPv6 网关地址, FE80::1322:33FF:FE44:5566 */
    GA6R[0:15] = { 0xFE, 0x80,0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x13, 0x22, 0x33, 0xFF, 0xFE, 0x44, 0x55, 0x66 };
}
```

网络配置锁定：

```
{
    /* 网络锁定前配置网络信息 */
    NETLCKR = Any value except 0x3A;
}
```

6.1.2 SOCKET TX/RX 缓冲区设置

在 SOCKET 打开之前，用户需要通过设置 Sn_TX_BSR/Sn_RX_BSR 来定义 SOCKET n TX/RX 缓冲区大小。

SOCKET n TX / RX 缓冲区大小可以设置为 0,1,2,4,8 或 16KB，但 TX 或 RX 缓冲区的总大小不应超过 16KB。

在这种情况下，为每个 SOCKET 分配 2Kbytes 的 RX/TX 缓冲区

```
{
    //设置 SOCKET n TX / RX 缓冲区的基地址
    TxTotalSize = 0;            //检查 SOCKET n TX 缓冲区的总大小
    RxTotalSize = 0;            //检查 SOCKET n RX 缓冲区的总大小

    for (n=0; n<7; n++)  {
        Sn_TX_BSR = 2; //为每个 SOCKET 分配 2 KB 的 TX 缓冲区
        Sn_RX_BSR = 2; //为每个 SOCKET 分配 2 KB 的 RX 缓冲区
        TxTotalSize = TxTotalSize + Sn_TX_BSR;
        RxTotalSize = RxTotalSize + Sn_RX_BSR;
        If( TxTotalSize > 16 or RxTotalSize > 16 ) goto ERROR;   // 无效的总大小
    } // 结束
}
```

6.2 TCP

TCP（传输控制协议）是基于传输层上的 1 对 1 面向连接的双向数据传输协议，它提供了一种由端口号指定的应用程序之间的通信方式。

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。

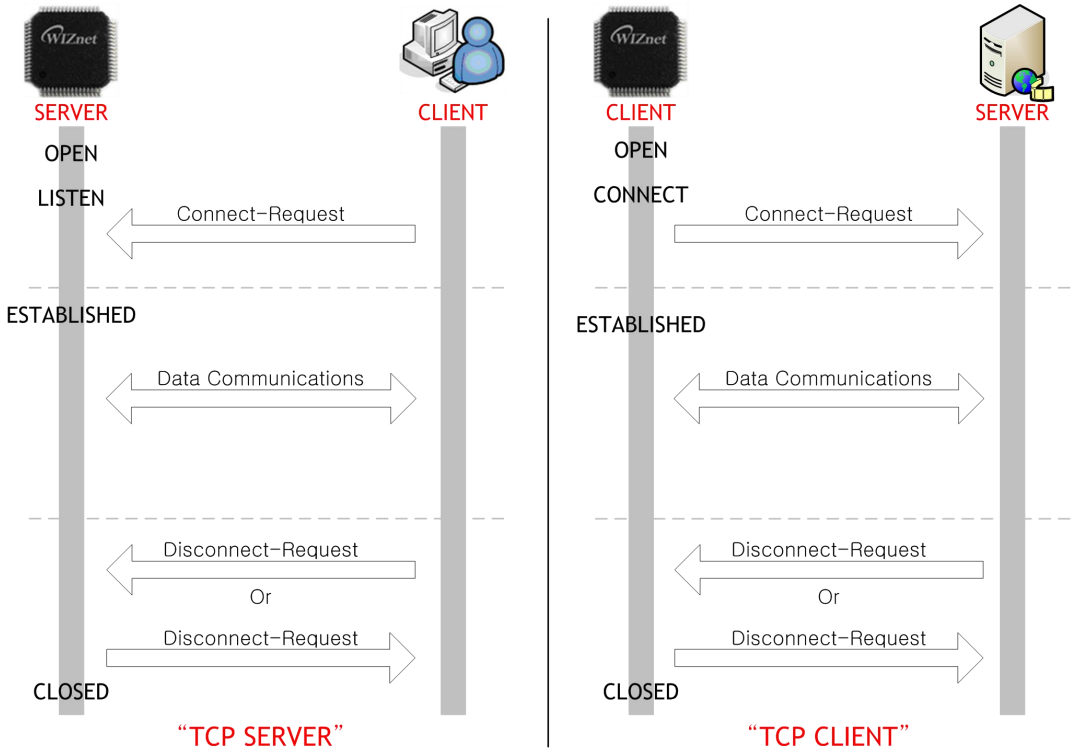


图 17 TCP 服务器和 TCP 客户端

6.2.1 TCP 服务器

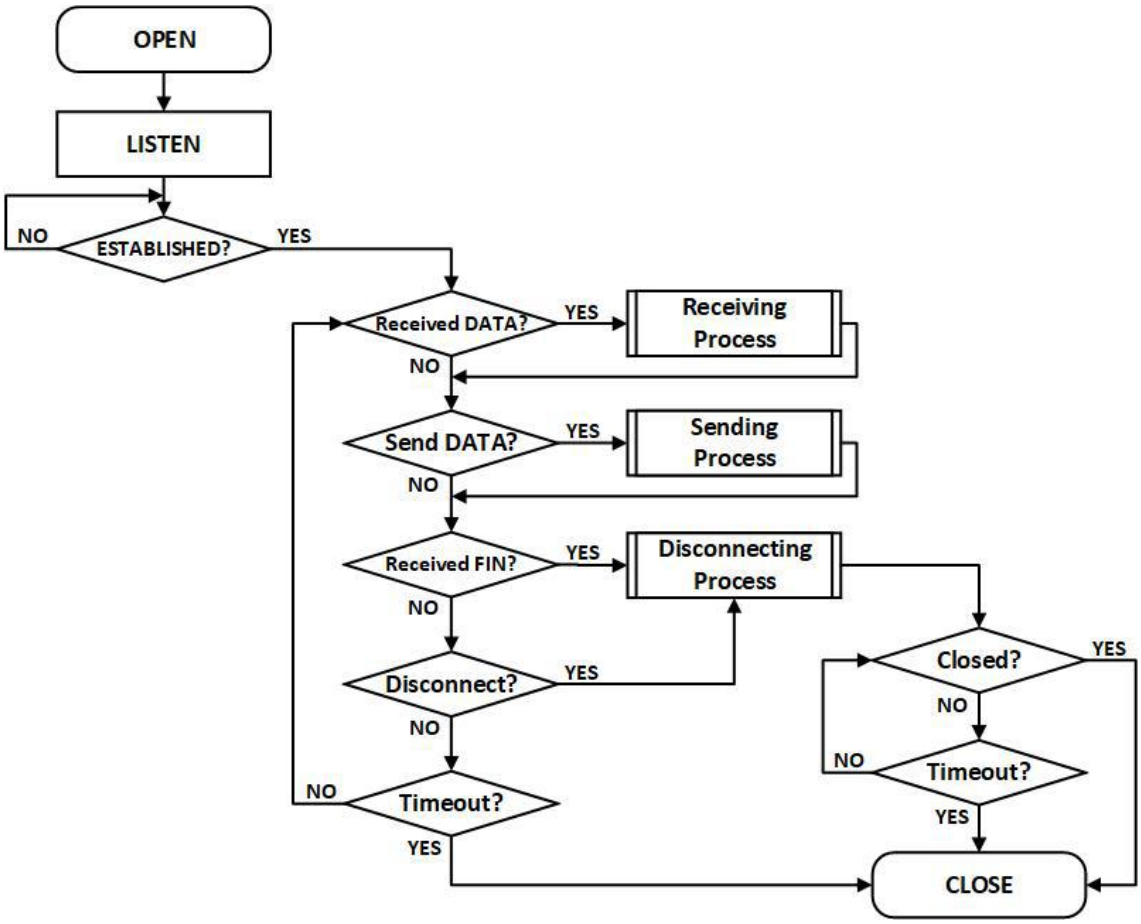


图 18 TCP 服务器操作流程

- **打开（OPEN）**

将 SOCKET n 打开为 TCP4 或 TCP6 模式。

```
TCP 模式 : TCP4, TCP6
{
  START :
  Sn_MR[3:0] = '0001'; /* 设置 TCP4 模式*/
  // Sn_MR[3:0] = '1001'; /* 设置 TCP6 模式 */
  Sn_PORTR[0:1] = {0x13,0x88}; /* 设置端口号, 5000(0x1388) */
  Sn_CR[OPEN] = '1'; /* 设置 OPEN 命令 */
  while(Sn_CR != 0x00); /*等待 OPEN 命令被清除*/
  /* 检查 SOCKET 状态 */
    if(Sn_SR != SOCK_INIT) goto START;
}
```

- **监听（LISTEN）**

SOCKET n 通过 Sn_CR[LISTEN]命令初始化为 TCP SERVER 建立侦听。用户可以通过读取 Sn_SR（SOCK_LISTEN）来检查它的状态。

```
{
    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（ESTABLISHED）**

TCP SERVER 在接收 SYN 数据包之前保持 LISTEN 状态（Sn_SR = SOCK_LISTEN）。若 TCP SERVER 从 TCPCLIENT 收到 SYN 数据包，则它将 SYN/ACK 数据包发送到 TCPCLIENT，如果收到 ACK 数据包，则建立 TCP SERVER 和 TCP CLIENT 之间的连接。

当连接建立时，Sn_IR[CON]中断发生，Sn_SR 值更改为 SOCK_ESTABLISHED。用户可以从 Sn_DIPR 或 Sn_DIP6R 寄存器中读取目标地址。

第一种方法：

```
{
    /* 检查 SOCKET 中断 */
    if(Sn_IR[CON] == ‘1’)
    {
        Sn_IRCLR[CON] = ‘1’; /* 清除 SOCKET 中断 */
        goto Received DATA; /* 接收数据*/
    } //end if
    else if(Sn_IR[TIMEOUT] == ‘1’) goto Timeout;

    /* 检查目标地址*/
    if(Sn_MR[3:0] == TCP6 Mode)
        destination_addr[0:15] = Sn_DIP6R;
    else if(Sn_MR[3:0] == TCP4 Mode)
        destination_addr[0:3] = Sn_DIPR;
}
```

第二种方法：

```
{
    /*检查 SOCKET 状态*/
    if (Sn_SR == SOCK_ESTABLISHED)
    {
        Sn_IRCLR[CON] = ‘1’; /*清除 SOCKET 中断*/
        goto Received DATA;   /*接收数据*/
    }
    else if(Sn_IR[TIMEOUT] == ‘1’) goto Timeout;

    /*检查目标地址*/
    if(Sn_MR[3:0] == TCP6 Mode)
```

```
        destination_addr[0:15] = Sn_DIP6R;
    else if(Sn_MR[3:0] == TCP4 Mode)
        destination_addr[0:3] = Sn_DIPR;
}
```

- **接收数据（Receive DATA）**

用户可以通过读取 Sn_IR[RECV]或 Sn_RX_RSR 来判断 SOCKET n 是否接收到数据。

```
第一种方法：
{
    /*检查 SOCKET RX 缓冲区接收大小*/
    if (Sn_RX_RSR > 0) goto Receiving Process;
}
```

```
第二种发法：
{
    /*检查 SOCKET RECV 中断位*/
    if (Sn_IR[RECV] == '1')
    {
        Sn_IRCLR[RECV] = '1'; /* 清除 SOCKET 中断 */
        goto Receiving Process;
    } //end if
}
```

- **接收处理（Receiving Process）**

以下是从 SOCKET n RX 缓冲区接收数据的读取过程。

在读取接收到的数据后，用户必须根据数据读取大小增加 Sn_RX_RD，并通过发出 Sn_CR[RECV]命令使 W6100 更新 RX 缓冲区。如果在 Sn_CR [RECV]命令之后数据仍然保留在 SOCKET n RX 缓冲区中，则再次发生 Sn_IR [RECV]中断以通知用户数据仍保留在缓冲区中。

```
{
    /* 获取接收数据的大小 */
    get_size = Sn_RX_RSR;
    /*计算 SOCKET n RX 缓冲区大小*/
    gSn_RX_MAX = Sn_RX_BSR * 1024;
    /*计算读取偏移地址*/
    get_start_address = Sn_RX_RD;
    /*将 get_start_address 的 get_size 复制到 destination_address */
    memcpy(get_start_address, destination_address, get_size);
    /*将 Sn_RX_RD 增加为 get_size */
    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（Send DATA / Sending Process）**

在将数据写入 SOCKET n TX 缓冲区后，用户应通过写入数据大小增加 Sn_TX_WD，并通过设置 Sn_CR [SEND]使 W6100 传输数据。在 Sn_IR[SENDOK]中断发生之前，用户不应该执行下一个数据传输过程。此外，在数据传输期间可能会发生 Sn_IR [TIMEOUT]中断。

Sn_IR[SENDOK]中断的发生取决于 SOCKET 数量、数据大小和网络流量。传输数据大小不应超过 SOCKET n TX 缓冲区大小。大于 MSS 的数据将被分成多个 MSS 单元。

```
{
    /*计算 SOCKET n TX 缓冲区大小*/
    gSn_TX_MAX = Sn_TX_BSR * 1024;
    /*检查最大数据（send_size）和 SOCKET n TX 缓冲区的自由大小（Sn_TX_FSR） */
    if( send_size > gSn_TX_MAX ) send_size = gSn_TX_MAX;
    while(send_size > Sn_TX_FSR);   //等待 SOCKET n TX 缓冲区空闲*/
    /* 如果不想等待 TX 缓冲区空闲
    send_size = Sn_TX_FSR; //  将数据写入空闲缓冲区的大小
    */
    /*计算写偏移地址*/
    get_start_address = Sn_TX_WR;
    /*将 get_start_address 的 get_size 复制到 destination_address */
    memcpy(get_start_address, destination_address, send_size);
    /*将 Sn_TX_WR 增加为 send_size */
    Sn_TX_WR += send_size;
    /*在每个 TCP 和 TCP6 模式下设置 SEND 和 SEND6 命令*/
    Sn_CR = SEND; /*在 TCP 模式下设置 SEND 命令*/
    while(Sn_CR != 0x00); /*等待 SEND 或 SEND6 命令被清除*/
    /*等待 SEND 或 SEND6 命令完成或发生超时*/
    while(Sn_IR[SENDOK] == ‘0’ and Sn_IR[TIMEOUT] = ‘0’);
    /*清除 SOCKET 中断*/
    if(Sn_IR[SENDOK] == ‘1’) Sn_IRCLR[SENDOK] = ‘1’;
    else
        goto Timeout;
}
```

- **断开连接（被动断开）（Received FIN (Passive Close)）**

这是一个被动关闭的过程。当 W6100 接收到来自对方发送的 FIN 包时，Sn_IR[DISCON]中断发生，Sn_SR 值将变为 SOCK_CLOSE_WAIT。

