高性能Internet连接解决方案

W5300

Version 1.1.1

为了获取更多的信息，请访问

WIZnet网站：http://www.wiznet.co.kr

浩然电子 Hasion Electronics **浩然电子网站：**http://www.hschip.com

2008-7

WIZnet在线技术支持

如果你有关于WIZnet产品事情咨询，请访问：
http://www.wiznet.co.kr

或在技术支持的Q&A版
http://www.wiznet.co.kr/rg4_board/list.php?bbs_code=en_qna
写下你的问题，WIZnet的工程师会很快回答你的问题。

浩然电子在线技术支持

如果你有技术或产品方面的问题，请访问成都浩然电子网站：

http://www.hschip.com

如果你想获取由浩然电子提供的技术文档，详细的设计资料，请在浩然电子的网站注册，浩然电子将升级你为高级会员。然后你再以高级会员登陆，就可以免费下载各种技术资料和设计资料，

http://www.hschip.com/adduser.aspx

你还可以用中文/英文提出你的问题，浩然电子的工程师会很快回答你的问题。

http://www.hschip.com/lyb.aspx?id=4

电话：+86-28-86127089
传真：+86-28-86127039

W5300

W5300是一款0.18μm CMOS工艺的单芯片器件，内部集成10/100M以太网控制器，MAC和TCP/IP协议栈。W5300使用方便、稳定可靠，广泛应用于高性能、低成本的Internet嵌入式领域。

W5300的目标是在高性能的嵌入式领域，如多媒体数据流服务。与WIZnet现有的芯片方案相比较，W5300在内存空间和数据处理能力等方面都有很大的提高。W5300特别适用于IPTV，IP机顶盒和数字电视等大流量多媒体数据的传输。

通过一个集成有TCP/IP协议和10/100M的以太网MAC和PHY的单芯片可以非常简单和快捷地实现Internet连接。

高性能硬件TCP/IP单芯片解决方案

WIZnet拥有全硬件通信协议技术，如TCP、UDP、IPv4、ICMP、IGMP、ARP和PPPoE。为了实现高性能的数据通信，W5300的通信数据存储器扩展到128K字节，与MCU的接口支持16位数据总线。用户可以使用8个独立的端口进行高速数据通信。

根据不同的应用提供更灵活的存储器分配

每一个端口的通信数据存储器可以分配0-64K字节。用户可以根据不同的应用更灵活地分配存储空间。用户还可以通过集中高性能的配置而使系统具有更高的性能。

初学者易于使用

W5300与主机（MCU）采用总线接口。通过直接访问方式或间接访问方式，W5300可以很容易与主机接口，就像访问SRAM存储器。W5300的通信数据可以通过每个端口的TX/RX FIFO寄存器访问。由于这些特性，即使一个初学者也很容易使用W5300实现Internet连接。

应用领域

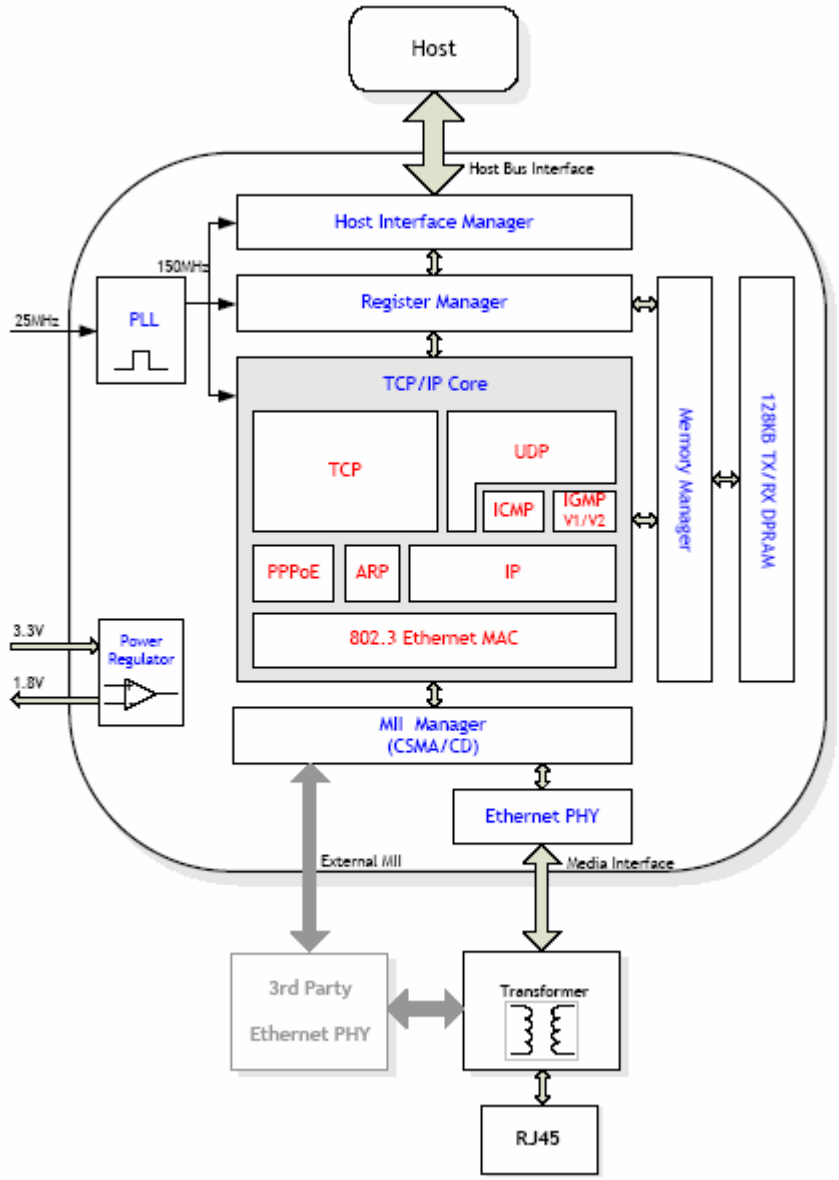
W5300在很多嵌入式系统中都非常适用，包括：

- 家庭网络设备：机顶盒，PVR和数字多媒体设备。
- 串口转以太网：访问控制，LED显示等
- 并行接口转以太网：POS/微型打印机，复印机
- USB转以太网：存储设备，网络打印机
- GPIO转以太网：家庭网络传感器
- 安防设备：DVR，网络摄像头
- 工厂和建筑自动化
- 医疗监控设备
- 嵌入式服务器

特性：

- 支持固件TCP/IP协议：TCP，UDP，ICMP，IPv4，ARP，IGMP，PPPoE，Ethernet
- 支持8个独立端口同时工作
- 高速网络数据传输，速率可达到50Mbps
- 支持混合网络TCP/IP协议栈（软件/硬件TCP/IP协议栈）
- 支持ADSL连接（支持带PAP/CHAP认证模式的PPPoE协议）
- 不支持IP分片功能
- 内部128K字节存储器用于数据通信（内部TX/RX存储器）
- 根据端口通信数据吞吐量动态调整内部TX/RX存储器的分配
- 支持存储器到存储器的DMA功能（只有16位数据总线宽度才支持，从模式）
- 内嵌10BaseT/100BaseTX的以太网物理层
- 支持自动握手功能（全双工，半双工）
- 支持自动MDI/MDIX（信号线极性交叉）
- 支持LED网络指示（TX，RX，全双工/半双工，IP地址冲突，网络连接和网络速度）
- 支持第三方物理（PHY）接口
- 支持8/16位数据总线
- 支持2种主机接口模式（直接访问模式和间接访问模式）
- 外部25MHz工作频率（给内部锁相环逻辑电路，周期40ns）
- 内部锁相环时钟输出150MHz（锁相环时钟，周期大约为6.67ns）
- 网络工作频率：25MHz（100BaseTX）或2.5MHz（10BaseT）
- 3.3V工作电压，I/O口可承受5V电压
- 内部带1.8V电压调整器
- 0.18um的CMOS工艺
- LQFP-100，14x14mm无铅封装

方框图



PLL(Phase-Locked Loop) 锁相环

将25MHz的时钟源经过6倍频，建立150MHz的时钟信号。150MHz的时钟用于内部单元的运行，如TCP/IP内核、主机接口管理和寄存器管理。锁相环在复位后锁定并提供稳定的时钟信号。

电源调节系统

电源调节系统通过3.3v的输入建立1.8v/150mA的输出电压。电源调节系统为W5300的内核提供电源。因此不需要其它电源调节器。为了使1.8v的电源更稳定，建议增加电容滤波。

主机接口管理

它根据数据总线的宽度或主机接口模式，检测主机总线信号，管理读写操作。

寄存器管理

它管理模式寄存器、通用（COMMON）寄存器和SOCKET（端口）寄存器。

存储器管理

它管理内部128K字节的数据存储器。由主机分配每个端口的TX/RX存储器。主机可以通过每个SOCKET的FIFO寄存器访问TX/RX存储器。

128K字节TX/RX DPRAM

这是128K字节通信数据存储器，组成16个8K字节的DPRAM (双端口RAM)。可以由主机灵活分配给每个SOCKET。

MII(Media Independent Interface)接口管理

它管理MII接口，根据TEST_MODE[3:0]的配置，MII接口可以在内部PHY和外部PHY（第三方PHY）之间切换。

内部以太网PHY

W5300内部集成了10BaseT/100BaseTX的以太网PHY。PHY支持半双工/全双工自动握手和MDI/MDIX自动检查。它还支持6种网络指示的LED输出，如LINK状态、速度和双工状态。

TCP/IP内核

TCP/IP内核是完全基于WIZnet网络协议处理技术进行硬件逻辑化。

- 802.3以太网MAC（介质访问控制）

它控制以太网的CSMA/CD（载波监听多路访问/冲突检测）访问。他是基于48位源/目的MAC地址的协议技术。它也允许主机通过SOCKET0控制MAC层。因此可以实现软件TCP/IP协议和硬件TCP/IP协议。

- PPPOE（通过以太网的点对点协议）

这是在以太网上实现PPP服务的协议。它将以太网数据帧的有效载荷数据封装为PPP数据帧而进行传输。当接收数据时，它拆封PPP数据帧。PPPoE支持与PPPoE服务器的PPP通信，支持PAP/CHAP验证方法。