```
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）（Disconnected (Active Close)）**

这是主动关闭过程，将 FIN 包发送给对方。

```
{
    Sn_CR[DISCON] = ‘1’;    /* 发送 FIN 包 */
    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 SOCKET n 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。在主动关闭模式下，SOCKET 将 FIN 数据包发送给对方并等待来自对方发送的 FIN 数据包。接收到对方发送的 FIN 数据包后，SOCKET 将被关闭。如果在整个重传时间内没有响应 FIN 包，则产生 Sn_IR [TIMEOUT]中断。

```
被动关闭：  /* 接收到从对方发来的数据包 */
{
    Sn_CR = DISCON;          /* 发送 FIN 数据包 */
    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’)
    {
        Sn_IRCLR[DISCON] = ‘1’;    /* 清除中断*/
        goto CLOSED;
    }
    else goto Timeout;
}
```

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

```
    else goto Timeout;
}
```

- **超时（Timeout）**

如果没有响应 SYN/DATA/FIN 包，则重传过程工作。当重传失败时，Sn_IR[TIMEOUT]中断发生。

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

- **关闭（CLOSE）**

SOCKET n 通过断开进程、Sn_IR[TIMEOUT]或 Sn_CR[CLOSE]来关闭。

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

6.2.2 TCP 客户端

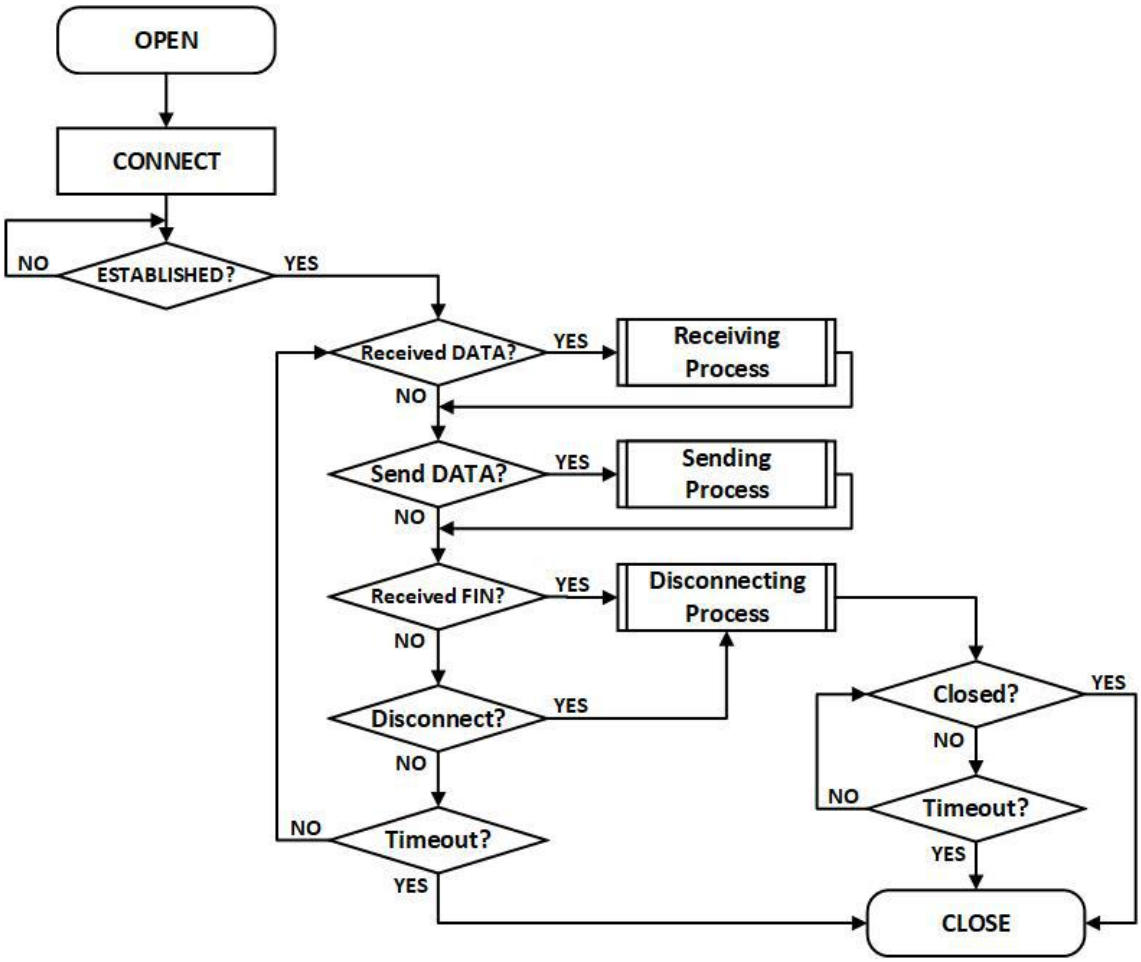


图 19 TCP 客户端操作流程

- **打开（OPEN）**

参考 6.2.1 TCP 服务器: OPEN

- **连接（CONNECT）**

SOCKET n 由 Sn_CR [CONNECT]作为 TCP 客户端运行。

它通过 Sn_CR [CONNECT]或 Sn_CR [CONNECT6]将 SYN 数据包发送到 TCP SERVER。

```
Sn_MR[3:0] = TCP4:
{
    /*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};
    Sn_CR = CONNECT; /*在 TCP 模式下设置 CONNECT 命令*/
    while(Sn_CR != 0x00); /*等待 CONNECT 或 CONNECT6 命令被清除*/
    goto ESTABLISHED;
}
```

```
Sn_MR[3:0] = TCP6:
```

```
{
    /*设置目标 IP 地址, FE80::10D:FC:34A:EF90 */
    Sn_DIP6R[0:15] = {0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    0x01, 0x0D, 0x00, 0xFC, 0x03, 0x4A, 0xEF, 0x90};
    /* 设置目标端口号, 5000(0x1388) */
    Sn_PORTR[0:1] = {0x13, 0x88};
    Sn_CR = CONNECT6;   /*在 TCP6 模式下设置 CONNECT6 命令*/
    while(Sn_CR != 0x00); /*等待 CONNECT 或 CONNECT6 命令被清除*/
    goto ESTABLISHED;
}
```

•建立连接（ESTABLISHED）

在发送 SYN 数据包之后，TCP CLIENT'维持 SOCK_SYNSENT 状态，直到从 TCP SERVER 接收到 SYN/ACK 数据包。当接收到从 TCP SERVER 发送的 SYN/ACK 数据包时，TCP SERVER 和 TCP CLIENT 之间的连接过程完成。如果连接完成，则发生 Sn_IR[CON]中断，并将 Sn_SR 更改为 SOCK_ESTABLISHED。用户可以通过 Sn_DIPR 或 Sn_DIP6R 寄存器查看目标地址。

参考 *6.2.1 TCP 服务器: 建立连接*

• 其他（Others flow）

参考 *6.2.1 TCP 服务器: 建立连接*

6.2.3 TCP DUAL

SOCKET 支持 IPv4 和 IPv6 的 TCP 双协议（TCPD）模式。

当在 TCPD 模式下打开的 SOCKET 通过 Sn_CR[LISTEN]作为“TCP 双服务器”运行时，它是作为 TCP4 工作，还是 TCP6 工作，取决于连接的目标 IP 版本。当作为“TCP 双客户端”运行时，它是作为 TCP4 还是 TCP6 运行由 Sn_CR [CONNECT]或 Sn_CR [CONNECT6]决定。

建立连接后，主机可以通过检查 Sn_ESR[TCPM]来了解 SOCKET 是作为 TCP4 还是 TCP6 运行。

6.4.1.1 TCP 双服务器

TCP DUAL SERVER 操作流程与图 18 相同。

•打开（OPEN）

以 TCPD 模式打开 SOCKET n。

```
TCP 模式 : TCP4, TCP6, TCPD
{
    START :
    Sn_MR[3:0] = '1101'; /* 设置 TCPD 模式 */
    Sn_PORTR[0:1] = {0x13,0x88}; /* 设置端口号 5000(0x1388) */
    Sn_CR[OPEN] = '1'; /*设置 OPEN 命令*/
    while(Sn_CR != 0x00); /*等待 OPEN 命令被清除*/
    /*检查 SOCKET 状态*/
    if(Sn_SR != SOCK_INIT) goto START;
}
```

• 其他（Others flow）

参考 *6.2.1 TCP 服务器*

6.4.1.2 TCP 双客户端

TCP DUAL CLIENT 操作流程与图 19 相同。

• 打开（OPEN）

参考 *6.2.3.1 TCP 双服务器 : 打开*

•连接（CONNECT）

SOCKET n 通过 Sn_CR [CONNECT]或 Sn_CR [CONNECT6]将 SYN 数据包发送到目的地。

```
TCP4 :
{
    /* 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};
    Sn_CR = CONNECT; /* 设置 CONNECT 命令 */
    while(Sn_CR != 0x00); /* 等待 CONNECT 或 CONNECT6 命令被清除 */
    goto ESTABLISHED;
}
```

```
TCP6 :
{
    /* 设置目标 IP 地址, FE80::10D:FC:34A:EF90 */
    Sn_DIP6R[0:15] = {0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    0x01, 0x0D, 0x00, 0xFC, 0x03, 0x4A, 0xEF, 0x90};
    /* 设置目标端口号, 5000(0x1388) */
    Sn_PORTR[0:1] = {0x13, 0x88};
    Sn_CR = CONNECT6; /* 设置 CONNECT6 命令 */
    while(Sn_CR != 0x00); /*等待 CONNECT 或 CONNECT6 命令被清除*/
    goto ESTABLISHED;
}
```

- **其他(Others flow)**

参考 *6.2.1 TCP 服务器*

6.2.4 其他功能

6.2.4.1 TCP SOCKET 设置

在 Sn_CR[OPEN]命令打开 SOCKET n 之前，用户需要使用 Sn_MR 和 Sn_MR2 设置 SOCKET。

- **无延迟 ACK：Sn_MR[ND] = '1'**

无延迟 ACK 选项用于在从对方接收 DATA 时无延迟地发送 ACK 数据包。

- **延迟 ACK：Sn_MR[ND] = '0'**

如果没有设置无延迟 ACK(No Delayed ACK)选型，则当 RTR 中的时间过去或者当通过 Sn_CR [RECV]命令使 TCP 窗口大小小于配置的 MSS 时，SOCKET 会响应来自对方发送的 ACK 数据包。

- **强制 PSH Flag：Sn_MR[FPSH] = '1'**

如果设置了强制 PSH 选项，SOCKET 将在每个要传输的数据包中添加 PSH 标志。

- **自动 PSH Flag：Sn_MR[FPSH] = '0'**

如果强制 PSH 选项被清除，SOCKET 将 PSH 标志放在 Sn_CR [SEND]发送的最后一个数据包中。

- **目标 MAC 地址为 Sn_DHAR：Sn_MR2[DHAM] = '1'**

如果设置了 Sn_MR2 [DHAM]，则跳过 ARP / ND 进程，并将 Sn_DHAR 用作目标 MAC 地址。

- **目标 MAC 地址为 ARP：Sn_MR2[DHAM] = '0'**

在"TCP 服务器"模式下，从接收到的 SYN 数据包中获取目标 MAC 地址。在"TCP 客户端"模式下，从 ARP / ND 进程获取目标 MAC 地址。

- **目标 MAC 地址为 Sn_DHAR：Sn_MR2[FARP] = '1'**

在"TCP 服务器"模式下，在响应从"TCP 客户端"接收到的 SYN 数据包的 SYN / ACK 数据包之前执行 ARP 过程。使用从 ARP/ND-process 中获取的地址作为目标 MAC 地址。 如果还设置了 Sn_MR2 [DHAM]，则执行 ARP/ND-process，但使用 Sn_DHAR 作为目标 MAC 地址。

- **目标 MAC 地址模式 ARP：Sn_MR2[FARP] = '0'**

在'TCP 服务器'模式下，从接收到的 SYN 数据包中获取目标 MAC 地址。在"TCP 客户端"模式下，从 ARP / ND 进程获取目标 MAC 地址。

6.2.4.2 Keep Alive

Keep Alive (KA)是重新传输已传输数据包的最后 1 个字节，以检查连接是否有效。在使用 Keep Alive 函数之前，必须传输一个或多个字节的数据大小。如果在整个重传时间内没有对 KA 包的响应，就会发生 Sn_IR[TIMEOUT]中断。

KA 包传输的周期在 Sn_KPALVTR 中设置。如果 Sn_KPALVTR 设置为零，则可以通过 Sn_CR [SEND_KEEP]命令发送 KA 数据包。

6.3 UDP

UDP（User Datagram Protocol）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。它也可以使用端口号在应用层之间通信。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 无法保证可靠性，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。图 20 所示为 UDP 操作流程.

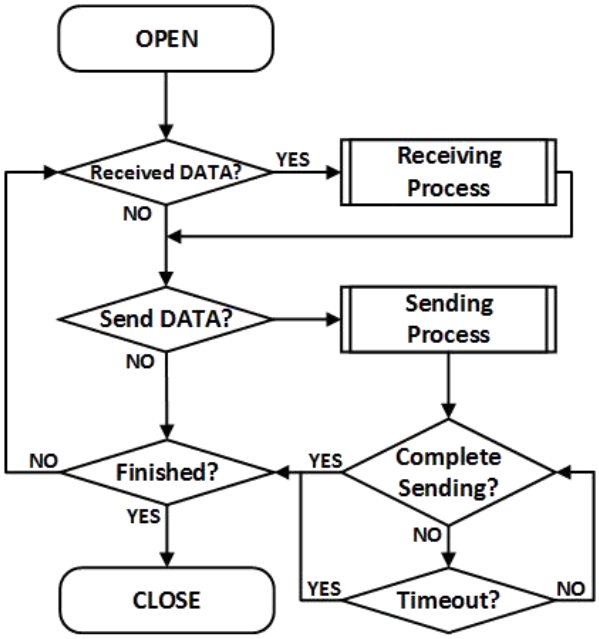


图 20 UDP 操作流程

6.3.1 UDP 单播

UDP 单播是一种一对一的通信方式。在数据发送之前，SOCKET 执行 ARP/ND-process。 在 ARP/ND-process 中，可能会发生 Sn_IR [TIMEOUT]中断。请参考 *6.7 重传*。
如果设置 Sn_MR2[DHAM]，则跳过 ARP/ND-process，并使用 Sn_DHAR 作为目标 MAC 地址。
UDP 单播操作流程如图 20 所示。

- **打开（OPEN）**

打开 SOCKET n 到 UDP4 或 UDP6 模式。

```
UDP4, UDP6 模式:
{
    START :
    Sn_MR[3:0] = '0010';   /* 设置 UDP4 模式 */
    // Sn_MR[3:0] = '1010; /* 设置 UDP6 模式 */
    /*设置源端口号, 5000(0x1388) */
    Sn_PORTR[0:1] = {0x13, 0x88};
    Sn_CR[OPEN] = '1'; /* 设置 OPEN 命令 */
    while(Sn_CR != 0x00); /* 等待 OPEN 命令被清除 */
    /*检查 SOCKET 是否为 UDP6 模式*/
```

```
    if(Sn_SR != SOCK_UDP) goto START;
}
```

- **接收数据（Received DATA）**

参考 ***6.2.1 TCP 服务器: 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。接收到的数据包存储在具有“PACKET INFO”的 SOCKET n RX 缓冲区中，如图 21 所示。主机必须从 SOCKET n RX 缓冲区内读取数据，格式如图 21 所示。如果接收到的数据被分段或大于 SOCKET n RX 缓冲区空闲大小，则丢弃该数据。

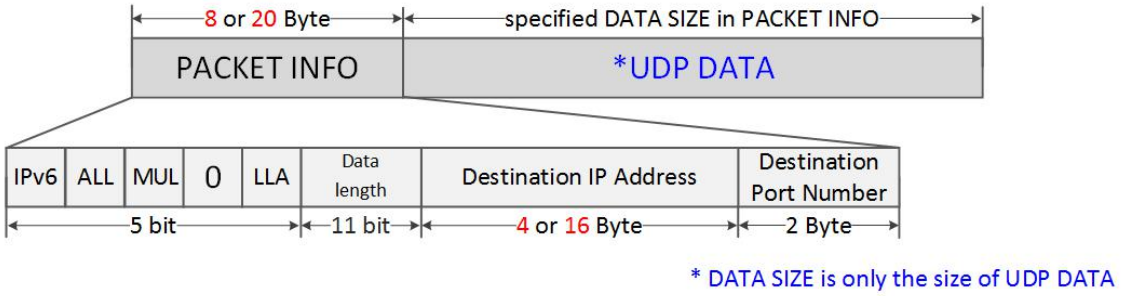


图 21 在 SOCKET n 接收缓冲区中接收到 UDP 数据

表 2 PACKET INFO 包参数描述

PACKET INFO	描述
IPv6	0： 接收 UDP / IPv4 数据包 1： 接收 UDP / IPv6 数据包
BRD/ALL	0： 其他 1： 接收广播/全节点组播包
MUL	0： 其他 1： 收到组播包
0	永远为‘0’
LLA	0：GUA 1：LLA
数据长度	* UDP 数据长度
目的 IP 地址	如果接收到 UDP4 包，保存目标 IPv4 地址(4 字节)，如果接收到 UDP6 包，保存目标 IPv6 地址(16 字节)
目标端口号	目标端口号

```
UDP4 模式:
{
    /*接收 PACKINFO */
    参考 6.2.1 TCP 服务器: 接收处理 with get_size = 8 bytes;

    /*在 PACKET INFO 中提取目标 IP，端口，大小*/
```

```
    data_Info = destination_address[0] & “11111000”;
    data_size = (destination_address[0] & “00000111” << 8) + destination_address[1];
    if( data_info & ‘10000000’ == 0 ) /* 目标地址是 IPv4 */{
        dest_ip[0:3] = destination_address[2:5];
        dest_port = (destination_address[6] << 8) + destination_address[7];
    }
    /*读取 UDP 数据*/
    参考 6.2.1 TCP 服务器: 接收处理 with get_size = data_size;
}
```

```
UDP6 模式:
{
    /*接收 PACKINFO */
    参考 6.2.1 TCP 服务器: 接收处理 with get_size = 20 bytes;;
    /*在 PACKET INFO 中提取目标 IP，端口，大小*/
    data_Info = destination_address[0] & “11111000”;
    data_size = (destination_address[0] & “00000111” << 8) + (destination_address[1];
    if( data_info & ‘1000000’ ! = 0)   /* 目标地址 IPv6 */{
        dest_ip[0:15] = destination_address[2:17];
        dest_port = (destination_address[18] << 8) + destination_address[19];
    }
    /* 读取 UDP 数据 */
    参考 6.2.1 TCP 服务器: 接收处理 with get_size = data_size;
}
```

- **发送数据/发送处理（Send DATA /Sending Process）**

参考 *6.2.1 TCP 服务器: 发送数据/发送处理*

```
UDP4 模式
{
    /*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};

    参考  6.2.1 TCP 服务器: 发送处理 with Sn_CR[SEND];
}
```

```
UDP6 模式
{
    /*设置目标 IP 地址, FE80::10D:FC:34A:EF90 */
    Sn_DIP6R[0:15] = {0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    0x01, 0x0D, 0x00, 0xFC, 0x03, 0x4A, 0xEF, 0x90};
    /*设置目标端口号, 5000(0x1388) */
    Sn_DPORTR[0:1] = {0x13, 0x88};

    参考  6.2.1 TCP 服务器: 发送处理 with Sn_CR[SEND6];
}
```

- **发送完成/超时（Complete Sending/ Timeout）**

当主机开始或不同的目标发送数据到一个目标时，ARP 进程在发送数据包之前执行。在 ARP-process 中，如果在整个重传时间内在没有响应来自对方的 ARP 请求，则会发生 Sn_IR[TIMEOUT]中断。

与 TCP 不同，UDP SOCKET 不会被 Sn_IR [TIMEOUT]关闭，因为它支持 1:N 通信。

参考 *6.7 重传*。

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

- **完成/关闭（Finished / CLOSE）**

由 Sn_CR 关闭[关闭]。

```
{
    Sn_CR = CLOSE; /*设置 CLOSE 命令*/
    while(Sn_CR != 0x00); /*等待 CLOSE 命令被清除*/
    /*等待 SOCKET n 关闭*/
    while(Sn_SR == SOCK_CLOSED);
}
```

6.3.2 UDP 广播

UDP 广播是一种通信方法，发送方在同一网络上向所有设备发送数据。广播有两种类型。指向网络的广播和指向子网的广播。

在 UDP6 模式下，使用 FF02::01 地址（全节点多播地址）与 UDP4 的全节点广播进行相同的操作。

- **打开（OPEN）**

参考 ***6.3.1 UDP 单播： 打开***

- **接收数据（Received DATA）**

参考 ***6.2.1 TCP 服务器: 接收数据***

- **接收处理（Receiving Process）**

参考参考 ***6.3.1 UDP 单播： 接收处理***

- **发送数据/发送处理（Send DATA / Sending Process）**

设置 UDP4 广播和 UDP6 全节点组播的目标地址

```
UDP4 全节点广播:
{
/* 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_DPORTR[0:1] = {0x13,0x88};

goto 6.2.1 TCP 服务器: 发送处理 with Sn_CR[SEND];
}
```

```
UDP4 子网广播: 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};
/* set 设置目标端口号 Destination PORT Number, 5000(0x1388) */
```

Sn_DPORTR[0:1] = {0x13,0x88}; goto *6.2.1 TCP 服务器: 发送处理 with Sn_CR[SEND];* }

UDP6 全节点组播:
{
/* 设置目标 IP 地址, FF02::01 */
Sn_DIP6R[0:15] = {0xFF, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01};
/* 设置目标端口号, 5000(0x1388) */
Sn_DPORTR[0:1] = {0x13,0x88};
 	
goto *6.2.1 TCP 服务器: 发送处理 with Sn_CR[SEND];*
}

- **发送完成/超时（Complete sending / Timeout）**

参考 ***6.3.1 UDP 单播： 发送完成/超时***

- **完成/关闭（Finished / CLOSE）**

参考 ***6.3.1 UDP 单播： 完成/关闭***

6.3.3 UDP 组播

UDP 组播是一对多的通信方式。

在 IPv4 模式下，组播组地址范围为 224.0.0.0~239.255.255.255（参考 *IANA_Multicast Address*），相对应的 MAC 地址为 01:00:5E:00:00:00~01:00:5E:FF:FF:FF。在设置组播 MAC 地址时，MAC 地址的低 23 位具有不组播组相同的地址。(Ref *rfc1112*)

在 IPv6 模式下，设置组播组地址如*图 22 IPv6 组播组地址格式*所示。UDP 组播过程参考图 20。UDP 组播操作流程与图 22 相同。

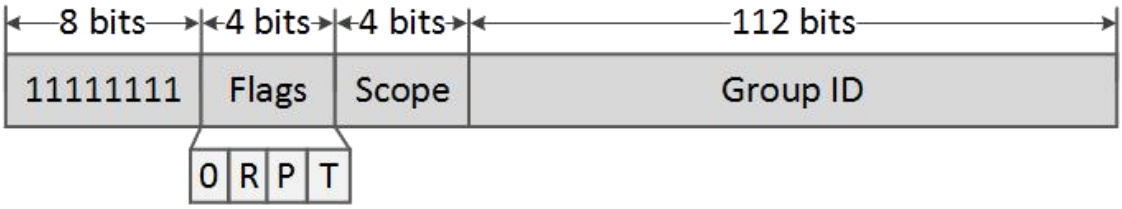


图 22 IPv6 组播组地址格式

表 3 IPv6 组播地址的标志参数

标志	描述
0	常为 ‘0’
R	1： 嵌入式 RP 0： 非嵌入式 RP * 若 R = ‘1’，则 P 必须为 ‘1’
P	1 ： 基于单播网络前缀 0 ： 非基于单播网络前缀 *若 P =‘1’，则 T 必须为 ‘1’
T	1： 临时地址（本地分配） 0： 永久地址（IANA 分配）

表 4 IPv6 组播地址范围定义

范围	描述
1	节点（Node）
2	链接(Link)
3	子网（Subnet）
4	管理员（Admin）
5	网站（Site）
8	组织（Organization）
E	全球（Global）

- **打开（OPEN）**

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

在 UDP4 组播模式下，通过 Sn_CR [OPEN]命令发送 IGMP（Internet Group Management Protocol）连接消息。IGMP 版本由 Sn_MR [MS]设置为版本 1 或版本 2。

在 UDP6 组播模式下，使用 MLDv1 加入组播组。

```
UDP4 组播模式：{
开始：/*设置组播组 MAC 地址, 01:00:5E:00:00:64 */
Sn_DHAR[0:5] = {0x01, 0x00, 0x5E, 0x00, 0x00, 0x64};
/*设置组播组 IP 地址, 224.0.0.100 */
Sn_DIPR[0:3] = {0xE0, 0x00, 0x00, 0x64}
/*设置组播组端口号, 3000(0x0BB8) */
Sn_DPORTR[0:1] = {0x0B, 0xB8};

Sn_MR[MULTI] = '1'; /*设置 UDP 组播*/

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

goto  6.3.1 UDP 单播: 打开(UDP Mode)
}
```

```
UDP6 组播模式：
{
开始：
/*设置组播组 MAC 地址, 33:33:00:AB:34:56 */
Sn_DHAR[0:5] = {0x33, 0x33, 0x00, 0xAB, 0xCD, 0xEF};
/*设置组播组 IP 地址, FF02::100:00AB:CDEF */
Sn_DIP6R[0:15] = {0xFF, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x00, 0x00, 0x00, 0x01, 0x00, 0xAB, 0xCD, 0xEF};
/*设置组播组端口号, 3000(0x0BB8) */
Sn_DPORTR[0:1] = {0x0B, 0xB8};

Sn_MR[MULTI] = '1'; /*设置 UDP 组播*/
goto 6.3.1 UDP 单播: 打开(UDP6 Mode)
}
```

- **接收数据（Received DATA）**

参考 ***6.2.1 TCP 服务器: 接收数据***

- **接收处理（Receiving Process）**

参考 6.3.1 UDP 单播: ***接收处理***

- **发送数据/发送处理（Send DATA / Sending Process）**

参考 6.3.1 UDP 单播: *发送处理*

- **发送完成/超时（Complete sending / Timeout）**

参考 6.3.1 UDP 单播: *发送完成/超时*

- **完成/关闭（Finished? / CLOSE）**

参考 6.3.1 UDP 单播: *完成/关闭*

6.3.4 UDP DUAL 模式

SOCKET 提供基于 W6100 双协议栈（IPv4 / IPv6）的 UDP 双（UDPD）模式。 在 UDPD 模式下打开的 SOCKET 可以发送/接收所有 UDP4 / UDP6 数据包。UDP4 数据和 UDP6 数据可以分别通过 Sn_CR [SEND]和 Sn_CR [SEND6]发送，并且接收的 UDP4 数据和 UDP6 数据可以通过接收分组的“PACKET INFO”来区分。

UDPD 操作流程如图 20 所示。

- **打开（OPEN）**

打开 SOCKET n 为双 UDP 模式。