- ARP（地址解析协议）

ARP是通过IP地址解析MAC地址的协议。它发送ARP响应给来自对端的ARP请求。它也发送ARP请求查找对端的MAC地址，同时处理对该请求的ARP响应。

- IP（网络协议）

IP协议支持IP层的数据通信。不支持IP分片。不能接收分片的数据包。除了TCP和UDP，所有的协议号都支持。在TCP和UDP情况下，使用硬件的协议栈。

- ICMP（Internet控制信息协议）

它接收ICMP数据包，如分片的MTU、无法访问的目标及标识主机等。当收到Ping请求的ICMP数据包时，它将响应Ping应答的ICMP数据包。它支持最大119个字节的Ping请求。如果超过119个字节时，它将不再支持。

- IGMPv1/v2（Internet组管理协议 版本1/2）

它处理IGMP协议，如加入/脱离组、在UDP多播模式下报告等等。只支持IGMP的版本1和版本2。如果使用更高版本的IGMP， 则需要在IP层手动实现。

- UDP（数据报文协议）

这是在UDP层实现数据传输的协议。它支持用户报文，如单播、多播和广播。

- TCP（传输控制协议）

这是在TCP层实现数据传输的协议。它支持“ TCP客户端 ”和“ TCP服务器 ”。

W5300不需要主机的干预，内部处理所有的通信协议。W5300基于TOE（TCP/IP Offload引擎），通过减少主机处理TCP/IP协议时的负荷，可以极大地提升主机的性能。

1. 引脚描述

类型	描述	类型	描述
I	输入	D	内部75K电阻下拉
O	输出	M	多功能
IO	输入/输出（双向）	H	高电平有效
U	内部75K电阻上拉	L	低电平有效

注意：IUL：内部75K电阻上拉输入引脚，低电平有效
OM：多功能输出

1.1 引脚排列

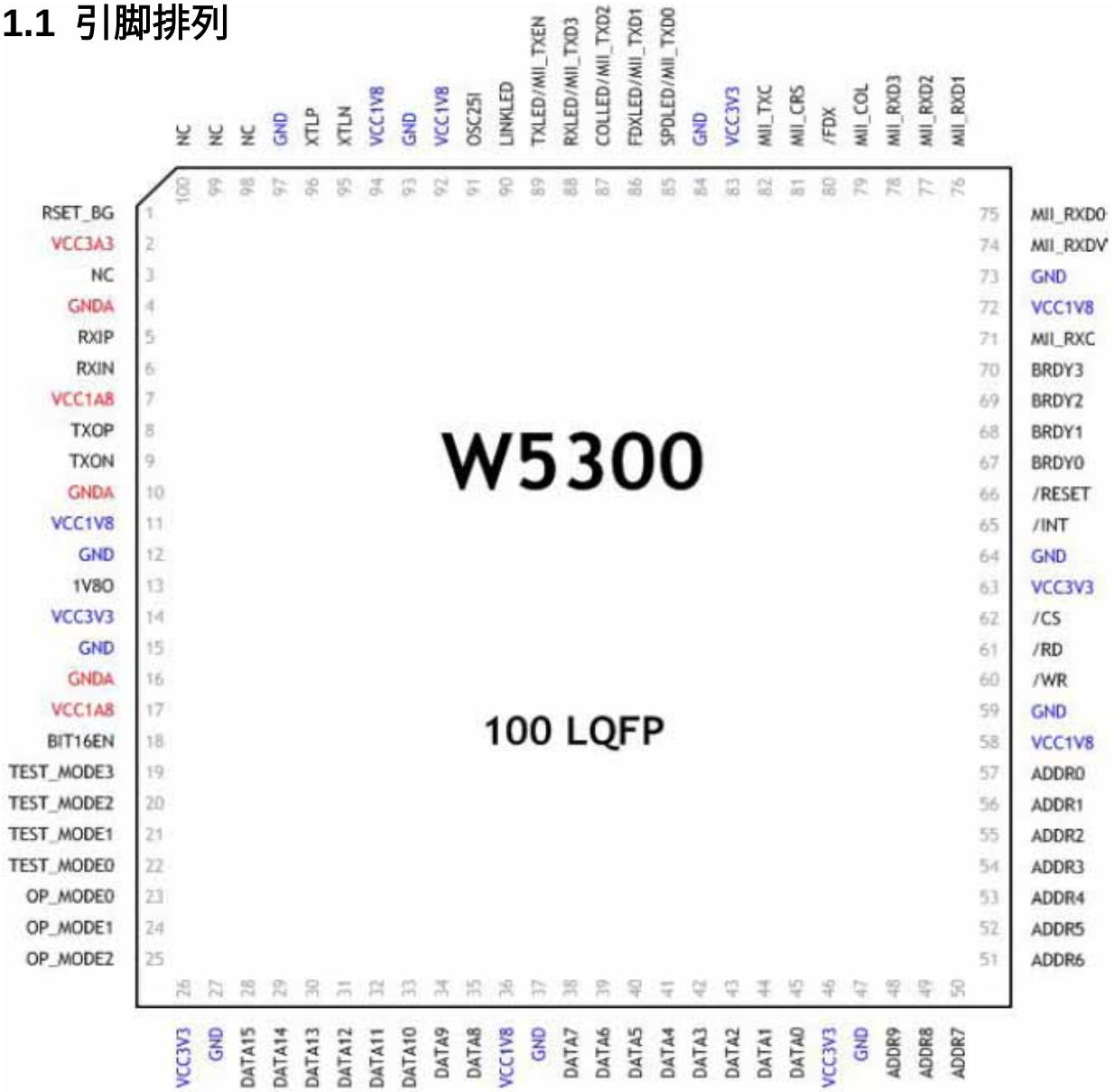


图1 引脚排列

1.2 配置信号

<table>
<tr><th>符号</th><th>类型</th><th>描述</th></tr>
<tr><td>TEST_MODE[3:0]</td><td>ID</td><td>W5300模式选择
它配置W5300的PHY模式和厂家测试模式
TEST_MODE
<table>
<tr><th>3</th><th>2</th><th>1</th><th>0</th><th>描述</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>内部PHY模式（正常运行模式）</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>外部PHY模式，晶体时钟信号</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>外部PHY模式，振荡器时钟信号</td></tr>
<tr><td colspan="4">其它</td><td>保留</td></tr>
</table>
在外部PHY模式，时钟输入引脚根据时钟信号源改变</td></tr>
<tr><td>OP_MODE[2:0]</td><td>ID</td><td>内部物理（PHY）层工作模式控制。配置内部PHY的运行模式
OP_MODE
<table>
<tr><th>2</th><th>1</th><th>0</th><th>描述</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>正常运行模式，推荐使用
全功能自动握手</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>100BASE-TX FDX/HDX自动握手</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>10BASE-T FDX/HDX自动握手</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>保留</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>手动选择100BASE-TX FDX</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>手动选择100BASE-TX HDX</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>手动选择10BASE-T FDX</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>手动选择10BASE-T HDX</td></tr>
</table>
FDX：全双工，HDX：半双工
硬件复位后该设置值被锁定</td></tr>
</table>

1.3 主机接口信号

符号	类型	描述
/RESET	IL	硬件复位信号输入，低电平有效 它对W5300进行初始化。RESET信号低电平至少持续2us，为了使锁相环逻辑稳定，复位信号恢复高电平后至少等待10ms 参考“ 7 电气特性 ”中RESET时序 W5300不支持上电复位。因此必须由外部系统给出复位信号
BIT16EN	IU	16/8位数据位选择，它确定W5300的数据位的宽度 ● 高电平选择16位数据位 ● 低电平选择8位数据位 在复位期间，它被锁存在模式寄存器（MR）的第15位，复位后它的改变不会产生影响。即数据位的宽度在复位后不会发生改变。如果使用8位数据宽度，该引脚须接地
ADDR9-0	ID	地址位 这些信号由W5300的主机接口模式和数据宽度来选择。当使用16位数据宽度时，ADDR0内部不起作用。参考“ 6 外部接口 ”
DATA[15:8]	IO	数据高8位 用于读/写W5300寄存器的操作。在8位数据总线时，这些引脚呈高阻状态
DATA[7:0]	IO	数据低8位 用于读/写W5300寄存器的操作
/CS	IL	片选信号 在主机读写操作时使W5300选通有效 当/CS为高电平时，DATA[15：0]呈高阻状态
/WR	IL	写使能信号 主机通过DATA[15：0]向W5300的ADDR[9：0]寄存器写入数据 根据存取数据的时序状态，DATA[15：0]锁存在W5300的寄存器中 参考MR的13-11位（WDF[2：0]）
/RD	IL	读使能信号 主机通过DATA[15：0]读W5300的ADDR[9：0]寄存器的数据
/INT	OL	中断请求输出 当产生中断请求时（TCP连接/断开、数据接收/发送、超时等）输出低电平。当主机完成中断服务处理，并清除中断寄存器（IR）后，输出恢复高电平 参考：中断寄存器（IR），中断屏蔽寄存器（IMR），端口中断寄存器（Sn_IR），端口中断屏蔽寄存器（Sn_IMR）
BRDY[3:0]	O	缓冲区准备就绪状态指示 这些引脚由用户根据SOCKET号、存储器类型及缓冲区的深度进行配置，当对应SOCKET的TX的剩余字节长度或RX接收数据的字节长度等于或大于配置的缓冲区的深度，那么相应的引脚输出高电平或低电平。参考“ 4.3 通用寄存器 ”的Pn_BRDYR和Pn_DPTHR寄存器

1.4 介质接口信号

介质接口（10Mbps/100Mbps）是内部PHY模式使用（TEST_MODE[3:0]=0000），参考“ 1.2 配置信号 ”）。

符号	类型	说明
RXIP	I	RXIP/RXIN信号对 差分接收输入信号对。从介质接收数据。为了更好地阻抗匹配，这对信号需要2个50Ω（±1%）终端电阻和一个0.1uF的电容，且电阻和电容靠近变压器。如果不使用，直接接地
RXIN	I	
TXOP	O	TXOP/TXON信号对 差分输出信号对。传输数据到介质。为了更好地阻抗匹配，这对信号需要2个50Ω（±1%）终端电阻和一个0.1uF的电容，且电阻和电容靠近W5300。如果不使用，这对引脚悬空
TXON	O	
RSET_BG	O	外置电阻 该引脚需要经过12.3K（±1%）的电阻接地