```
UDP6 模式
{
开始:
Sn_MR[3:0] = ‘1110’; /* 设置 UDPD 模式 */

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

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

/*检查 SOCKET 是否是 UDPD 模式*/
if(Sn_SR != SOCK_UDP) goto START;
}
```

- **接收数据（Received DATA）**

参考 *6.2.1 TCP 服务器: 接收数据*

- **接收处理（Receiving Process）**

在 UDP DUAL 模式下，SOCKET 可以接收一个或多个目标发送的 UDP4 / UDP6 数据包，接收到的数据包以'PACKET INFO'存储在 SOCKET n RX 缓冲区中。主机可以从'PACKET INFO’中获取所接收数据包的 IP 版本、传输方式和目标信息。

主机必须从 SOCKET n RX 缓冲区读取数据，格式如图 21 所示。如果接收到的数据被分段或大于 SOCKET n RX 缓冲区空闲大小，则丢弃该数据。参考图 21 和表 2 PACKET INFO 中的参数描述

```
{
/*在 PACKINFO 中提取上面的 2 个字节*/
goto 6.2.1 TCP 服务器: 接收处理 with get_size = 2 bytes;

data_size = (destination_address[0] & “00000111” << 8) + destination_address[1];

/*
检查 UDP4 或 UDP6 数据包，在 PACKE INFO 中提取目标 IP，端口，大小
*/
if(destination_address[0] & “10000000” == 0) /* UDP4 数据包*/
{
    goto 6.2.1 TCP 服务器: 接收处理 with get_size = 6 bytes;
dest_ip[0:3] = destination_address[0:3];
dest_port = (destination_address[4] << 8) + destination_address[5];
}
else /* UDP6 数据包*/
{
goto 6.2.1 TCP 服务器: 接收处理 with get_size = 18 bytes;
dest_ip[0:15] = destination_address[0:15];
dest_port = (destination_address[16] << 8) + destination_address[17];
}
/* 读取 UDP 数据 */
goto 6.2.1 TCP 服务器 : 接收处理 with get_size = data_size;
}
```

•发送数据/发送处理（Send DATA / Sending Process）

参考 ***6.3.1 UDP 单播: 发送数据/ 发送处理***

• 发送完成/超时（Complete sending? / Timeout?）

参考 ***6.3.1 UDP 单播: 发送完成/ 超时***

•完成/关闭（Finished? / CLOSE）

参考 ***6.3.1 UDP 单播: 完成/ 关闭***

6.3.5 其他功能

6.3.5.1 UDP 模式 SOCKET 选项

在 Sn_CR [OPEN]命令之前，可以通过 Sn_MR 和 Sn_MR2 设置 SOCKET 选项。

- **目标 MAC 地址 by Sn_DHAR : Sn_MR2[DHAM]= '1'**

  跳过 ARP/ND-process，使用 Sn_DHAR 作为目标 MAC 地址。

- **目标 MAC 地址 by ARP : Sn_MR2[DHAM]= '0'**

  要传输的 UDP 数据的目标 MAC 地址用作从 ARP/ND-process 获取的地址。

- **强制 ARP : Sn_MR2[FARP]= '1'**

  每当 UDP 数据包由 Sn_CR [SEND]或 Sn_CR [SEND4]发送时，就执行 ARP/ND-process。

  如果还设置了 Sn_MR2 [DHAM]，则执行 ARP/ND-process，但使用 Sn_DHAR 作为目标 MAC 地址。

- **自动 ARP : Sn_MR2[FARP]= '0'**

当发送第一个 UDP 数据包或更改目标时，执行 ARP/ND-process。

6.3.5.2 UDP 阻塞

在 UDP 模式下，可以接收单播和广播数据包。但是如果 Sn_MR[BRDB]被设置为' 1 '，广播数据包就会被阻塞。在 UDP 组播模式下，可以接收单播、广播和组播包。但是，如果将 Sn_MR[UNIB]或 Sn_MR[BRDB]设置为' 1 '，则分别阻塞单播包或广播包。必须在 Sn_CR [OPEN]命令之前设置这些区域。

Sn_MR[MULTI]	Sn_MR[BRDB]	Sn_MR[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

在 UDP6 或 UDPD 模式下，当 Sn_MR [SMB]设置为 "1"时，将阻止请求的组播数据包。

6.3.5.3 端口无法到达

当发送方将 UDP 数据包发送到 W6100 未打开的端口时，W6100 会自动将目标端口不可达数据包发送给发送方。但它可能是端口扫描攻击的目标。

在 UDP4 或 UDP6 中，通过设置 NET4MR[UNRB] = ' 1 '或 NET6MR[UNRB] = ' 1 '来阻塞端口不可到达包。

6.4 IPRAW

IPRAW 支持表 7 中所示的协议通信，其中包括在网络协议层中定义的各种上层协议（参考 *IANA_Protocol Numbers*）。当 SOCKET n 作为 IPRAW4 或 IPRAW6 打开时，Sn_PNR 配置字段或 IPv6 扩展头的值。 SOCKET n 无法使用与 Sn_PNR 中设置的协议不同的协议进行通信。

表 3 IPRAW 模式支持的 Internet 协议

协议	编号	Semantic	支持

HOPOPT	0	IPv6 Hop-by-Hop Option	O
ICMP	1	Internet 控制消息协议	O
IGMP	2	Internet Group Management	O
IPv4	4	IPv4 封装	O
TCP	6	传输控制	X
UDP	17	用户数据报	X
IPv6	41	IPv6 封装	O
ICMP6	58	ICMP for IPv6	O
others	-	其他协议	O

在 IPRAW4 模式 Sn_PNR = ICMP 的情况下，不支持自动 PING 响应来自发送方的 PING 请求。PING 请求包存储在用于 IPRAW 的 SOCKET n RX 缓冲区中。 它应该由用户处理。

在 IPRAW6 模式 Sn_PNR = ICMP6 的情况下，可以通过 ICMP6BLKR 设置阻止自动应答数据包传输到回应请求，NA（邻居通告），NS（路由器通告）和 RA（路由器通告）。阻塞的数据包不存储在 SOCKET n RX Buffer 中。

*图 23 IPRAW 操作流程*显示了 IPRAW4 / IPRAW6 模式下的 SOCKET n 操作流程。

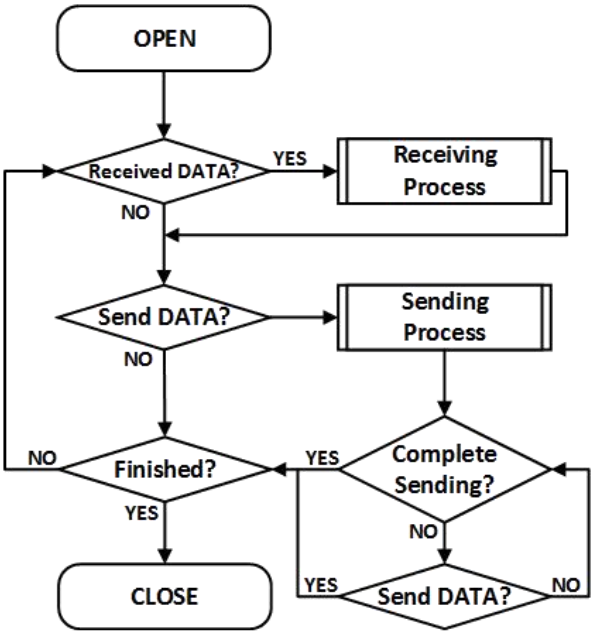


图 23 IPRAW 操作流程

- **打开（OPEN）**

在 IPRAW4 或 IPRAW6 模式下打开 SOCKET n。

```
IPRAW4 模式:
{
开始 :
Sn_PNR = protocol_num;     /*设置协议号*/

Sn_MR[3:0] = '0011';     /*设置 IPRAW4 模式*/
```

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

/*检查 IPRAW6 模式的 SOCKET*/
if(Sn_SR != SOCK_IPRAW6) goto START;
}
```

```
IPRAW6 模式:
{
开始 :
Sn_PNR = protocol_num;     /*设置协议号(下一个头) */

Sn_MR[3:0] = ‘1001’;     /*设置 IPRAW6 模式*/

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

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

- **接收数据（Received DATA?）**

参考 ***6.2.1 TCP: 接收数据***

- **接收处理（Receiving Process）**

IPRAW4/IPRAW6 模式接收来自一个或多个发送方的 IP 数据包。

为了区分每个发送方，数据包存储在 SOCKET n RX 缓冲区中，前面有“PACKET INFO”，如图 24 或图 25 所示。根据 IPRAW4/IPRAW6 模式，数据包信息有不同的格式，如表 5 和表 6 所示。

如果接收的 DATA 大于 SOCKET n RX 缓冲区空闲大小，则将其丢弃。 因此，必须以图 4 或图 25 为单位读取主机。

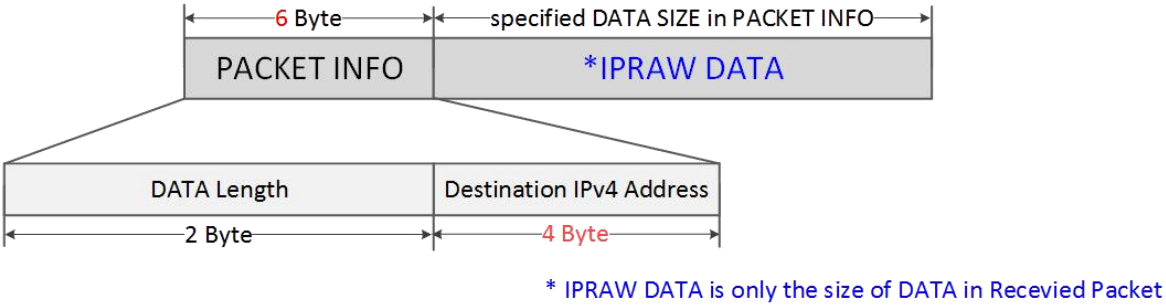
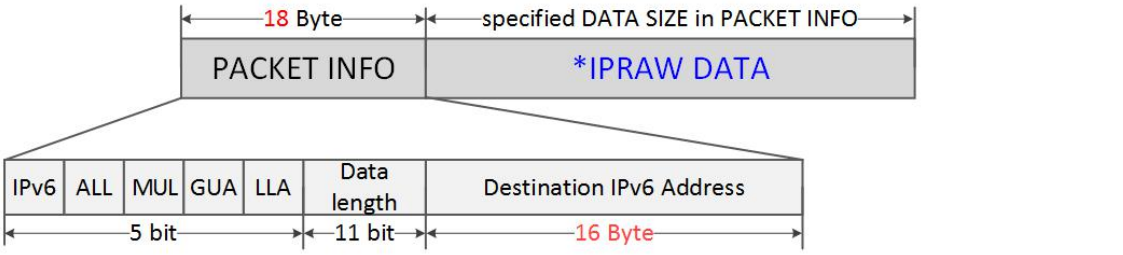


图 24 在 IPRAW4 模式 SOCKET RX 缓冲区中接收数据

表 5 IPRAW4 模式中“PACKET INFO”的参数

PACKET INFO	描述
DATA Length	* IPRAW DATA 的长度
Destination IPv4 Address	目标 IPv4 地址（4 字节）



* IPRAW DATA is only the size of DATA in Recevied Packet

图 25 在 IPRAW6 模式 SOCKET RX 缓冲区中接收到的数据

表 6 IPRAW6 模式中“PACKET INFO”的参数

PACKET INFO	描述
IPv6	如果收到 IPv6 数据包，则设置为“1”
ALL	如果收到所有节点数据包，则设置为“1”
MUL	如果收到组播数据包，则设置为“1”
GUA	如果目标地址是 GUA，则设置为“1”
LLA	如果目标地址是 LLA，则设置为“1”
数据长度	* IPRAW DATA 的长度
目标 IPv6 地址	目标 IPv6 地址（16 字节）

IPRAW4 模式 ：

{

/*接收 PACKINFO */

goto *6.2.1 TCP: Receiving Process* with *get_size* = *6*;

/*在 PACKET INFO 中提取目标数据大小，IP 地址*/

data_size = (destination_address[0] << 8) + destination_address[1];

dest_ip[0:3] = destination_address[2:5];

/* 读取 UDP 数据 */

goto *6.2.1 TCP 服务器* ： *接收处理* with *get_size* = *data_size;*

}

IPRAW6 模式:

{

/* 接收 PACKINFO */

goto *6.2.1 TCP 服务器: 接收处理* with *get_size* = *18*;

```
/*在 PACKET INFO 中提取目标信息，数据大小，IP 地址*/
data_Info = destination_address[0] & “11111000”;
data_size = (destination_address[0] & “00000111” << 8) + (destination_address[1];
dest_ip[0:15] = destination_address[2:17];

/* 读取 UDP 数据 */
goto  6.2.1 TCP 服务器: 接收处理 with get_size = data_size;
}
```

- **发送数据/发送处理（Sending DATA? / Sending Process）**

要发送的数据不得超过 SOCKET n TX 缓冲区空闲大小。 如果数据大小大于 MSS，则主机必须将较大数据拆分为多个 MSS 单元。IPRAW6 模式的 MSS 不能大于 1460，IPRAW 模式的 MSS 不能大于 1480。

```
IPRAW4 模式:
{
/*设置目标 IP 地址, 192.168.0.11 */
    Sn_DIPR[0:3] = {0xC0, 0xA8, 0x00, 0x0B};

goto  6.2.1 TCP 服务器: 发送处理;
}
```

```
IPRAW6 模式:
{
/*设置目标 IP 地址, FE80::10D:FC:34A:EF90 */
Sn_DIP6R[0:15] = {0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x01, 0x0D, 0x00, 0xFC, 0x03, 0x4A, 0xEF, 0x90};

goto  6.2.1 TCP 服务器: 发送处理;
}
```

- **发送完成/超时（Complete sending? / Timeout?）**

在 Sn_CR [SEND]或 Sn_CR [SEND6]发送第一个 DATA 数据包之前或者在 DATA 数据包从前一个目的地发送到不同目的地之前执行 ARP / ND-process。在 ARP / ND-process 中，可能会发生 Sn_IR[TIMEOUT]，并丢弃相应的数据包。因为 IPRAW4 或 IPRAW6 像 UDP 一样支持 1:N 通信，所以即使发生 Sn_IR[TIMEOUT]，SOCKET n 也不会关闭。

```
{
/*检查 TIMEOUT 中断*/
if(Sn_IR[TIMEOUT] == ‘1’)
{
```

```
Sn_IR[TIMEOUT] = ‘1’;    /*清除 TIMEOUT 中断*/
goto Finished?;
}
}
```

- **完成/关闭（Finished? / CLOSE）**

如果没有更多数据要发送，请通过 Sn_CR [CLOSE]关闭 SOCKET n。

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

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

6.4.1 其他功能

6.4.1.1 IPRAW 模式 SOCKET 设置

在 IPRAW4/IPRAW6 模式下打开 SOCKER n 的过程中，通过 Sn_MR 和 Sn_MR2 设置 SOCKET 选项。

- **目标 MAC 地址 by Sn_DHAR : Sn_MR2[DHAM]= ‘1’**

跳过 ARP/ND-process，使用 Sn_DHAR 作为目标 MAC 地址。

- **目标 MAC 地址 by ARP : Sn_MR2[DHAM]= ‘0’**

将要传输的 IPRAW 数据的目标 MAC 地址用作从 ARP/ND-process 获取的地址。

- **强制 ARP : Sn_MR2[FARP]= ‘1’**

当通过 Sn_CR[SEND]或 Sn_CR[SEND6]发送 IPRAW4 或 IPRAW6 数据时，执行 ARP/ND-process。

- **自动 ARP : Sn_MR2[FARP]= ‘0’**

当发送第一个 IPRAW 数据包或改变目标时，执行 ARP / ND-process。

6.5 MACRAW

MACRAW 模式支持以太网 MAC 的数据通信，并且仅适用于 SOCKET 0。 在 Sn_MR [MF] ='0' 的情况下，MACRAW SOCKET 0 接收所有以太网数据包。在 Sn_MR[MF] = ' 1 '的情况下，MACRAW SOCKET0 只能接收到一个包，该包具有广播、组播或源 MAC 地址(SHAR)的目标 MAC 地址。图 29 显示 MACRAW 模式 SOCKET0 操作流程。

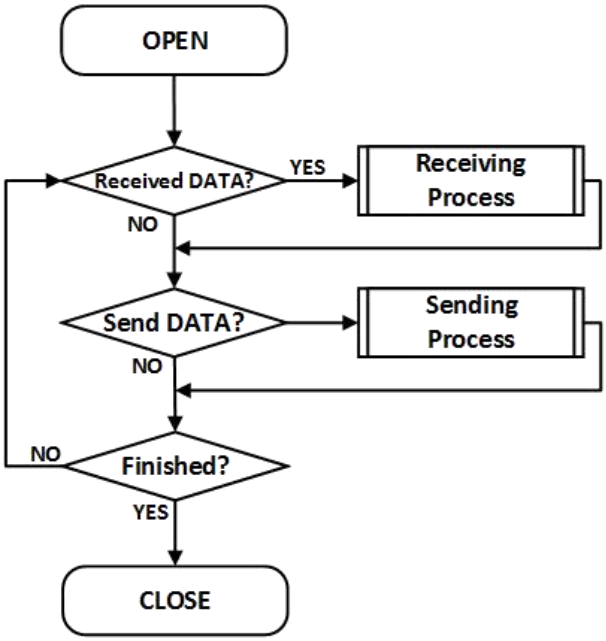


图 5 MACRAW 操作流程

- **打开(OPEN)**

在 MACRAW 模式下打开 SOCKET 0。

```
{
开始 :
S0_MR[3:0] = '0100'; /*设置 MACRAW 模式*/

/* MACRAW SOCKET 选项 MACRAW SOCKET 选项*/
/* S0_MR[MF] = '1';              // enable MAC Filter
S0_MR[UNIB] = '1';            // Broadcast Packet Block
S0_MR[MMB] = '1';             // Multicast Packet Block
S0_MR[MMB6] = '1';          // IPv6 Packet Block */

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

/*检查 SOCKET 0 是否为 MACRAW 模式*/
if(S0_SR != SOCK_MACRAW) S0_CR = CLOSE; goto START;
}
```

- **接收数据（Received DATA?）**

参考 6.2.1 TCP 服务器: 接收处理

- **接收处理（Receiving Process）**

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 30 所示。

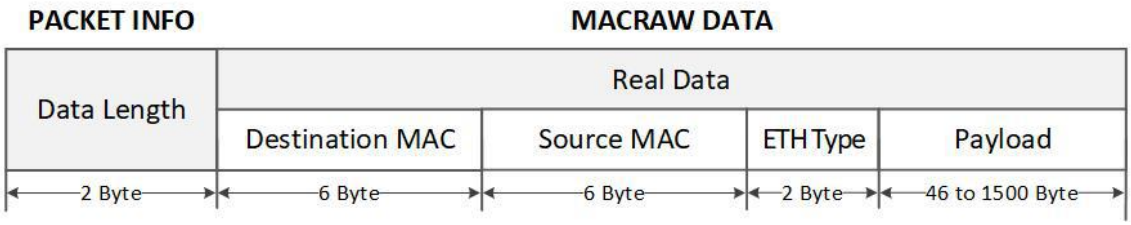


图 26 在 MACRAW 中接收数据的格式

```
{
/*接收 PACKINFO */
  goto  6.2.1 TCP: 接收处理 with get_size = 2;

/*在 PACKET INFO 中提取大小*/
data_size = (destination_address[0] << 8) + destination_address[1];

/* 读取 MACRAW 数据*/
goto  6.2.1 TCP: 接收处理 with get_size = data_size;
}
```

- **发送数据/发送处理（Sending DATA? / Sending Process）**

要发送的数据不得超过 SOCKET 0 TX 缓冲区大小。 如果数据大于 MSS，必须由 MSS（1512）划分。如果收送的数据小于 60 字节，则数据为‘0‘，填充为 60 字节。

参考 6.2.1TCP 服务器: 发送数据/ 发送处理

- **完成/关闭（Finished? / CLOSE）**

参考 6.3.1 UDP 单播: 完成/关闭

6.6 SOCKET-less 命令（SLCR）

SOCKET-less 命令（SLCR）在不使用 SOCKET 资源的情况下发送特定的包，如 ARP 请求，PING 请求，NS 和 RS。 可以通过 SLIR 检查对请求数据包的响应。 在重传时间到期之前没有响应时设置 SLIR [TOUT]。参考 *6.7 重传*。

多个 SLCR 命令不能同时执行。

设置一点 SLIR 之后，就可以执行下一个命令。图 27 显示了 SOCKET-less 的命令操作流程。

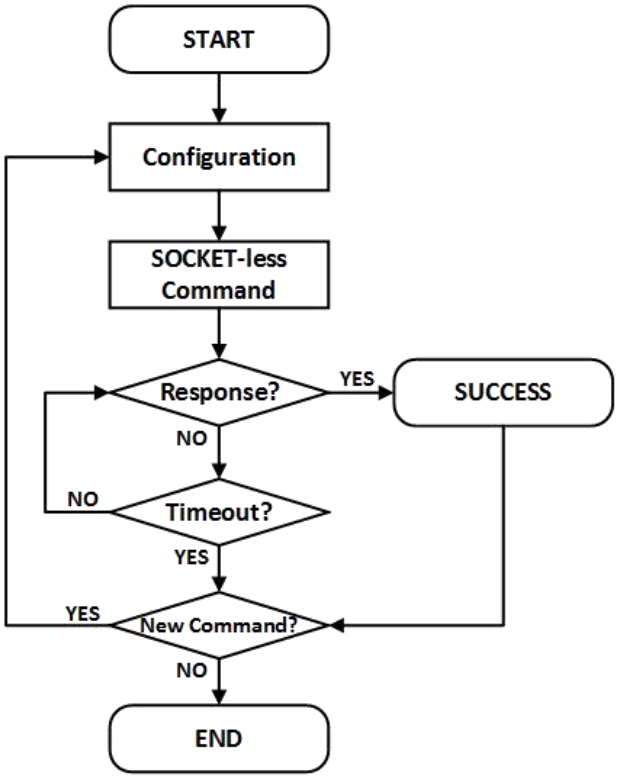


图 27 SOCKET-less 命令操作流程

6.6.1 ARP

SLCR [ARP]将 ARP 请求包发送到 SLDIPR 指定的目的地。 如果从设备收到 ARP 回复，则设置 SLIR [ARP]并将目标 MAC 地址保存在 SLDHAR 中。 如果在重传时间到期之前没有响应，则设置 SLIR [TOUT]。参考 6.7 重传。

- **配置（Configuration）**

配置重传时间，ARP 和 TOUT 中断掩码和目标 IP 地址，

```
{
开始 :
/*设置 SOCKET-less 重传时间, 100ms(0x03E8) (Unit 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[TOUT] = '1'; /* TIMEOUT 中断屏蔽位 */

/* 设置目标 IP 地址, 192.168.0.100 */
SLDIP6R[12:15] = {0xC0, 0xA8, 0x00, 0x64};
```

```
}
```

- **SOCKET-less 命令**

SLCR [ARP]命令发送 ARP 请求。

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

- **响应（Response?）**

如果目标有 ARP 响应，则设置 SLIR [ARP]。

```
{
/*检查 ARP 中断*/
if(SLIR[ARP] == ‘1’) /*收到 ARP 响应报文 */
{
SLIRCLR[ARP] = ‘1’; /* 清除中断*/
goto SUCCESS;
}
else goto Timeout?;
}
```

- **超时（Timeout?）**

如果在重传时间内对方没有收送对的 ARP 请求数据包的响应，则设置 SLIR [TOUT]。

```
{
/* 检查 TIMEOUT 中断 */
if(SLIR[TOUT]  == 1)
{
SLIRCLR[TOUT] = ‘1’; /* 清除中断*/
goto END;
}
else goto Response?;
}
```

- **成功（SUCCESS）**

如果收到 ARP 响应，目标 MAC 地址存储在 SLDHAR 中

```
{
dst_haddr [0:5] = SLDHAR[0:5]; /*获取目标 MAC 地址*/
goto END;
}
```

6.6.2 PING

SLCR[PING]将 ARP 和 PING 请求包同时发送到 SLDIPR 指定的目标 IP 地址。如果从目标接收到 ARP 应答和 PING 应答，则设置 SLIR[PING]，并将目标 MAC 地址保存在 SLDHAR 中。如果在重传时间内没有响应，则设置 SLIR[TOUT]。

配置（Configuration）配置重传时间、PING 和 TOUT 中断掩码、目标 IP 地址以及 PING 请求报文的序列号和 ID。

```
{
开始:
/*设置 SOCKET-less 重传时间, 100ms(0x03E8)(Unit 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[TOUT] = '1'; /* TIMEOUT 中断屏蔽位 */

/* 设置目标 IP 地址, 192.168.0.100 */
SLDIPR[12:15] = {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**

SLCR[PING]命令发送 PING 请求包。

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

- **响应（Response?）**

如果有来自目标的 PING 应答包，则设置 SLIR[PING]。

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

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

•超时/成功（Timeout? / SUCCESS）

参考 6.6.1 ARP :超时/成功

6.6.3 ARP6 (ND, Neighbor Discovery)

SLCR [ARP6]将 ICMPv6 NS（Neighbor Solicitation）数据包发送到 SLDIP6R 指定的目标 IP 地址，类似于 ARP-process。 如果收到从目标发送的 NA（Neighbor Advertisement），则设置 SLIR [ARP6]并在 SLDHAR 中保存目标 MAC 地址。 如果在重发时间内没有响应，则设置 SLIR [TOUT]。参考 *6.7 重传*。

• 配置（Configuration）

配置重传时间，ARP6 和 TOUT 中断掩码和目标 IP 地址。

```
{
开始:
/* 设置 t SOCKET-less 重传时间, 100ms(0x03E8) (Unit 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[ARP6] = ‘1’; /* ARP6 中断屏蔽位*/
//SLIMR[TOUT] = ‘1’; /* TIMEOUT 中断屏蔽位*/

/*设置目标 IP 地址, FE80::1D0:AABB:CCDD */
SLDIP6R[0:15] = { 0xFE, 0x80, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,
0x13, 0x22, 0x33, 0xFF, 0xFE, 0xAA, 0xBB, 0xCC };
}
```

• SOCKET-less 命令

SLCR [ARP6]命令发送 NS 数据包

```
{
SLCR[ARP6] = ‘1’; /* 设置 ARP6 命令*/
while(SLCR != 0x00) ; /*等待 ARP6 命令完成*/
```

```
}
```

- **响应（Response?）**

如果有来自目标发送的的 NA 包，则设置 SLUR [ARP6]。

```
{
/*检查 ND 中断*/
if(SLIR[ARP6] == ‘1’) /* 收到 NA 包 */
{
SLIRCLR[ARP6] = ‘1’; /* 清除中断 */
goto SUCCESS;
}
else goto Timeout;
}
```

- **超时（Timeout?）**

如果目标没有发送 NA 数据包，则设置 SLIR[TOUT]。

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

- **成功（SUCCESS）**

目标 MAC 地址保存在 SLDHAR 中。

```
{
dst_haddr[0:5] = SLDHAR[0:5]; /*获取目标 MAC 地址*/
goto END;
}
```

6.6.4 PING6 (ICMPv6 Echo)

SLCR [PING6]将 ICMPv6 NS 和 PING 请求包发送到 SLDIP6R 指定的目标 IP 地址。 如果从目标接收到 ICMPv6 NA 和 ICMPv6 Echo PING 应答，则设置 SLIR [PING6]并将目标 MAC 地址保存在 SLDHAR 中。 如果在重发时间内没有响应，则设置 SLIR [TOUT]。参考 *6.7 重传*。

- **配置（Configuration）**

配置重传，PING6 和 TOUT 中断屏蔽，目标 IP 地址。

```
{
开始 ：
```

```
/*设置 SOCKET-less 重传时间, 100ms(0x03E8) (unit 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[PING6] = '1';    // PING6 中断屏蔽位
//SLIMR[TOUT] = '1';   // TIMEOUT 中断屏蔽位

/*设置目标 IPv6 地址, FE80::1D0:AABB:CCDD */
SLDIP6R[0:15] = { 0xFE, 0x80, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,
0x13, 0x22, 0x33, 0xFF, 0xFE, 0xAA, 0xBB, 0xCC };
}
```

• **SOCKET-less 命令**

SLCR [PING6]命令发送 NS 包和回应请求数据包。

```
{
SLCR[PING6] = '1'; /* 设置 PING6 命令*/
while(SLCR != 0x00) ; /*等待 PING6 命令完成*/
}
```

• **响应（Response?）**

如果目标有 echo 应答，则设置 SLIR[PING6]。

```
{
/* 检查 PING6 中断*/
if(SLIR[PING6] == '1') /*收到 PING6 包*/
{
SLIRCLR[PING6] = '1'; /* 清除中断 */
goto SUCCESS;
}
else goto Timeout?;
}
```

• 超时/完成（**Timeout? / SUCCESS**）

参考 ***6.6.3 ARP6(ND,Neighbor Discovery):超时/完成***

6.6.5 DAD (重复地址检测)

SLCR [NS]对由 SLDI6PR 指定的目标 IP 地址执行 DAD（重复地址检测）机制。 SLCR [NS]发送 DAD NS 数据包。如果目标发送 DAD NA 数据包，则设置 SLIR[NS]，并且目标 IP 地址作为源 IPv6 地址无效。 如果在重传时间内目标没有发送 DAD NA 数据包，则设置 SLIR[TOUT]，目标 IP 地址作为源 IPv6 地址有效。参考 6.7 重传。

图 28 显示 DAD 操作流程.