为了提高系统性能：

1. 使RXIP/RXIN信号对（RX）的引线长度尽量一致；
2. 使TXOP/TXON信号对（TX）的引线长度尽量一致；
3. 使RXIP/RXIN引线尽量靠近；
4. 使TXOP/TXON引线尽量靠近；
5. RX 和TX信号线尽量远离噪声信号，如偏置电阻或晶体振荡器。更多的信息请参考“ W5100 Layer Guide.pdf ”文件。

1.5 外部PHY的MII接口

如果不使用W5300的内部PHY，可以使用MII接口信号与外部PHY接口。在外部PHY模式（TEST_MODE[3:0] = 0001或0010）时使用这些信号。参考“ 1.2 配置信号 ”。

在内部PHY模式，这些引脚悬空。因为除了多功能引脚以外，这些引脚已经内部下拉接地。

符号	类型	说明
TXLED/MII_TXEN	OMH	输出ACT_LED/传输允许 该引脚指示MII_TXD[3:0]引脚输出了传输的数据包信号。如果MII_TXD[3:0]出现传输数据包的前半字节，该引脚为高电平，而后字节位锁定在MII_TXD[3:0]时，输出为低电平
RXLED/MII_TXD3	OM	RX,COL,FDX,SPD的LED指示输出/传输数据输出 传输数据包与MII_TXC时钟同步，以半字节方式输出到外部PHY MII_TXD3是最高位（MSB）
COLLED/MII_TXD2		
FDXLED/MII_TXD1		
SPDLED/MII_TXD0		
MII_TXC	ID	传输时钟输入 来自外部PHY的连续的传输时钟。在100BaseTX时为25MHz，在10BaseT时为2.5MHz。传输时钟用于MII_TXD[3:0]的时钟参考，并用于网络运行时钟。 上升沿有效
MII_CRS	IDH	载波感应 它用于指示与介质的连接通信。如果没有检测到介质载波，它输出高电平
MII_COL	IDH	冲突检测 当在介质中检测到冲突时，它输出高电平。该信号在半双工时有效而在全双工时忽略该信号。该信号为异步信号
MII_RXD3	ID	接收数据输入 当MII_RXDV为高电平时，接收数据包与MII_RXC同步，并输入半字节数据 MII_RXD3为MSB
MII_RXD2		
MII_RXD1		
MII_RXD0		
MII_RXDV	ID	接收有效数据指示 该信号指示在MII_RXD[3:0]有接收的数据。当在MII_RXD[3:0]接收数据的前半字节有效时，输出高电平，后半字节锁定在MII_RXD[3:0]时，输出低电平 在MII_RXC处于上升沿时有效
MII_RXC	ID	接收时钟输入 来自外部PHY的连续的输入时钟。在100BaseTX时为25MHz，在10BaseT时为2.5MHz。传输时钟用于MII_RXD[3:0]和MII_RXDV的时钟参考 上升沿有效
/FDX	IDL	全双工选择 0：全双工，1：半双工 来自外部PHY的输入信号，用于指示外部PHY的连接状态。绝大多数PHY支持自动握手，并通过LED或其它信号指示其状态。可以连接到这些信号，并可以通过接高电平或低电平进行手动配置

为了提高系统性能，建议：

1. MII接口信号尽量不超过25cm；
2. MII_TXD[3:0]信号线长度尽量相等；
3. MII_RXD[3:0]信号线的长度尽量相等；
4. MII_TXC的信号线长度不要超过MII_TXD[3:0]信号线2.5cm；
5. MII_RXC的信号线长度不要超过MII_RXD[3:0]信号线2.5cm。

1.6 网络LED指示信号

根据TEST_MODE[3:0]配置，除了LINKLED以外，其余引脚都是与其它信号复用。当这些信号作为网络状态指示时，应该配置为内部PHY（TEST_MODE[3:0]=0000）。

符号	类型	说明
LINKLED	OL	以太网连接指示 它指示介质连接（10/100M）状态
TXLED/MII_TXEN	OML	传输数据指示/传输允许 它指示有传输的数据通过TXOP/TXON端输出（传输激活）
RXLED/MII_TXD3	OML	接收数据指示/传输数据 它指示在RXIP/RXIN端有数据输入（接收激活） 将TXLED和RXLED用一个“与”门连接，可用于网络工作LED指示
COLLED/MII_TXD2	OML	冲突指示/传输数据 当冲突发生时它产生指示。只在半双工有效，全双工是无效的
FDXLED/MII_TXD1	OML	全双工指示/发送数据 根据自动握手或OP_MODE[2:0]手动配置所产生的状态，在全双工工作状态时输出低电平，而在半双工状态时输出高电平。
SPDLED/MII_TXD0	OML	连接速度指示/发送数据 根据自动握手或OP_MODE[2:0]手动配置所产生的状态，100Mbps时输出低电平，10Mbps时输出高电平

1.7 Clock Signals 时钟信号

无论是晶体还是振荡器都可以作W5300的时钟信号。25MHz的外部时钟信号通过锁相环产生150MHz的时钟，150MHz的时钟用于PLL_CLK(周期6.67ns)、以及W5300内核的工作时钟

符号	类型	说明
XTLP	I	25MHz晶体输入/输出 内部振荡器需要外接25MHz并行谐振晶体和匹配电容 参考“7 电参数”中的“时钟特性” 该时钟可用于内部PHY模式（TEST_MODE[3:0]=0000）和外部PHY模式（TEST_MODE[3:0]=0001） 在内部PHY模式使用外部振荡信号时，确定信号幅度为1.8v，并从XTLP输入，且XTLN悬空
XTLN	O	
OSC25I	I	25MHz振荡信号输入 只用于外部PHY模式（TEST_MODE[3:0]=0010）的振荡器信号。为了阻止泄露电流，必须保持XTLP为高电平并将XTLN悬空，振荡信号为1.8v

1.8 电源

符号	类型	说明
VCC3A3	电源	3.3v模拟电源 在VCC3A3与GNDA之间接10uF的钽电容
VCC3V3	电源	3.3v数字电源 在每一个VCC引脚和地之间都接一个0.1uF的去藕电容。VCC3V3与VCC3A3之间通过一个1uH的铁氧体磁珠电感
VCC1A8	电源	1.8v模拟电源 在VCC1A8和GNDA之间接10uF的钽电容和0.1uF的电容，以过滤W5300内核的电源噪声
VCC1V8	电源	1.8v数字电源 在每一个VCC和GND之间接0.1uF的去藕电容
GNDA	电源地	模拟电源地 在设计PCB板时，模拟电源地尽量宽
GND	电源地	数字电源地 在设计PCB板时，数字电源地尽量宽
1V8O	电源输出	1.8v电源输出 内部电源调整器输出电1.8v/150mA，用于W5300内核的工作（VCC1A8，VCC1V8）。在1V8O引脚与GND之间接3.3uF的钽电容用于频率补偿，接0.1uF的电容用于高频噪声去藕。1V8O连接到VCC1V8，经过1uH的铁氧体磁珠电感隔离连接到VCC1A8 注意：1V8O是为W5300的内核工作供电。它不能给其它器件提供电源

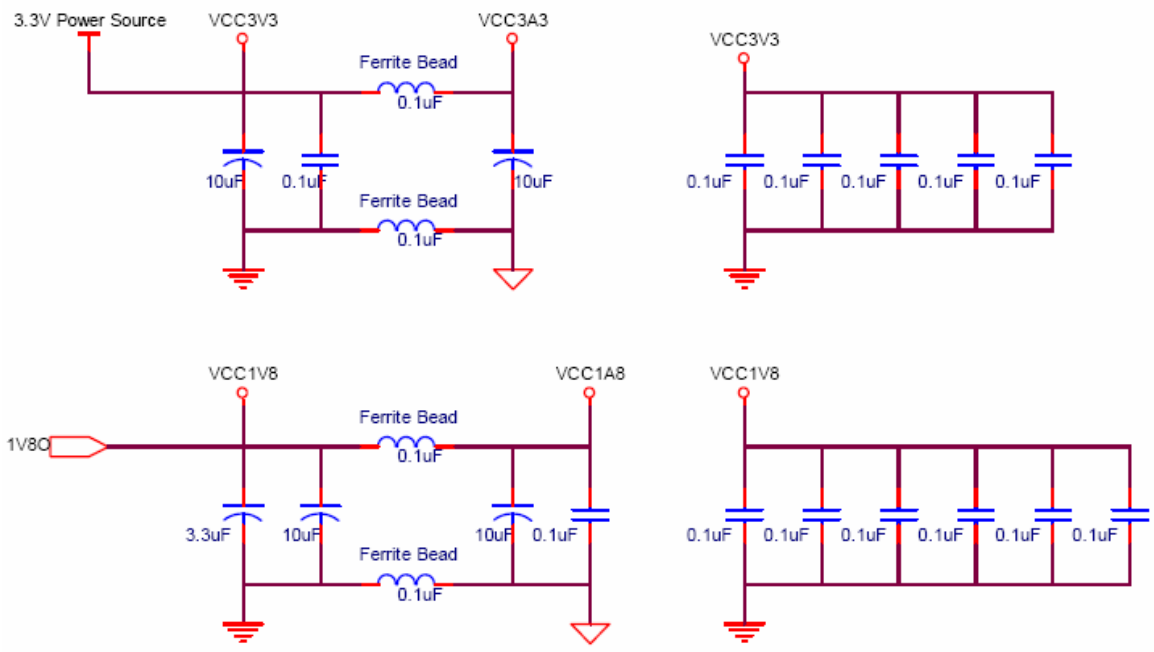


图2 电源设计

电源系统设计建议：

1. 去藕电容应尽量靠近W5300；
2. 电源地线尽可能的宽；
3. 如果地线的空间足够的大，尽量把模拟地和数字地分开；
4. 如果地线的空间不够大，模拟地和数字地设计在一起比分开更好。

2. 系统存储器映射

W5300与主机接口模式支持直接地址模式和间接地址模式两种。

在直接地址模式，主机可以将W5300的寄存器映射到存储空间而直接访问W5300的寄存器。直接地址模式中存储器映射是由模式寄存器（MR）、通用寄存器（COMMON）和端口寄存器（SOCKET）组成的。这些寄存器从主机系统的基地址开始以2字节递增方式连续映射在主机存储器空间。使用映射地址，主机系统可以直接访问MR、COMMON和SOCKET寄存器。使用直接地址模式，只需要0x400字节的存储器空间。