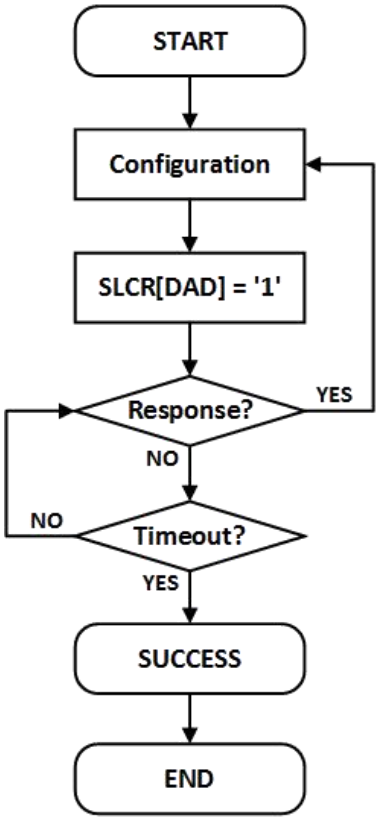


图 28 DAD 操作流程

- **配置（Configuration）**

配置重传时间、NS & TOUT 中断掩码和目标 IP 地址。

```
{
开始 :
/* 设置 SOCKET-less 重传时间, 100ms(0x03E8) (단위, 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[NS] = '1'; /* NS 中断屏蔽位*/
//SLIMR[TOUT] = '1'; /* TIMEOUT 中断屏蔽位 */

/*设置目标 IP 地址, FE80::1D0:AABB:CCDD */
SLDIP6R[0:15] = { 0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0xD0, 0xAA, 0xBB, 0xCC, 0xDD };
```

```
}
```

• **SOCKET-less 命令**

由 SLCR[NS]发送 DAD NS 数据包。

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

• **响应（Response?）**

如果目标发送 DAD NA 数据包，则设置 SLIR[NS]， SLDIPR 作为源 IPv6 地址无效。

```
{
/* 检查 NS 中断 */
if(SLIR[NS] == '1') /*收到 DAD NA 包*/
{
SLIRCLR[NS] = '1'; /*清除中断*/
goto Configuration;
}
else goto Timeout?;
}
```

• **超时（Timeout?）**

如果在重传时间内目标没有收到 DAD NA 包，则设置 SLIR[TOUT]，重传成功。

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

• **成功（SUCCESS）**

SLDIP6R 可用作源 IPv6 地址。

```
{
LLAR[0:15] = SLDIP6R[0:15]; /*获取源 link 本地地址*/
goto END;
}
```

6.6.6 RS (路由器请求)

SLCR [RS]发送 RS（路由器请求）数据包以链接本地所有路由组多播地址（FF02 :: 2）。如果路由器有 RA 报文，则设置 SLIR [RS]，RA 报文的前缀长度，标志，有效生存期，前缀生存时间和前缀地址分别保存在 PLR，PFR，VLTR，PLTR 和 PAR 中。 如果在重传时间内没有来自路由器的 RA 数据包，则设置 SLIR [TOUT]。 参考 *6.7 重传*。

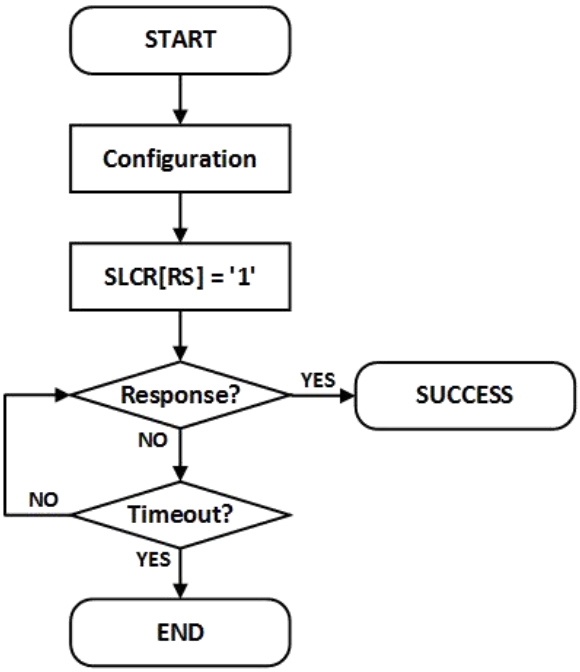


图 29 RS 操作流程

*警告 ：当接收到 RA 时，PLR、PFR、VLTR、PLTR 和 PAR 值未正确的处理，但是 RA 选项字段的第一种类型不是源链路层地址(0x01)，第二种类型不是前缀信息(0x03)。
在这种情况下，使用 IPRAW6 SOCKET 接收 RA 的数据。

• **配置（Configuration）**

配置重传时间，RS 和 TOUT 中断屏蔽，路由器 IP 地址。

```
{
开始 :
/* 设置 SOCKET-less 重传时间, 100ms(0x03E8) (단위, 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[RS] = ‘1’;        // RS 中断屏蔽位
//SLIMR[TOUT] = ‘1’;   // TIMEOUT 中断屏蔽位
}
```

• **SOCKET-less 命令**

由 SLCR [RS]发送 RS 数据包。

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

- **响应（Response?）**

如果路由器有 RA 数据包，则设置 SLIR [RS]。

```
{
/*检查 RS 中断*/
if(SLIR[RS] == ‘1’) /*收到 RA 包*/
{
SLIRCLR[RS] = ‘1’; /* 清除中断 */
goto Configuration;
}
else goto Timeout?;
}
```

- **超时（Timeout?）**

如果在重传时间内没有 RA 数据包，则设置 SLIR [TOUT]。

```
{
/*检查 TIMEOUT 中断 */
if(SLIR[TOUT]  == 1)
{
SLIRCLR[TOUT] = ‘1’; /* 清除中断 */
goto SUCCESS;
}
else goto Response?;
}
```

- **成功（SUCCESS）**

RA 数据包的前缀长度，标志，有效生存期，前缀生存期和前缀地址分别保存在 PLR，PFR，VLTR，PLTR 和 PAR 中。

```
{
    Prefix_length = PLR; /* RA 前缀长度*/
    Flags = RAFLGR; /* RA 标志*/
    Valid_Lifetime = VLTR; /* RA 有效生存期*/
    Prefix_Lifetime = PLTR; /* RA 前缀生存期 */
    Prefix_address[0:15] = PAR[0:15]; /* RA 前缀地址 */
}
```

6.6.7 未经请求的 NA(Neighbor Advertisement)

SLCR[NA]发送未经请求的 NA 数据包。目标地址自动配置 FF02::1(全节点组播地址)，目标地址根据 SLPR 由 LLAR 或 GUAR 自动配置。因为未经请求的 NA 是一个无响应的消息，所以在传输完成时设置 SLIR [TIOUT]。图 30 显示未经请求的 NA 操作的流程。

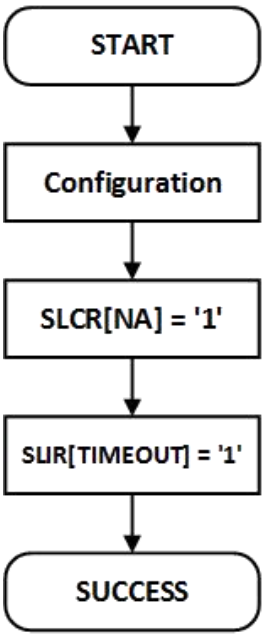


图 30 未经请求的 NA 操作流程

- **配置（Configuration）**

配置目标地址，地址类型和 TOUT 中断屏蔽。

```
{
开始 :
if (Target Address is Link Local Address)
{
LLAR[0:15] = { 0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x34, 0x56 };
      SLPR = 0x10;
}
else /*目标地址是全球单播地址*/
{
GUAR[0:15] = { 0x20, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x34, 0x56 };
           SLPR = 0x11;
}

     /*设置 SOCKET-less TIMEOUT 中断屏蔽位*/
     SLIMR[TOUT] = ‘1’;
```

```
}
```

• **SOCKET-less 命令**

SLCR[NA]向所有节点发送未经请求的 NA 包。

```
{
SLCR[NA] = '1';   /* 设置未经请求的 NA 命令 */
while(SLCR != 0x00) ; /*等到未经请求的 NA 命令完成*/
}
```

• **超时（Timeout）**

传输完成后设置 SLIR [TOUT]。

```
{
/*检查 TIMEOUT 中断*/
if(SLIR[TOUT]   == 1)
{
SLIRCLR[TOUT] = '1'; /* 清除中断 */
goto SUCCESS;
}
}
```

6.7 重传

6.7.1 ARP 或 PING 或 ND 重传

当对方没有响应 ARP / PING / ND 分组时，执行 ARP / PING / ND 重传。在重传过程中，每个 RTR 都重传请求包，直到接收到响应包为止。如果重传次数超过 RCR，则发生 TIMEOUT。下表显示了重传 TIMEOUT（ARPTO，PINGTO，NDTO）。

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

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

例) $TIMEOUT_{VAL}$ = 2000(0x07D0), $TIMEOUT_{CNT}$ = 8(0x0008)

ARP_{TO}= 2000 X 0.1ms X 9 = 1.8s

当在 TCP4 模式下 Sn_CR[CONNECT]、UDP4 和 IPRAW4 模式下 Sn_CR[SEND]、SOCKET-less 命令中 SLCR[ARP]没有来自 ARP-prcoess 目标的响应时，就会发生 ARP_{TO}。
Sn_IR[TIMEOUT]或 SLIR[TOUT]由 ARP_{TO} 设置。

当 ARP-process 中没有来自目标的 SLCR[PING]和 SLCR[PING6]的响应，或者在 ARP-process 之后没有来自标的 PING 响应时，就会发生 $PING_{TO}$。
$PING_{TO}$ 由 $PING_{TO}$ 设置。

NDTO 发生在 ND 过程中，Sn_CR[CONNECT6]在 TCP6 & TCPD 模式下，Sn_CR[SEND6]在 UDP6 & UDPD & IPRAW6 模式下，SLCR[ARP6] & SLCR[NS] & SLCR[RS]在 SOCKET-less 命令中。
Sn_IR[TIMEOUT]或 SLIR[TOUT]由 NDTO 设置。

6.7.2 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 重传超时(*TCP$_{TO}$*)。

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

N : Retransmission Counter, $0 \le N \le M$

M : $\mathrm{TIMEOUT_{VAL}} \times 2^{(M+1)} > 65535$ and $0 \le M \le \mathrm{TIMEOUT_{CNT}}$

$\mathrm{TIMEOUT_{VAL}}$ = Sn_RTR

$\mathrm{TIMEOUT_{CNT}}$ = Sn_RCR

$\mathrm{TIMEOUT_{MAXVAL}}$: $\mathrm{TIMEOUT_{VAL}} \times 2^{M}$

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

TCP$_{TO}$=(0x07D0+0x0FA0+0x1F40+0x3E80+0x7D00+0xFA00+0xFA00+0xFA00+0xFA00)X0.1ms

= (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、CONNECT6、SEND、SEND6 和 DISCON 命令执行，Sn_IR [TIMEOUT]由 *TCP$_{TO}$* 设置。

6.8 其他功能

6.8.1 系统时钟切换(SYS_CLK)

SYS_CLK 设置为 25MHz 或 100MHz。它由 SYCR1[CLKSEL]，PHYCR1[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 *6.7 重传。*

SYCR1[CLKSEL]	PHYCR1[RST]	PHYCR1[PWDN]	SYS_CLK(MHz)
0	0	X	25
0	1	0	100 (Default)
0	1	1	25
1	X	X	25

6.8.2 以太网 PHY 操作模式配置

PHY 操作模式（速度，双工）由 PHYCR0 设置，并在以太网 PHY HW 复位后生效。 在以太网 PHY 链接之后，可以使用 PHYSR [5：3]检查 PHY 操作模式，并使用 PHYSR [2：0]链接状态。只有在解锁 PHYLCKR 时才能配置 PHYCR0。

例）设置 PHY 操作模式

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

/*设置 PHYCR0 100 / 10BASE 和全/双工*/
phy_mode = ‘000’    // 自动协商
//phy_mode = ‘100’    // 100BASE-TX FDX
//phy_mode = ‘101’    // 100BASE-TX HDX
//phy_mode = ‘110’    // 10BASE-TX FDX
//phy_mode = ‘111’    // 10BASE-TX HDX
PHYCR0[2:0] = phy_mode;

/* PHY 复位过程*/
PHYCR1[RST] = ‘1’;
Wait   TPRST;    // 参考 8.4.1 复位时序

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

    /*等待 PHY 链接启动*/
```

```
while(PHYSR[LNK] != ‘0’);