在间接地址模式，主机通过IDM_AR寄存器（间接模式地址寄存器）和IDM_DR寄存器（间接模式数据寄存器）间接访问W5300的COMMON寄存器和SOCKET寄存器。IDM_AR、IDM_DR和MR直接映射在主机地址空间。间接地址模式寄存器映射是由可直接访问的MR、IDM_AR和IDM_DR寄存器和可间接访问的COMMON、SOCKET寄存器构成。只有MR、IDM_AR和IDM_DR寄存器从主机基地址开始以2字节递增方式连续映射在主机的地址空间。而COMMON和SOCKET寄存器不映射在主机地址空间，这些寄存器是通过IDM_AR和IDM_DR间接访问。

使用间接地址模式，只需要0x06个字节的地址空间，当主机在间接地址模式下访问COMMON寄存器的中断寄存器（IR）时，操作如下：

主机写入：设置IDM_AR为0x0002，IR的地址(IDM_AR=0x0002)

设置IDM_DR为0xFFFF (IDM_DR=0xFFFF)

主机读：设置IDM_AR为0x0002，IR的地址（IDM_AR=0x0002）

读IDM_DR并保存读取的值

W5300的主机接口模式由MR寄存器的‘ IND ’位（第0位）确定。

MR(0) = 0 => 直接地址模式

MR(0) = 1 => 间接地址模式

每种地址模式下存储映射如下：

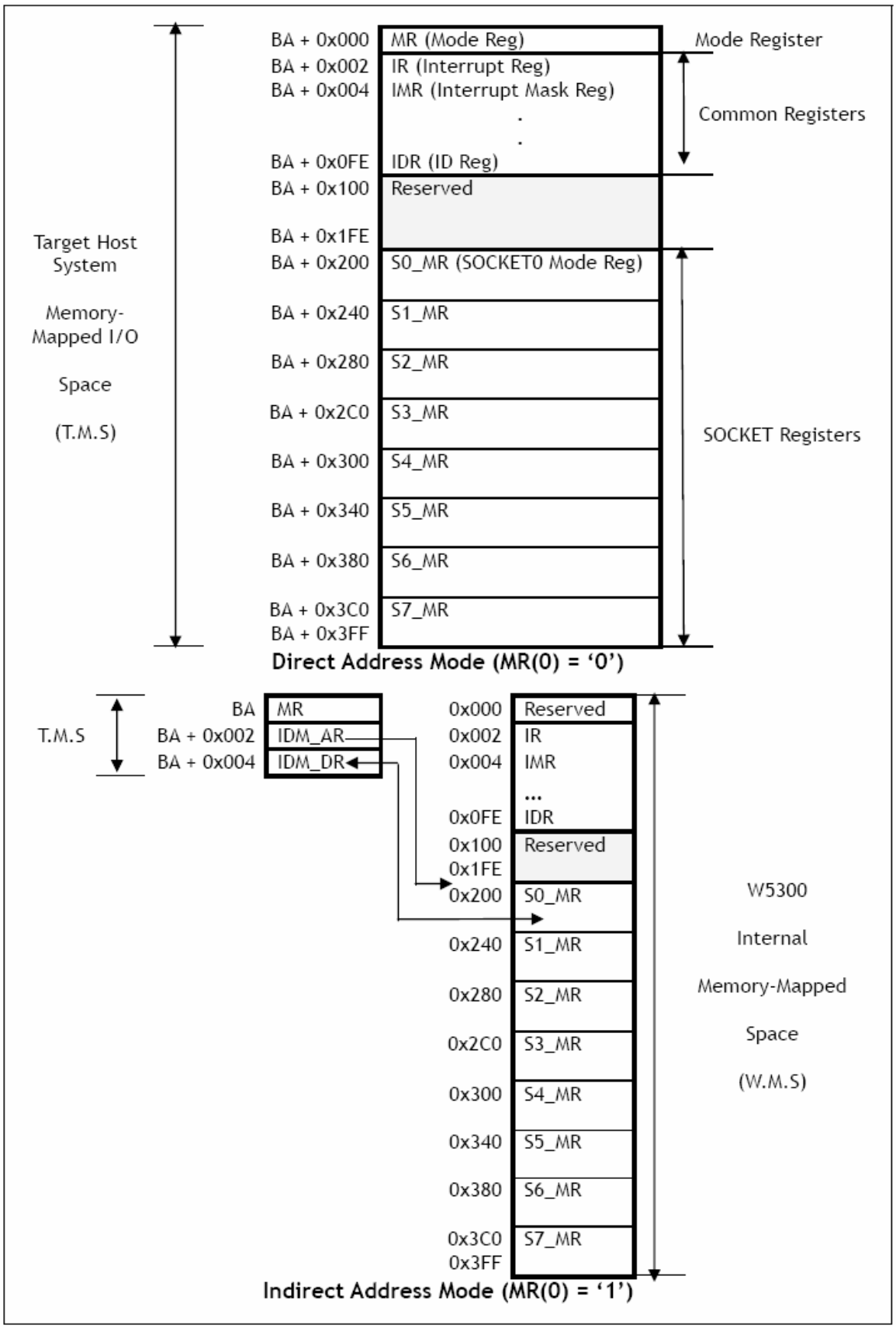


图3 存储器映射

3. W5300寄存器

W5300的寄存器由MR寄存器（直接地址模式和间接地址模式）、IDM_AR和IDM_DR寄存器（只用于间接地址模式）、COMMON寄存器和SOCKET寄存器组成。

MR、IDM_AR和IDM_DR映射在主机地址空间，COMMON寄存器和SOCKET寄存器则根据地址模式映射在主机地址空间或W5300内部地址空间。

W5300所有寄存器有1字节、2字节、4字节或6字节。根据主机系统数据总线宽度，主机在访问W5300时，在16位数据宽度时为2字节偏移，在8位数据数据宽度时为1字节偏移。

当W5300的寄存器映射在主机存储空间时，W5300的实际物理地址计算如下：

W5300 寄存器的实际物理地址 = 存储器空间的基地址 + W5300 寄存器偏移量

W5300寄存器顺序采用大端模式，即低地址的寄存器存储在高字节。

寄存器符号

MR：MR（模式）寄存器
MR0：MR 的低地址寄存器（地址偏移量为 0），高字节
MR1：MR 的高地址寄存器（地址偏移量为 1），低字节
MR(15:5)：11 位（MR 寄存器的第 15 位到第 5 位）
MR(0)：MR 寄存器的第 0 位，MR1 的第 0 位
MR(13)：MR 寄存器的第 13 位，MR0 的第 5 位
MR0(7)：MR 寄存器的第 15 位，MR0 寄存器的最高位
MR(DWB)：MR 寄存器的 DWB 位

SHAR：本机硬件地址寄存器
SHAR0：本机硬件地址的第一个字节寄存器（地址偏移量：0x008）
SHAR1：本机硬件地址的第二个字节寄存器（地址偏移量：0x009）
SHAR2：本机硬件地址的第三个字节寄存器（地址偏移量：0x00A）
SHAR3：本机硬件地址的第四个字节寄存器（地址偏移量：0x00B）
SHAR4：本机硬件地址的第五个字节寄存器（地址偏移量：0x00C）
SHAR5：本机硬件地址的第六个字节寄存器（地址偏移量：0x00D）

3.1 模式寄存器

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x000	0x000	MR	MR0	模式寄存器
	0x001		MR1	

3.2 间接模式寄存器

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x002	0x002	IDM_AR	IDM_AR0	间接模式地址寄存器
	0x003		IDM_AR1	
0x004	0x004	IDM_DR	IDM_DR0	间接模式数据寄存器
	0x005		IDM_DR1	

3.3 通用（COMMON）寄存器

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x002	0x002	IR	IR0	中断寄存器
	0x003		IR1	
0x004	0x004	IMR	IMR0	中断屏蔽寄存器
	0x005		IMR1	
0x006	0x006	-		保留
	0x007			
0x008	0x008	SHAR	SHAR0	本机硬件地址寄存器
	0x009		SHAR1	
0x00a	0x00A	SHAR2	SHAR2	
	0x00B		SHAR3	
0x00c	0x00C	SHAR4	SHAR4	
	0x00D		SHAR5	
0x00e	0x00E	-		保留
	0x00F			
0x010	0x010	GAR	GAR0	网关地址寄存器
	0x011		GAR1	
0x012	0x012	GAR2	GAR2	
	0x013		GAR3	
0x014	0x014	SUBR	SUBR0	子网掩码寄存器
	0x015		SUBR1	
0x016	0x016	SUBR2	SUBR2	
	0x017		SUBR3	
0x018	0x018	SIPR	SIPR0	本机IP地址寄存器
	0x019		SIPR1	
0x01a	0x01A	SIPR2	SIPR2	
	0x01B		SIPR3	
0x01c	0x01C	RTR	RTR0	超时设置值寄存器，超时启动重新传输
	0x01D		RTR1	
0x01e	0x01E	RCR	RCR0	保留
	0x01F		RCR1	重新传输计数寄存器
0x020	0x020	TMS01R	TMSR0	端口0传输存储器大小配置寄存器
	0x021		TMSR1	端口1传输存储器大小配置寄存器
0x022	0x022	TMS23R	TMSR2	端口2传输存储器大小配置寄存器
	0x023		TMSR3	端口3传输存储器大小配置寄存器
0x024	0x024	TMS45R	TMSR4	端口4传输存储器大小配置寄存器
	0x025		TMSR5	端口5传输存储器大小配置寄存器
0x026	0x026	TMS67R	TMSR6	端口6传输存储器大小配置寄存器
	0x027		TMSR7	端口7传输存储器大小配置寄存器

0x028	0x028	RMS01R	RMSR0	端口0接收存储器大小配置寄存器
	0x029		RMSR1	端口1接收存储器大小配置寄存器
0x02A	0x02A	RMS23R	RMSR2	端口2接收存储器大小配置寄存器
	0x02B		RMSR3	端口3接收存储器大小配置寄存器
0x02C	0x02C	RMS45R	RMSR4	端口4接收存储器大小配置寄存器
	0x02D		RMSR5	端口5接收存储器大小配置寄存器
0x02E	0x02E	RMS67R	RMSR6	端口6接收存储器大小配置寄存器
	0x02F		RMSR7	端口7接收存储器大小配置寄存器
0x030	0x030	MTYPER	MTYPER0	存储器单元类型寄存器
	0x031		MTYPER1	
0x032	0x032	PATR	PATR0	PPPoE认证寄存器
	0x033		PATR1	
0x034	0x034	-		保留
	0x035			
0x036	0x036	PTIMER	PTIMER0	PPP LCP请求时间寄存器
	0x037		PTIMER1	
0x038	0x038	PMAGICR	PMAGICR0	PPP LCP魔数寄存器
	0x039		PMAGICR1	
0x03A	0X03A	-		保留
	0X03B			
0x03C	0x03C	PSIDR	PSIDR0	PPP对话ID寄存器
	0x03D		PSIDR1	
0x03E	0x03E	-		保留
	0x03F			
0x040	0x040	PDHAR	PDHAR0	PPP目的硬件地址寄存器
	0x041		PDHAR1	
0x042	0x042	PDHAR2	PDHAR2	
	0x043		PDHAR3	
0x044	0x044	PDHAR4	PDHAR4	
	0x045		PDHAR5	
0x046	0x046	-		保留
	0x047			
0x048	0x048	UIPR	UIPR0	无法访问的IP地址寄存器
	0x049		UIPR1	
0x04A	0x04A	UIPR2	UIPR2	
	0x04B		UIPR3	
0x04C	0x04C	UPORTR	UPORTR0	无法访问的端口号寄存器
	0x04D		UPORTR1	
0x04E	0x04E	FMTUR	FMTUR0	分片最大传输单元寄存器
	0x04F		FMTUR1	
0x050	0x050			保留
…				
0x05E	0x05F			
0x060	0x060	P0_BRDYR	P0_BRDYR0	保留
	0x061		P0_BRDYR1	BRDY0引脚配置寄存器
0x062	0x062	P0_BDPTHR	P0_BDPTHR0	BRDY0引脚缓冲区大小寄存器
	0x063		P0_BDPTHR1	
0x064	0x064	P1_BRDYR	P1_BRDYR0	保留
	0x065		P1_BRDYR1	BRDY1引脚配置寄存器
0x066	0x066	P1_BDPTHR	P1_BDPTHR0	BRDY0引脚缓冲区大小寄存器
	0x067		P1_BDPTHR1	
0x068	0x068	P2_BRDYR	P2_BRDYR0	保留
	0x069		P2_BRDYR1	BRDY2引脚配置寄存器
0x06A	0x06A	P2_BDPTHR	P2_BDPTHR0	BRDY2引脚缓冲区大小寄存器
	0x06B		P2_BDPTHR0	
0x06C	0x06C	P3_BRDYR	P3_BRDYR0	保留
	0x06D		P3_BRDYR1	BRDY3引脚配置寄存器

0x06E	0x06E 0x06F	P3_BDPTHR	P3_BDPTHR0 P3_BDPTHR1	BRDY3引脚缓冲区大小寄存器
0x070	0x070 0x071			保留
…				
0x0FC	0x0FC 0x0FD			
0x0FE	0x0FE 0x0FF	IDR	IDR0 IDR1	W5300的ID寄存器

3.4 SOCKET（端口）寄存器

SOCKET0：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x200	0x200 0x201	S0_MR	S0_MR0 S0_MR1	SOCKET0模式寄存器
0x202	0x202 0x203	S0_CR	S0_CR0 S0_CR1	保留 SOCKET0命令寄存器
0x204	0x204 0x205	S0_IMR	S0_IMR0 S0_IMR1	保留 SOCKET0中断屏蔽寄存器
0x206	0x206 0x207	S0_IR	S0_IR0 S0_IR1	保留 SOCKET0中断寄存器
0x208	0x208 0x209	S0_SSR	S0_SSR0 S0_SSR1	保留 SOCKET0状态寄存器
0x20A	0x20A 0x20B	S0_PORTR	S0_PORT0 S0_PORT1	SOCKET0的端口号
0x20C	0x20C 0x20D	S0_DHAR	S0_DHAR0 S0_DHAR1	SOCKET0目的硬件地址寄存器
0x20E	0x20E 0x20F	S0_DHAR2	S0_DHAR2 S0_DHAR3	
0x210	0x210 0x211	S0_DHAR4	S0_DHAR4 S0_DHAR5	
0x212	0x212 0x213	S0_DPORTR	S0_DPORTR0 S0_DPORTR1	SOCKET0目的端口号地址寄存器
0x214	0x214 0x215	S0_DIPR	S0_DIPR0 S0_DIPR1	SOCKET0目的IP地址寄存器
0x216	0x216 0x217	S0_DIPR2	S0_DIPR2 S0_DIPR3	
0x218	0x218 0x219	S0_MSSR	S0_MSSR0 S0_MSSR1	SOCKET0最大分片字节长度寄存器
0x21A	0x21A 0x21B	S0_PORTOR	S0_KPALVTR S0_PROTOR	SOCKET0生存期时间寄存器 SOCKET0协议号寄存器
0x21C	0x21C 0x21D	S0_TOSR	S0_TOSR0 S0_TOSR1	保留 SOCKET0的TOS寄存器
0x21E	0x21E 0x21F	S0_TTLR	S0_TTLR0 S0_TTLR1	保留 SOCKET0的TTL寄存器
0x220	0x220 0x221	S0_TX_WRSR	S0_TX_WRSR0 S0_TX_WRSR1	保留 SOCKET0的TX写入字节长度寄存器
0x222	0x222 0x223	S0_TX_WRSR2	S0_TX_WRSR2 S0_TX_WRSR3	
0x224	0x224 0x225	S0_TX_FSR	S0_TX_FSR0 S0_TX_FSR1	保留 SOCKET0的TX剩余字节长度寄存器
0x226	0x226 0x227	S0_TX_FSR2	S0_TX_FSR2 S0_TX_FSR3	

0x228	0x228	S0_RX_RSR	S0_RX_RSR0	保留
	0x229		S0_RX_RSR1	SOCKET0的RX接收数据字节长度寄存器
0x22A	0x22A	S0_RX_RSR2	S0_RX_RSR2	
	0x22B		S0_RX_RSR3	
0x22C	0x22C	S0_FRAGR	S0_FRAGR0	保留
	0x22D		S0_FRAGR1	SOCKET0标志寄存器
0x22E	0x22E	S0_TX_FIFOR	S0_TX_FIFOR0	SOCKET0的TX FIFO寄存器
	0x22F		S0_TX_FIFOR1	
0x230	0x230	S0_RX_FIFOR	S0_RX_FIFOR0	SOCKET0的RX FIFO寄存器
	0x231		S0_RX_FIFOR1	
0x232	0x232			保留
	0x233			
…				
0x23E	0x23E			
	0x23			

SOCKET1：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x240	0x240	S1_MR	S1_MR0	SOCKET1的模式寄存器
	0x241		S1_MR1	
0x242	0x242	S1_CR	S1_CR0	保留
	0x243		S1_CR1	SOCKET1命令寄存器
0x244	0x244	S1_IMR	S1_IMR0	保留
	0x245		S1_IMR1	SOCKET1中断屏蔽寄存器
0x246	0x246	S1_IR	S1_IR0	保留
	0x247		S1_IR1	SOCKET1中断寄存器
0x248	0x248	S1_SSR	S1_SSR0	保留
	0x249		S1_SSR1	SOCKET1状态寄存器
0x24A	0x24A	S1_PORT	S1_PORT0	SOCKET1的端口号
	0x24B		S1_PORT1	
0x24C	0x24C	S1_DHAR	S1_DHAR0	SOCKET1目的硬件地址寄存器
	0x24D		S1_DHAR1	
0x24E	0x24E	S1_DHAR2	S1_DHAR2	
	0x24F		S1_DHAR3	
0x250	0x250	S1_DHAR4	S1_DHAR4	
	0x251		S1_DHAR5	
0x252	0x252	S1_DPORTR	S1_DPORTR0	SOCKET1的目的端口号寄存器
	0x253		S1_DPORTR1	
0x254	0x254	S1_DIPR	S1_DIPR0	SOCKET1的目的IP地址寄存器
	0x255		S1_DIPR1	
0x256	0x256	S1_DIPR2	S1_DIPR2	
	0x257		S1_DIPR3	
0x258	0x258	S1_MSSR	S1_MSSR0	SOCKET1最大分片字节长度寄存器
	0x259		S1_MSSR1	
0x25A	0x25A	S1_PORTOR	S1_KPALVTR	SOCKET1生存期时间寄存器
	0x25B		S1_PROTOR	SOCKET1协议号寄存器
0x25C	0x25C	S1_TOSR	S1_TOSR0	保留
	0x25D		S1_TOSR1	SOCKET1的TOS寄存器
0x25E	0x25E	S1_TTLR	S1_TTLR0	保留
	0x25F		S1_TTLR1	SOCKET1的TTL寄存器
0x260	0x260	S1_TX_WRSR	S1_TX_WRSR0	保留
	0x261		S1_TX_WRSR1	SOCKET1的TX写入字节长度寄存器

0x262	0x262	S1_TX_WRSR2	S1_TX_WRSR2	
	0x263		S1_TX_WRSR3	
0x264	0x264	S1_TX_FSR	S1_TX_FSR0	保留
	0x265		S1_TX_FSR1	SOCKET1的TX剩余字节长度寄存器
0x266	0x266	S1_TX_FSR2	S1_TX_FSR2	
	0x267		S1_TX_FSR3	
0x268	0x268	S1_RX_RSR	S1_RX_RSR0	保留
	0x269		S1_RX_RSR1	SOCKET1的RX接收数据字节长度寄存器
0x26A	0x26A	S1_RX_RSR2	S1_RX_RSR2	
	0x26B		S1_RX_RSR3	
0x26C	0x26C	S1_FRAGR	S1_FRAGR0	保留
	0x26D		S1_FRAGR1	SOCKET1的IP标志段寄存器
0x26E	0x26E	S1_TX_FIFOR	S1_TX_FIFOR0	SOCKET1的TX FIFO寄存器
	0x26F		S1_TX_FIFOR1	
0x270	0x270	S1_RX_FIFOR	S1_RX_FIFOR0	SOCKET1的RX FIFO寄存器
	0x271		S1_RX_FIFOR1	
0x272	0x272			保留
	0x273			
…				
0x27E	0x27E			
	0x27F			