    /* 读取 PHYSR */
    If( (PHYSR[5:3] == phy_mode) ) SUCCESS;
else FAIL;
}
```

6.8.3 以太网 PHY 并行检测

如果链接伙伴不支持自动协商模式，则 W6100 的嵌入式以太网 PHY 通过并行检测建立链路。

注意 10F / 10H 等双工模式不匹配可能会降低网络性能。

PHY \ Link Partner	Auto	10H	10F	100H	100F
Auto	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

6.8.4 以太网 PHY Auto-MDIX

当以太网 PHY 设置为自动协商（PHYCR0 [MODE2] ='0'）时，W6100 支持 Auto-MDIX，并操作对称变压器（图 43 变压器类型）。

如果没有设置自动协商功能，则由于不支持 Auto-MDIX 功能，因此必须使用交叉 UTP 电缆。

注意：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。

6.8.5 以太网 PHY 掉电模式

当 PHYCR1[PWDN] = ' 1 '时，以太网 PHY 进入掉电模式，SYS_CLK 设置为 25MHz。在 PHYCR1 [PWDN] ='0'的情况下，以太网 PHY 进入正常模式，SYS_CLK 由 SYCR1[CLKSEL]设置。 参考 *4.1.5 SYCR1 (系统配置寄存器 1).*

```
进入掉电模式
{
/* PHYCR0 和 PHYCR1 解锁*/
PHYLCKR = 0x53;
```

```
    /*启用掉电模式*/
PHYCR1[PWDN]   = '1';

/* PHYCR0 和 PHYCR1 锁定*/
PHYLCKR = 0x00; //对于 Lock，写入任何值

/*等待时钟稳定切换*/
Wait TPRST; // 参考 8.4.1 复位时序
}
```

```
退出关机模式：{
/* PHYCR0 和 PHYCR1 解锁 */
PHYLCKR = 0x53;

    /*启用掉电模式*/
PHYCR1[PWDN]   = '0';

/* PHYCR0 和 PHYCR1 锁定*/
PHYLCKR = 0x00; //对于 Lock，写入任何值

/*等待时钟稳定切换*/
Wait TPRST; // 参考 8.4.1 复位时序

/* wait until Clock is switched 25 to 100MHz*/
Wait TLF; // refer to 8.4.1 复位时序
}
```

6.8.6 以太网 PHY 控制寄存器

以太网 PHY 寄存器可以通过 MDC/MDIO(管理数据时钟/输入输出)接口访问。W6100 集成了 MDC/MDIO 控制器，主机通过 PHYDIVR、PHYRAR、PHYDOR、PHYDIR 和 PHYACR 控制它。图 31 显示 MDC/MDIO 写控制流程。

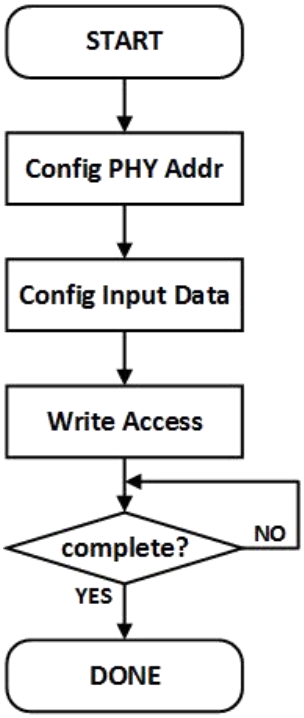


图 31 MDC/MDIO 写控制流程

•配置 PHY 寄存器地址

存储 PHY 寄存器的地址以访问 PHYRAR。

```
{
开始:
    /*将 PHY 寄存器地址设置为 PHYRAR */
    PHYRAR = 0x00;   /* BMCR 地址为 0x00 */
}
```

•配置输入数据

将要写入 PHY 寄存器的 16 位数据存储到 PHYDIR0 和 PHYDIR1 中。PHYDIR1 中最高位的 8 位数据/ PHYDIR0 中最低位 8 位数据。

```
{
/*声明 16bits 变量*/
    Data = 0x8000;   /*在 BMCR 中设置 RST 位*/

PHYDIR1 = (Data & 0xFF00) >> 8; /*设置高 8 位数据*/
PHYDIR0 = Data & 0x00FF;        /*设置低 8 位数据*/
}
```

• 写访问/完成（Write Access / Complete?）

如果 PHYACR 设置为“0x01”，则 PHYDIR 中的数据将写入 PHY 寄存器，PHY 寄存器在 PHYRAR 中指定。 PHYACR 将自动清除。

```
{
PHYACR = 0x01;     /*设置写访问权限*/
```

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

图 32 显示 MDC / MDIO 读取控制流程。

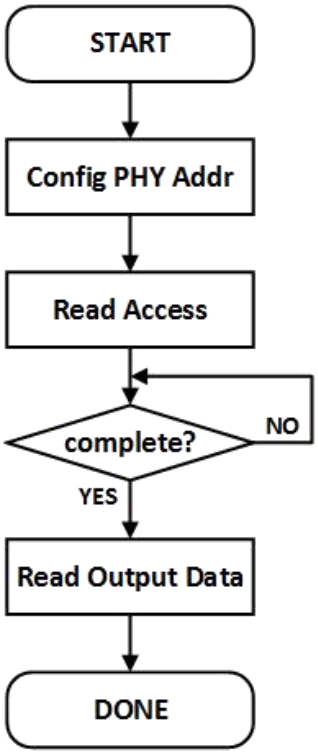


图 32 MDC/MDIO 读取控制流程

- **配置 PHY 寄存器地址（Config PHY Register Address）**

存储 PHY 寄存器的地址访问 PHYRAR。

```
{
开始:
    /*将 PHY 寄存器地址设置为 PHYRAR */
    PHYRAR = 0x01;   /* BMSR 地址为 0x01 */
}
```

- **读访问/完成（Read Access / Complete）**

如果 PHYACR 设置为“0x02”，PHY 寄存器中的数据（在 PHYRAR 中指定）将传输到 PHYDOR。 PHYACR 将自动清除。

```
{
PHYACR = 0x02;      /*设置读访问权限*/
while(PHYACR != 0); /*等待 MDC / MDIO 控制完成*/
}
```

- **读取输出数据（Read Output Data）**

PHY 寄存器中的数据存储在 PHYDOR0 和 PHYDOR1 中。PHYDOR1 中高 8 位有效/ PHYDOR0 中低 8 位有效。

```
{
    Data = (PHYDOR1 & 0x00FF) << 8;     /*获取高 8 位数据*/
    Data = Data + (PHYDOR0 & 0x00FF); /*获取低 8 位数据*/
}
```

6.8.7 以太网 PHY 10BASE-Te 模式

W6100 以太网 PHY 可以在 10BASE-Te 模式下运行，以下是设置步骤。

```
{
/* PHYCR0 和 PHYCR1 解锁*/
PHYLCKR = 0x53;

/*启用自动协商*/
PHYCR0[2:0] = '000';

/*  设置 PHY Te 模式  */
PHYCR1[TE] = '1';