SOCKET2：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x280	0x280	S2_MR	S2_MR0	SOCKET2的模式寄存器
	0x281		S2_MR1	
0x282	0x282	S2_CR	S2_CR0	保留
	0x283		S2_CR1	SOCKET2命令寄存器
0x284	0x284	S2_IMR	S2_IMR0	保留
	0x285		S2_IMR1	SOCKET2中断屏蔽寄存器
0x286	0x286	S2_IR	S2_IR0	保留
	0x287		S2_IR1	SOCKET2中断寄存器
0x288	0x288	S2_SSR	S2_SSR0	保留
	0x289		S2_SSR1	SOCKET2状态寄存器
0x28A	0x28A	S2_PORT	S2_PORT0	SOCKET2的端口号
	0x28B		S2_PORT1	
0x28C	0x28C	S2_DHAR	S2_DHAR0	SOCKET2目的硬件地址寄存器
	0x28D		S2_DHAR1	
0x28E	0x28E	S2_DHAR2	S2_DHAR2	
	0x28F		S2_DHAR3	
0x290	0x290	S2_DHAR4	S2_DHAR4	
	0x291		S2_DHAR5	
0x292	0x292	S2_DPORTR	S2_DPORTR0	SOCKET2的目的端口号寄存器
	0x293		S2_DPORTR1	
0x294	0x294	S2_DIPR	S2_DIPR0	SOCKET2的目的IP地址寄存器
	0x295		S2_DIPR1	
0x296	0x296	S2_DIPR2	S2_DIPR2	
	0x297		S2_DIPR3	
0x298	0x298	S2_MSSR	S2_MSSR0	SOCKET2最大分片字节长度寄存器
	0x299		S2_MSSR1	
0x29A	0x29A	S2_PORTOR	S2_KPALVTR	SOCKET2生存期时间寄存器
	0x29B		S2_PROTOR	SOCKET2协议号寄存器
0x29C	0x29C	S2_TOSR	S2_TOSR0	保留
	0x29D		S2_TOSR1	SOCKET2的TOS寄存器

0x29E	0x29E	S2_TTLR	S2_TTLR0	保留
	0x29F		S2_TTLR1	SOCKET2的TTL寄存器
0x2A0	0x2A0	S2_TX_WRSR	S2_TX_WRSR0	保留
	0x2A1		S2_TX_WRSR1	SOCKET2的TX写入字节长度寄存器
0x2A2	0x2A2	S2_TX_WRSR2	S2_TX_WRSR2	
	0x2A3		S2_TX_WRSR3	
0x2A4	0x2A4	S2_TX_FSR	S2_TX_FSR0	保留
	0x2A5		S2_TX_FSR1	SOCKET2的TX剩余字节长度寄存器
0x2A6	0x2A6	S2_TX_FSR2	S2_TX_FSR2	
	0x2A7		S2_TX_FSR3	
0x2A8	0x2A8	S2_RX_RSR	S2_RX_RSR0	保留
	0x2A9		S2_RX_RSR1	SOCKET2的RX接收数据字节长度寄存器
0x2AA	0x2AA	S2_RX_RSR2	S2_RX_RSR2	
	0x2AB		S2_RX_RSR3	
0x2AC	0x2AC	S2_FRAGR	S2_FRAGR0	保留
	0x2AD		S2_FRAGR1	SOCKET2的IP标志段寄存器
0x2AE	0x2AE	S2_TX_FIFOR	S2_TX_FIFOR0	SOCKET2的TX FIFO寄存器
	0x2AF		S2_TX_FIFOR1	
0x2B0	0x2B0	S2_RX_FIFOR	S2_RX_FIFOR0	SOCKET2的RX FIFO寄存器
	0x2B1		S2_RX_FIFOR1	
0x2B2	0x2B2			保留
	0x2B3			
…				
0x2BE	0x2BE			
	0x2BF			

SOCKET3：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x2C0	0x2C0	S3_MR	S3_MR0	SOCKET3的模式寄存器
	0x2C1		S3_MR1	
0x2C2	0x2C2	S3_CR	S3_CR0	保留
	0x2C3		S3_CR1	SOCKET3命令寄存器
0x2C4	0x2C4	S3_IMR	S3_IMR0	保留
	0x2C5		S3_IMR1	SOCKET3中断屏蔽寄存器
0x2C6	0x2C6	S3_IR	S3_IR0	保留
	0x2C7		S3_IR1	SOCKET3中断寄存器
0x2C8	0x2C8	S3_SSR	S3_SSR0	保留
	0x2C9		S3_SSR1	SOCKET3状态寄存器
0x2CA	0x2CA	S3_PORT	S3_PORT0	SOCKET3的端口号
	0x2CB		S3_PORT1	
0x2CC	0x2CC	S3_DHAR	S3_DHAR0	SOCKET3目的硬件地址寄存器
	0x2CD		S3_DHAR1	
0x2CE	0x2CE	S3_DHAR2	S3_DHAR2	
	0x2CF		S3_DHAR3	
0x2D0	0x2D0	S3_DHAR4	S3_DHAR4	
	0x2D1		S3_DHAR5	
0x2D2	0x2D2	S3_DPORTR	S3_DPORTR0	SOCKET3的目的端口号寄存器
	0x2D3		S3_DPORTR1	
0x2D4	0x2D4	S3_DIPR	S3_DIPR0	SOCKET3的目的IP地址寄存器
	0x2D5		S3_DIPR1	
0x2D6	0x2D6	S3_DIPR2	S3_DIPR2	
	0x2D7		S3_DIPR3	

0x2D8	0x2D8	S3_MSSR	S3_MSSR0	SOCKET3最大分片字节长度寄存器
	0x2D9		S3_MSSR1	
0x2DA	0x2DA	S3_PORTOR	S3_KPALVTR	SOCKET3生存期时间寄存器
	0x2DB		S3_PROTOR	SOCKET3协议号寄存器
0x2DC	0x2DC	S3_TOSR	S3_TOSR0	保留
	0x2DD		S3_TOSR1	SOCKET3的TOS寄存器
0x2DE	0x2DE	S3_TTLR	S3_TTLR0	保留
	0x2DF		S3_TTLR1	SOCKET3的TTL寄存器
0x2E0	0x2E0	S3_TX_WRSR	S3_TX_WRSR0	保留
	0x2E1		S3_TX_WRSR1	SOCKET3的TX写入字节长度寄存器
0x2E2	0x2E2	S3_TX_WRSR2	S3_TX_WRSR2	
	0x2E3		S3_TX_WRSR3	
0x2E4	0x2E4	S3_TX_FSR	S3_TX_FSR0	保留
	0x2E5		S3_TX_FSR1	SOCKET3的TX剩余字节长度寄存器
0x2E6	0x2E6	S3_TX_FSR2	S3_TX_FSR2	
	0x2E7		S3_TX_FSR3	
0x2E8	0x2E8	S3_RX_RSR	S3_RX_RSR0	保留
	0x2E9		S3_RX_RSR1	SOCKET3的RX接收数据字节长度寄存器
0x2EA	0x2EA	S3_RX_RSR2	S3_RX_RSR2	
	0x2EB		S3_RX_RSR3	
0x2EC	0x2EC	S3_FRAGR	S3_FRAGR0	保留
	0x2ED		S3_FRAGR1	SOCKET3的IP标志段寄存器
0x2EE	0x2EE	S3_TX_FIFOR	S3_TX_FIFOR0	SOCKET3的TX FIFO寄存器
	0x2EF		S3_TX_FIFOR1	
0x2F0	0x2F0	S3_RX_FIFOR	S3_RX_FIFOR0	SOCKET3的RX FIFO寄存器
	0x2F1		S3_RX_FIFOR1	
0x2F2	0x2F2			保留
	0x2F3			
…				
0x2FE	0x2FE			
	0x2FF			

SOCKET4：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x300	0x300	S4_MR	S4_MR0	SOCKET4的模式寄存器
	0x301		S4_MR1	
0x302	0x302	S4_CR	S4_CR0	保留
	0x303		S4_CR1	SOCKET4命令寄存器
0x304	0x304	S4_IMR	S4_IMR0	保留
	0x305		S4_IMR1	SOCKET4中断屏蔽寄存器
0x306	0x306	S4_IR	S4_IR0	保留
	0x307		S4_IR1	SOCKET4中断寄存器
0x308	0x308	S4_SSR	S4_SSR0	保留
	0x309		S4_SSR1	SOCKET4状态寄存器
0x30A	0x30A	S4_PORT	S4_PORT0	SOCKET4的端口号
	0x30B		S4_PORT1	
0x30C	0x30C	S4_DHAR	S4_DHAR0	SOCKET4目的硬件地址寄存器
	0x30D		S4_DHAR1	
0x30E	0x30E	S4_DHAR2	S4_DHAR2	
	0x30F		S4_DHAR3	
0x310	0x310	S4_DHAR4	S4_DHAR4	
	0x311		S4_DHAR5	

0x312	0x312	S4_DPORTR	S4_DPORTR0	SOCKET4的目的端口号寄存器
	0x313		S4_DPORTR1	
0x314	0x314	S4_DIPR	S4_DIPR0	SOCKET4的目的IP地址寄存器
	0x315		S4_DIPR1	
0x316	0x316	S4_DIPR2	S4_DIPR2	
	0x317		S4_DIPR3	
0x318	0x318	S4_MSSR	S4_MSSR0	SOCKET4最大分片字节长度寄存器
	0x319		S4_MSSR1	
0x31A	0x31A	S4_PORTOR	S4_KPALVTR	SOCKET4生存期时间寄存器
	0x31B		S4_PROTOR	SOCKET4协议号寄存器
0x31C	0x31C	S4_TOSR	S4_TOSR0	保留
	0x31D		S4_TOSR1	SOCKET4的TOS寄存器
0x31E	0x31E	S4_TTLR	S4_TTLR0	保留
	0x31F		S4_TTLR1	SOCKET4的TTL寄存器
0x320	0x320	S4_TX_WRSR	S4_TX_WRSR0	保留
	0x321		S4_TX_WRSR1	SOCKET4的TX写入字节长度寄存器
0x322	0x322	S4_TX_WRSR2	S4_TX_WRSR2	
	0x323		S4_TX_WRSR3	
0x324	0x324	S4_TX_FSR	S4_TX_FSR0	保留
	0x325		S4_TX_FSR1	SOCKET4的TX剩余字节长度寄存器
0x326	0x326	S4_TX_FSR2	S4_TX_FSR2	
	0x327		S4_TX_FSR3	
0x328	0x328	S4_RX_RSR	S4_RX_RSR0	保留
	0x329		S4_RX_RSR1	SOCKET4的RX接收数据字节长度寄存器
0x32A	0x32A	S4_RX_RSR2	S4_RX_RSR2	
	0x32B		S4_RX_RSR3	
0x32C	0x32C	S4_FRAGR	S4_FRAGR0	保留
	0x32D		S4_FRAGR1	SOCKET4的IP标志段寄存器
0x32E	0x32E	S4_TX_FIFOR	S4_TX_FIFOR0	SOCKET4的TX FIFO寄存器
	0x32F		S4_TX_FIFOR1	
0x330	0x330	S4_RX_FIFOR	S4_RX_FIFOR0	SOCKET4的RX FIFO寄存器
	0x331		S4_RX_FIFOR1	
0x332	0x332			保留
	0x333			
…				
0x3FE	0x33E			
	0x33F			

SOCKET5：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x340	0x340	S5_MR	S5_MR0	SOCKET5的模式寄存器
	0x341		S5_MR1	
0x342	0x342	S5_CR	S5_CR0	保留
	0x343		S5_CR1	SOCKET5命令寄存器
0x344	0x344	S5_IMR	S5_IMR0	保留
	0x345		S5_IMR1	SOCKET5中断屏蔽寄存器
0x346	0x346	S5_IR	S5_IR0	保留
	0x347		S5_IR1	SOCKET5中断寄存器
0x348	0x348	S5_SSR	S5_SSR0	保留
	0x349		S5_SSR1	SOCKET5状态寄存器
0x34A	0x34A	S5_PORT	S5_PORT0	SOCKET5的端口号
	0x34B		S5_PORT1	

0x34C	0x34C	S5_DHAR	S5_DHAR0	SOCKET5目的硬件地址寄存器
	0x34D		S5_DHAR1	
0x34E	0x34E	S5_DHAR2	S5_DHAR2	
	0x34F		S5_DHAR3	
0x350	0x350	S5_DHAR4	S5_DHAR4	
	0x351		S5_DHAR5	
0x352	0x352	S5_DPORTR	S5_DPORTR0	SOCKET5的目的端口号寄存器
	0x353		S5_DPORTR1	
0x354	0x354	S5_DIPR	S5_DIPR0	SOCKET5的目的IP地址寄存器
	0x355		S5_DIPR1	
0x356	0x356	S5_DIPR2	S5_DIPR2	
	0x357		S5_DIPR3	
0x358	0x358	S5_MSSR	S5_MSSR0	SOCKET5最大分片字节长度寄存器
	0x359		S5_MSSR1	
0x35A	0x35A	S5_PORTOR	S5_KPALVTR	SOCKET5生存期时间寄存器
	0x35B		S5_PROTOR	SOCKET5协议号寄存器
0x35C	0x35C	S5_TOSR	S5_TOSR0	保留
	0x35D		S5_TOSR1	SOCKET5的TOS寄存器
0x35E	0x35E	S5_TTLR	S5_TTLR0	保留
	0x35F		S5_TTLR1	SOCKET5的TTL寄存器
0x360	0x360	S5_TX_WRSR	S5_TX_WRSR0	保留
	0x361		S5_TX_WRSR1	SOCKET5的TX写入字节长度寄存器
0x362	0x362	S5_TX_WRSR2	S5_TX_WRSR2	
	0x363		S5_TX_WRSR3	
0x364	0x364	S5_TX_FSR	S5_TX_FSR0	保留
	0x365		S5_TX_FSR1	SOCKET5的TX剩余字节长度寄存器
0x366	0x366	S5_TX_FSR2	S5_TX_FSR2	
	0x367		S5_TX_FSR3	
0x368	0x368	S5_RX_RSR	S5_RX_RSR0	保留
	0x369		S5_RX_RSR1	SOCKET5的RX接收数据字节长度寄存器
0x36A	0x36A	S5_RX_RSR2	S5_RX_RSR2	
	0x36B		S5_RX_RSR3	
0x36C	0x36C	S5_FRAGR	S5_FRAGR0	保留
	0x36D		S5_FRAGR1	SOCKET5的IP标志段寄存器
0x36E	0x36E	S5_TX_FIFOR	S5_TX_FIFOR0	SOCKET5的TX FIFO寄存器
	0x36F		S5_TX_FIFOR1	
0x370	0x370	S5_RX_FIFOR	S5_RX_FIFOR0	SOCKET5的RX FIFO寄存器
	0x371		S5_RX_FIFOR1	
0x372	0x372			保留
	0x373			
…				
0x37E	0x37E			
	0x37F			

SOCKET6：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x380	0x380	S6_MR	S6_MR0	SOCKET6的模式寄存器
	0x381		S6_MR1	
0x382	0x382	S6_CR	S6_CR0	保留
	0x383		S6_CR1	SOCKET6命令寄存器
0x384	0x384	S6_IMR	S6_IMR0	保留
	0x385		S6_IMR1	SOCKET6中断屏蔽寄存器

0x386	0x386	S6_IR	S6_IR0	保留
	0x387		S6_IR1	SOCKET6中断寄存器
0x388	0x388	S6_SSR	S6_SSR0	保留
	0x389		S6_SSR1	SOCKET6状态寄存器
0x38A	0x38A	S6_PORT	S6_PORT0	SOCKET6的端口号
	0x38B		S6_PORT1	
0x38C	0x38C	S6_DHAR	S6_DHAR0	SOCKET6目的硬件地址寄存器
	0x38D		S6_DHAR1	
0x38E	0x38E	S6_DHAR2	S6_DHAR2	
	0x38F		S6_DHAR3	
0x390	0x390	S6_DHAR4	S6_DHAR4	
	0x391		S6_DHAR5	
0x392	0x392	S6_DPORTR	S6_DPORTR0	SOCKET6的目的端口号寄存器
	0x393		S6_DPORTR1	
0x394	0x394	S6_DIPR	S6_DIPR0	SOCKET6的目的IP地址寄存器
	0x395		S6_DIPR1	
0x396	0x396	S6_DIPR2	S6_DIPR2	
	0x397		S6_DIPR3	
0x398	0x398	S6_MSSR	S6_MSSR0	SOCKET6最大分片字节长度寄存器
	0x399		S6_MSSR1	
0x39A	0x39A	S6_PORTOR	S6_KPALVTR	SOCKET6生存期时间寄存器
	0x39B		S6_PROTOR	SOCKET6协议号寄存器
0x39C	0x39C	S6_TOSR	S6_TOSR0	保留
	0x39D		S6_TOSR1	SOCKET6的TOS寄存器
0x39E	0x39E	S6_TTLR	S6_TTLR0	保留
	0x39F		S6_TTLR1	SOCKET6的TTL寄存器
0x3A0	0x3A0	S6_TX_WRSR	S6_TX_WRSR0	保留
	0x3A1		S6_TX_WRSR1	SOCKET6的TX写入字节长度寄存器
0x3A2	0x3A2	S6_TX_WRSR2	S6_TX_WRSR2	
	0x3A3		S6_TX_WRSR3	
0x3A4	0x3A4	S6_TX_FSR	S6_TX_FSR0	保留
	0x3A5		S6_TX_FSR1	SOCKET6的TX剩余字节长度寄存器
0x3A6	0x3A6	S6_TX_FSR2	S6_TX_FSR2	
	0x3A7		S6_TX_FSR3	
0x3A8	0x3A8	S6_RX_RSR	S6_RX_RSR0	保留
	0x3A9		S6_RX_RSR1	SOCKET6的RX接收数据字节长度寄存器
0x3AA	0x3AA	S6_RX_RSR2	S6_RX_RSR2	
	0x3AB		S6_RX_RSR3	
0x3AC	0x3AC	S6_FRAGR	S6_FRAGR0	保留
	0x3AD		S6_FRAGR1	SOCKET6的IP标志段寄存器
0x3AE	0x3AE	S6_TX_FIFOR	S6_TX_FIFOR0	SOCKET6的TX FIFO寄存器
	0x3AF		S6_TX_FIFOR1	
0x3B0	0x3B0	S6_RX_FIFOR	S6_RX_FIFOR0	SOCKET6的RX FIFO寄存器
	0x3B1		S6_RX_FIFOR1	
0x3B2	0x3B2			保留
	0x3B3			
…				
0x3BE	0x3BE			
	0x3BF			

SOCKET7：

地址偏移		符号		描述
16位数据	8位数据	16位数据	8位数据	
0x3C0	0x3C0	S7_MR	S7_MR0	SOCKET7的模式寄存器
	0x3C1		S7_MR1	