/* PHY 复位过程*/
PHYCR1[RST] = '1';
Wait TPRST;     // 参考 8.4.1 复位时序
}
```

7 时钟和变压器要求

7.1 无源晶振要求

表 4 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 条件下	-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	W6100 复位			60	ms
Trans-conductance(g_m)			16.7		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.

- C_0, C_L, ESR and DL are provided by the crystal manufacturer.
- The W6100 has no feedback resistor. Therefore, it must be inserted outside.

* 图 33 无源晶振电路模型图。

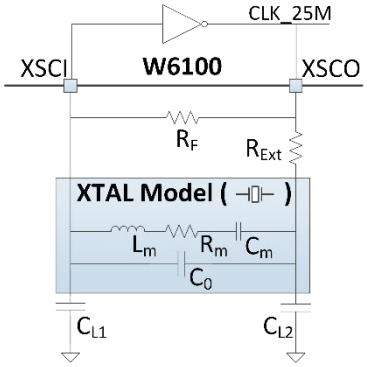


图 33 无源晶振电路模型图

表 7 无源晶振推荐特性

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

7.2 有源晶振特性

表 8 有源晶振特性

参数	条件/ 描述	最小	典型	最大	单位
频率			25		MHz
频率误差	25°C 条件下	-50		+50	ppm
频率稳定性	25°C 条件下 1 年老化速度	-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	10% 到 90% 的波形			8ns	
启动时间		-	-	10ms	
工作电压		1.08V	1.2V	1.32V	
老化速度(25℃条件下)			最大±3/年		ppm

7.3 变压器特性

表 9 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

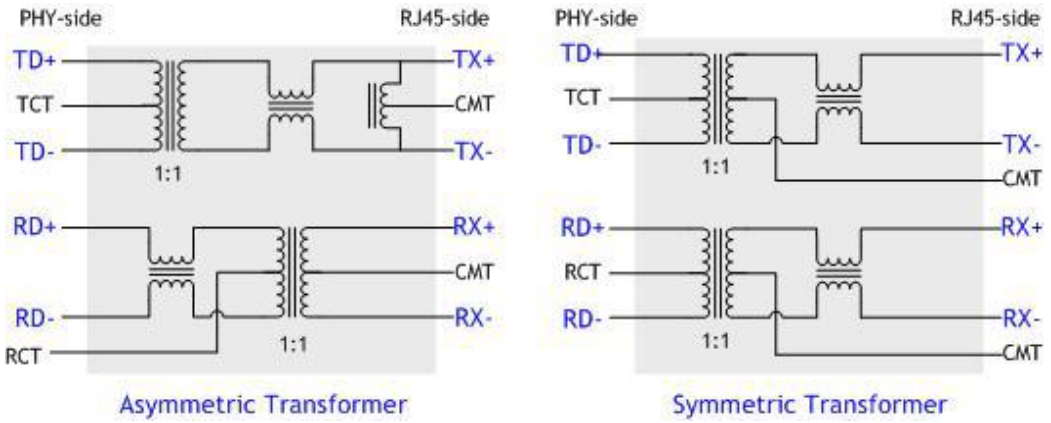

PHY-side
RJ45-side
TD+
TCT
TD-
TX+
CMT
TX-
1:1
RD+
RD-
RCT
RX+
CMT
RX-
1:1
Asymmetric Transformer
PHY-side
RJ45-side
TD+
TCT
TD-
TX+
TX-
CMT
1:1
RD+
RCT
RD-
RX+
RX-
CMT
1:1
Symmetric Transformer

图 34 变压器特性

8 电气规范

8.1 最大额定值

表 10 最大额定值

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	℃
T_{STG}	存储温度	-65 ~ +150	℃

*注释: 超过最大额定值的操作可能会引起元器件的永久损伤.

8.2 最大极限值 (ESD)

表 11 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 条件下符合 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 条件下符合 JEDEC EIA/JESD22 A115-A	B	200	V
V_{ESD} CDM	静电放电电压（充点装置模型）	TA = +25 °C 条件下符合 JEDEC JESD22 C101-C	III	500	V

表 12 闩锁测试

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

8.3 直流特性

表 13 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	提供 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	除模拟引脚之外的所有输入	0.8	1.1	-	V
V_{T-}	施密特触发器从高到低阈值点	除模拟引脚之外的所有输入	-	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	KΩ
R_{PD}	下拉电阻		30	75	190	KΩ
V_{OL}	低电平输出电压	IOL = 2.0mA ~ 8.0mA，除 XSCO 以外的所有输出			0.4	V
V_{OH}	高电平输出电压	IOH = 2.0m ~ 8.0mA，除 XSCO 以外的所有输出	2.4			V

8.4 交流特性

8.4.1 复位时序

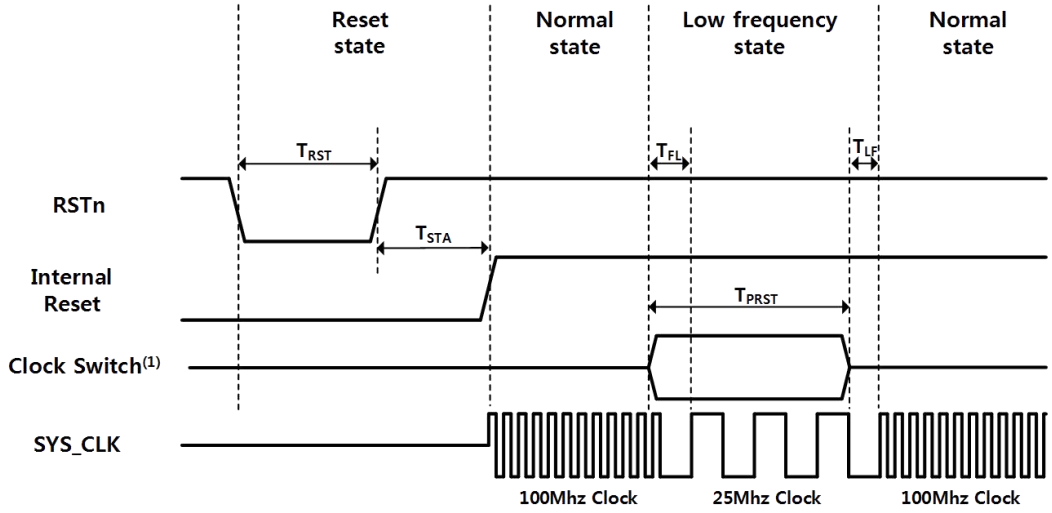


图 35 复位时序

表 14 复位时序表

符号	描述	最小	典型	最大
T_{RST}	复位时间	350 ns	580 ns	1.0 us
T_{STA}	稳定时间	-		60.3 ms
T_{FL}	时钟从高频率到低频率的时间，通过 MR2[CLKSEL]触发	100 ns		-
	时钟从高频率到低频率的时间，通过 PHYCR1[RST] 或者 PHYCR1[PWDN]触发	300 ns		
T_{PRST}	PHY 自动复位时间	0.6 ms		-
	PHY 掉电时间	200 us		
	时钟切换时间	200 ns		
T_{LF}	时钟从低频率到高频率的时间，通过 MR2[CLKSEL]触发	100 ns		-
	时钟从低频率到高频率的时间，通过 PHYCR1[RST] 或 PHYCR1[PWDN]触发	100 ns		

**注释: PHY 掉电模式具有 T_{FL} 和 T_{LF} (在 PHY 掉电模式下,)SYS_CLK 切换到低时钟， 在 T_{FL} 后 用户可以禁用 PHY 的掉电模式。*

**注意: 用户不能同时设置 PHY 自动复位和 PHY 掉电模式*

8.4.2 总线访问时序

8.4.2.1 读时序

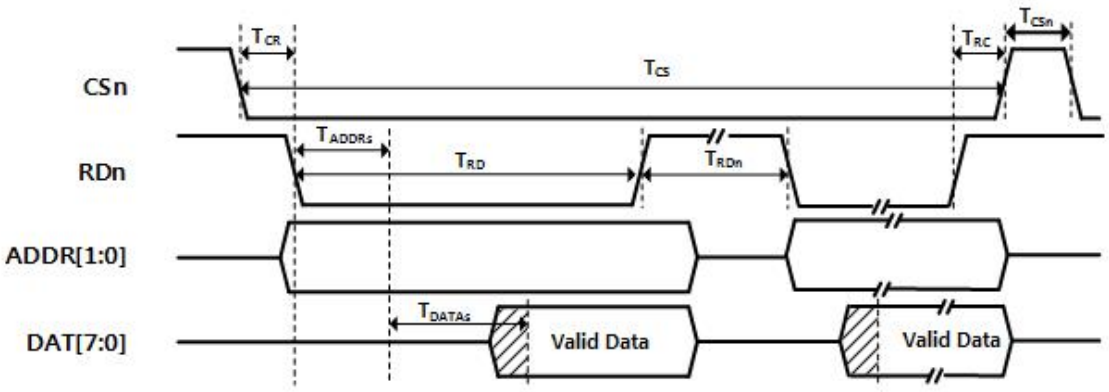


图 36 总线读时序

表 15 总线读时序

符号	描述	最小	最大
T_{ADDRs}	地址设置时间	SYS_CLK	
T_{CR}	CSn 拉低到 /RD 拉低的时间	0 ns	
T_{cs}	CSn 拉低时间	4 SYS_CLK	
T_{RC}	RDn 拉高到 CSn 拉高的时间	0 ns	
T_{csn}	CSn 下次拉低的时间	3 SYS_CLK	
T_{RD}	RDn 拉低时间	4 SYS_CLK	
T_{RDn}	RDn 下次拉低的时间	3 SYS_CLK	
T_{DATAs}	数据设置时间	3 SYS_CLK+5ns	

8.4.2.2 写时序

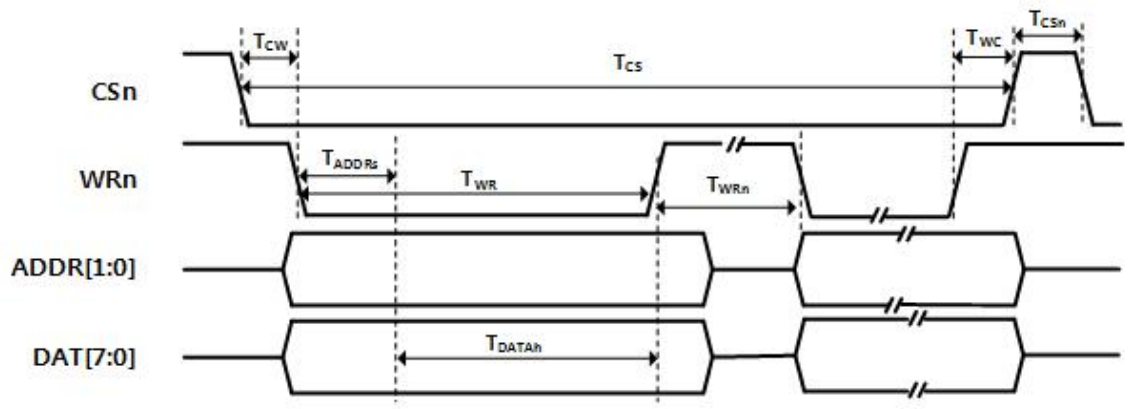


图 37 总线写时序

表 16 总线写时序

符号	描述	最小	最大

T_{ADDRs}	地址设置时间	SYS_CLK	
T_{CW}	CSn 拉低到 WRn 拉低时间	0 ns	
T_{cs}	CSn 拉低时间	4 SYS_CLK	
T_{WC}	WRn 拉高到 CSn 拉高时间	0 ns	
T_{csn}	CSn 下次拉低时间	3 SYS_CLK	
T_{WR}	WRn 拉低时间	4 SYS_CLK	
T_{WRn}	WRn 下次拉低时间	3 SYS_CLK	
T_{DATAs}	数据设置时间	2 SYS_CLK	

8.4.3 SPI 访问时序

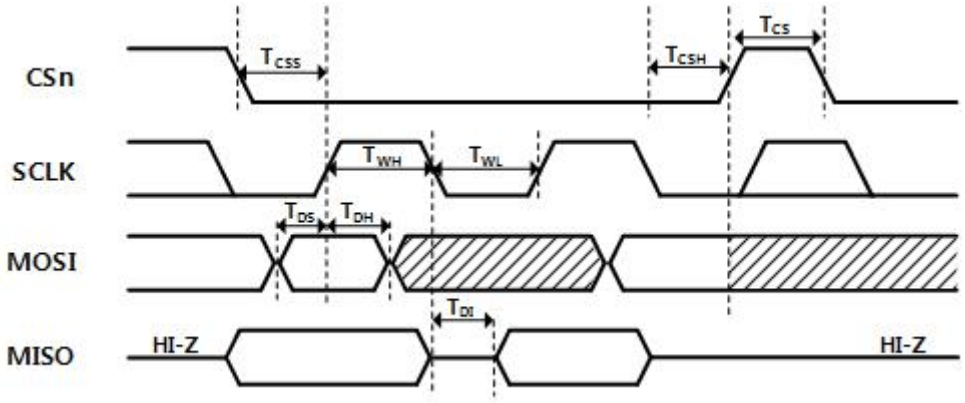


图 38 SPI 访问时序

表 17 SPI 访问时序

符号	描述	最小	最大	单位
F_{SCLK}	SCLK 时钟频率		70	MHz
T_{CSS}	CSn 设置时间	3 SYS_CLK		ns
T_{CSH}	CSn 保持时间	2 SYS_CLK		ns
T_{CS}	CSn 拉高时间	2 SYS_CLK		ns
T_{WH}	SCLK 拉高时间	3		ns
T_{WL}	SCLK 拉低时间	3		ns
T_{DS}	数据设置时间	3		ns
T_{DH}	数据保持时间	3		ns
T_{DI}	无效数据时间	7		ns

8.4.4 变压器特性

表 18 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

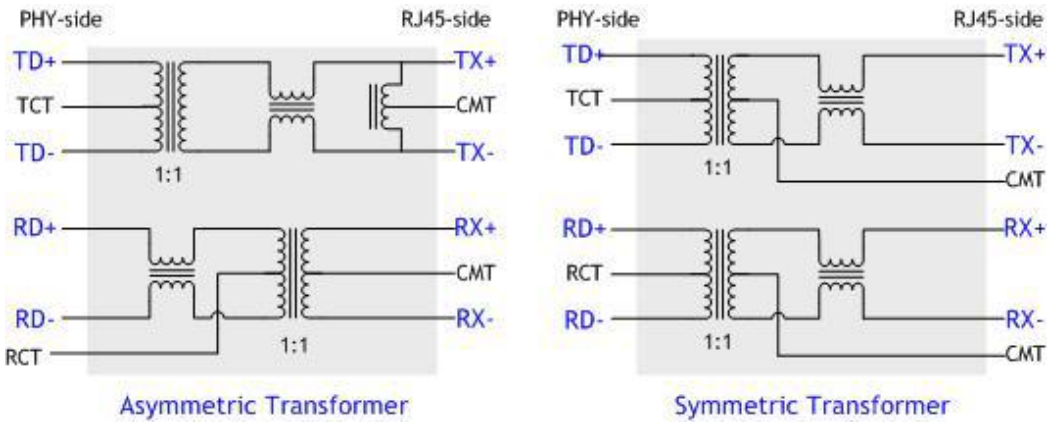


图 39 变压器特性

8.4.5 MDIX

W6100 仅在自动协商模式下支持 Auto-MDIX。

8.5 功耗

表 19 功耗

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

条件	最小值	典型值	最大值	单位
100M Link	-	93	110	mA
10M Link	-	150	170	mA
10M-Te Link		130	150	mA
100M Unlink (实际测量)	-	45		mA
10M Unlink (实际测量)	-	17		mA
10M-Te Unlink (实际测量)		17		mA
Un-Link (自动协商模式下) (实际测量	-	43	-	mA
掉电模式	-	17	-	mA

9 封装

9.1 LQFP48 封装

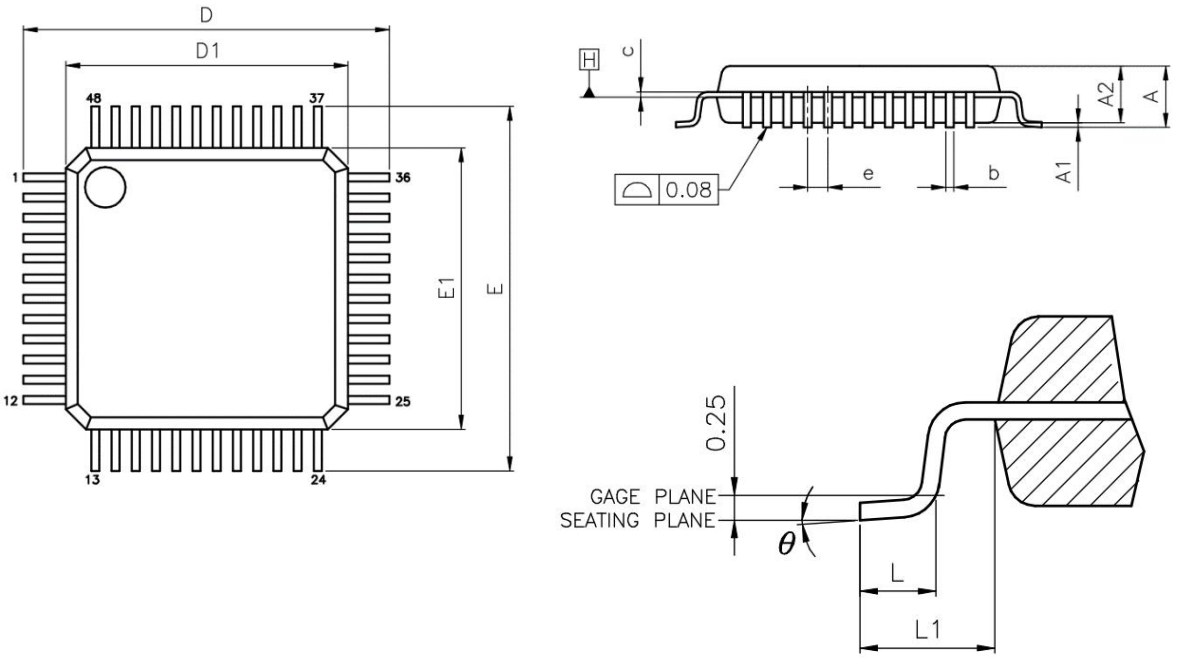


表 20 LQFP48

(单位：MM)

符号	最小值	标准值	最大值
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.

9.2 QFN48 封装

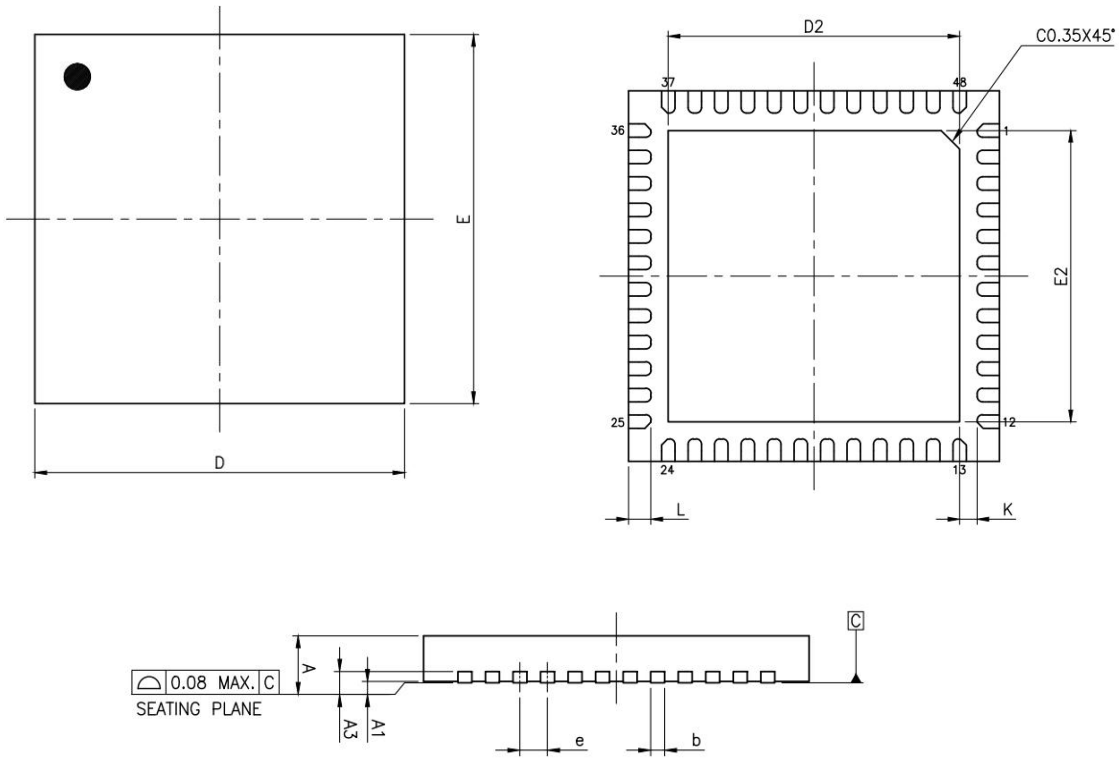


表 21 QFN48

(单位：MM)

符号	最小值	标准值	最大值
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 TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, 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.

文档修订历史

版 本	日期	描 述
V1.0.0	2019.05.10	中文初版发布

技术支持 e-mail：support@hschip.com
技术支持 QQ：2232725509

销售：

浩然成都公司：四川省成都市高新区天和路 69 号皇城花卉 8 栋 2503 室
电话：028-86127089，86120475
传真：028-86127039
商务 QQ：402856564

浩然深圳公司：深圳市福田区皇岗路高科利大厦 A 座 28B
电话：0755-86066647, 18575510340
商务 QQ：272805085
更多信息,请登录 http://www.hschip.com