0x3C2	0x3C2	S7_CR	S7_CR0	保留
	0x3C3		S7_CR1	SOCKET7命令寄存器
0x3C4	0x3C4	S7_IMR	S7_IMR0	保留
	0x3C5		S7_IMR1	SOCKET7中断屏蔽寄存器
0x3C6	0x3C6	S7_IR	S7_IR0	保留
	0x3C7		S7_IR1	SOCKET7中断寄存器
0x3C8	0x3C8	S7_SSR	S7_SSR0	保留
	0x3C9		S7_SSR1	SOCKET7状态寄存器
0x3CA	0x3CA	S7_PORT	S7_PORT0	SOCKET7的端口号
	0x3CB		S7_PORT1	
0x3CC	0x3CC	S7_DHAR	S7_DHAR0	SOCKET7目的硬件地址寄存器
	0x3CD		S7_DHAR1	
0x3CE	0x3CE	S7_DHAR2	S7_DHAR2	
	0x3CF		S7_DHAR3	
0x3D0	0x3D0	S7_DHAR4	S7_DHAR4	
	0x3D1		S7_DHAR5	
0x3D2	0x3D2	S7_DPORTR	S7_DPORTR0	SOCKET7的目的端口号寄存器
	0x3D3		S7_DPORTR1	
0x3D4	0x3D4	S7_DIPR	S7_DIPR0	SOCKET7的目的IP地址寄存器
	0x3D5		S7_DIPR1	
0x3D6	0x3D6	S7_DIPR2	S7_DIPR2	
	0x3D7		S7_DIPR3	
0x3D8	0x3D8	S7_MSSR	S7_MSSR0	SOCKET7最大分片字节长度寄存器
	0x3D9		S7_MSSR1	
0x3DA	0x3DA	S7_PORTOR	S7_KPALVTR	SOCKET7生存期时间寄存器
	0x3DB		S7_PROTOR	SOCKET7协议号寄存器
0x3DC	0x3DC	S7_TOSR	S7_TOSR0	保留
	0x3DD		S7_TOSR1	SOCKET7的TOS寄存器
0x3DE	0x3DE	S7_TTLR	S7_TTLR0	保留
	0x3DF		S7_TTLR1	SOCKET7的TTL寄存器
0x3E0	0x3E0	S7_TX_WRSR	S7_TX_WRSR0	保留
	0x3E1		S7_TX_WRSR1	SOCKET7的TX写入字节长度寄存器
0x3E2	0x3E2	S7_TX_WRSR2	S7_TX_WRSR2	
	0x3E3		S7_TX_WRSR3	
0x3E4	0x3E4	S7_TX_FSR	S7_TX_FSR0	保留
	0x3E5		S7_TX_FSR1	SOCKET7的TX剩余字节长度寄存器
0x3E6	0x3E6	S7_TX_FSR2	S7_TX_FSR2	
	0x3E7		S7_TX_FSR3	
0x3E8	0x3E8	S7_RX_RSR	S7_RX_RSR0	保留
	0x3E9		S7_RX_RSR1	SOCKET7的RX接收数据字节长度寄存器
0x3EA	0x3EA	S7_RX_RSR2	S7_RX_RSR2	
	0x3EB		S7_RX_RSR3	
0x3EC	0x3EC	S7_FRAGR	S7_FRAGR0	保留
	0x3ED		S7_FRAGR1	SOCKET7的IP标志段寄存器
0x3EE	0x3EE	S7_TX_FIFOR	S7_TX_FIFOR0	SOCKET7的TX FIFO寄存器
	0x3EF		S7_TX_FIFOR1	
0x3F0	0x3F0	S7_RX_FIFOR	S7_RX_FIFOR0	SOCKET7的RX FIFO寄存器
	0x3F1		S7_RX_FIFOR1	
0x3F2	0x3F2			保留
	0x3F3			
…				
0x3FE	0x3FE			
	0x3FF			

4. 寄存器描述

表示方法

1. 符号（名称） [R/W，RO，WO] [AO1/AO2] [Reset]
   Symbol：寄存器的符号
   Name：寄存器的名称
   R/W：读/写
   RO：只读
   WO：只写
   AO1：W5300 寄存器的物理地址，直接总线模式 CPU 系统中的地址映射
   AO2：W5300 寄存器的地址偏移，间接总线模式的地址
   Reset：复位后的默认值
   为了方便起见，假设 W5300 在 CPU 系统的映射地址为 0x08000，即 W5300 寄存器的物理基地址即为 0x08000

2. Pn_：缓存准备 PIN n（BTDYn）寄存器前缀
   Pn_BRDYR（BRDYn 配置寄存器，0 <= n <= 3）

3. Sn_：SOCKETn 寄存器前缀
   Sn_MR（SOCKETn 的模式寄存器，0 <= n <= 7）

4. 寄存器的描述方法

Symbol of low address Reg.	Bit 15	14	13	12	11	10	9	8
Physical Address	Symbol	-	-	-	-	-	-	-
Address offset	Reset Value	1	0	0	X	U(R)	0	0
Symbol of high address Reg.	Bit 7	6	5	4	3	2	1	0
Physical Address	Symbol	-	-	-	-	-	-	-
Address offset	Reset Value	0	0	0	0	0	0	0

- : Reserved Bit 1 : Logical High 0 : Logical Low
X : Don't Care U : 1 or 0 (R) : Read Only Bit

16 bit Register Symbol(AO1/AO2)	
8bit Register Symbol (AO1/AO2)	8bit Register Symbol (AO1/AO2)
MSB(Value)	LSB(Value)

4.1 模式（MR）寄存器

MR (Mode Register) [R/W] [0x08000/----][0x3800 or 0xB800]

MR寄存器设置W5300的工作模式，诸如接口模式，Sn_TX_FIFOR和Sn_RX_FIFOR的MSB/LSB的交换、软件复位、内部TX/RX存储器测试、数据总线的MSB/LSB交换和地址访问模式等。

MR0	15	14	13	12	11	10	9	8
0x08000	DBW	MPF	WDF2	WDF1	WDF0	RDH	-	FS
----	U(R)	0(R)	1	1	1	0	0	0
MR1	7	6	5	4	3	2	1	0
0x08001	RST	-	MT	PB	PPPoE	DBS	-	IND
----	0	0	0	0	0	0	0	0

MR(15:8)/MR0(7:0)

位	符号	描述
15	DBW	数据总线宽度 0：8位数据总线宽度 1：16位数据总线宽度 在W5300复位期间，这个值由BIT16EN引脚的电平确定。复位后，这个值不改变 参考“ 1.1 管脚定义 ”的BIT16EN的描述
14	MPF	MAC层终止数据报文 0：正常报文 1：终止报文 当从路由器或交换机收到终止报文时，该位置‘ 1 ’。当设置为‘ 1 ’时，将停止数据传输，直到该位为‘ 0 ’
13	WDF2	写数据访问时间 当主机写数据操作时，/CS为低电平后，W5300在WRF×PLL_CLK时间后取写入的数据 如果主机写操作在WRF×PLL_CLK完成（/CS恢复为高电平），写入的数据在‘ /CS ’为高电平时取走
12	WDF1	
11	WDF0	
10	RDH	读数据保持时间 0：没有数据保持时间 1：数据保持时间为2×PLL_CLK 在主机进行读操作时，当主机的读操作完成（/CS恢复高电平）后，W5300在2×PLL_CLK时间之内保持读取的数据。在这种情况下，注意数据总线上的数据冲突
9	-	保留
8	FS	FIFO交换 0：禁止交换 1：允许交换 它用于高字节和低字节的交换。W5300的字节一般采用大端模式。如果主机系统采用小端模式，那么将该位置‘ 1 ’，将Sn_TX_FIFOR 和Sn_RX_FIFOR的字节顺序交换，使用效果与小端模式相同

MR(7:0)/MR1(7:0)

位	符号	描述
7	RST	软件复位 该位置‘ 1 ’，对W5300软件复位。复位结束后自动清‘ 0 ’
6	-	保留
5	MT	存储器测试 0：禁止内部RX/TX存储器测试 1：允许内部存储器测试 一般来讲，W5300内部TX存储器支持主机通过Sn_TX_FIFOR寄存器的写操作，而内部RX存储器只支持主机通过Sn_RX_FIFOR寄存器的读操作。如果该位置‘ 1 ’，内部RX/TX存储器同时支出通过Sn_TX_FIFOR和Sn_RX_FIFOR的读写操作，从而校验内部TX/RX存储器。测试W5300内部TX/RX完成后，需要对系统重新复位或关闭端口 详细信息参考“ 怎样测试内部TX/RX存储器 ”
4	PB	Ping功能阻止模式 0：允许Ping 1：禁止Ping 当该位置‘ 1 ’，ICMP逻辑功能模块的自动‘ Ping ’应答功能被阻止。不产生对‘Ping’请求所进行的‘ Ping ’ 应答过程（ICMP请求响应） 即使Ping阻止模式为‘ 0 ’，当用户使用ICMP端口（Sn_MR[P3:P0] = Sn_MR_IPRAW和Sn_PROTOR1=0x01），自动Ping应答是不处理的。自动Ping应答支持最多119个字节
3	PPPoE	PPPoE模式 0：禁止PPPoE模式 1：启动PPPoE模式 当不通过路由器或其它设备与PPPoE服务器连接时，该位置‘ 1 ’。 详细信息参考“ 怎样在W5300上使用PPPoE ”
2	DBS	数据总线交换 0：禁止交换 1：允许交换 DBS位只交换Sn_TX_FIFOR/Sn_RX_FIFOR的高字节和低字节。然而该位交换所有寄存器的高字节和低字节，包括Sn_TX_FIFOR/ Sn_RX_FIFOR寄存器。该位只有DBW为‘ 1 ’时有效
1	-	保留
0	IND	间接总线模式 0：直接总线模式 1：间接总线模式 它设置W5300与主机的接口模式

4.2 间接模式寄存器

当MR（IND）=1时，W5300工作在间接寻址模式。目标主机系统仅通过MR、ID_MAR和IDM_DR就可以间接访问COMMON和SOCKET寄存器（也即是只需要访问映射在存储空间的MR，IDM_AR，IDM_DR寄存器）。

IDM_AR (间接模式地址寄存器) [R/W] [0x08002/----][0x0000]

它设置需要间接访问的COMMON寄存器或SOCKET寄存器的地址偏移。IDM_AR(0)或IDM_AR1(0)

是IDM_AR的最低位，该位忽略不计。

例）访问S4_RX_FIFOR (0x330)的方法如下：

IDM_AR0 = S4_RX_FIFOR寄存器地址偏移的MSB (0x03)

IDM_AR1 = S4_RX_FIFOR寄存器地址偏移的LSB (0x30)

IDM_AR(0x08002/----)	
IDM_AR0(0x08002/----)	IDM_AR1(0x08003/----)
0x03	0x30

IDM_DR(间接模式数据寄存器) [R/W] [0x08004/----][0x0000]

它用于间接访问COMMON寄存器和SOCKET寄存器的实际的值。IDM_DR0对应IDM_AR的高字节，IDM_DR1对应IDM_AR的低字节。

当使用8位数据总线访问任何寄存器的低字节时应该使用IDM_DR1寄存器。而访问高字节时则使用IDM_DR0寄存器。

它访问由IDM_AR地址寄存器的地址偏移量所确定的COMMON寄存器或SOCKET寄存器的实际值。

由地址IDM_AR所确定的高字节和低字节分别对应IDM_DR0和IDM_DR1寄存器。

在8位数据总线中，如果主机访问地址IDM_AR的低字节则使用IDM_DR1，访问高字节则使用IDM_DR0。

例1）当主机向IR (0x002) 写入0x80F0

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')
IDM_AR = 0x0002 IDM_DR = 0x80F0	IDM_AR0 = 0x00 IDM_AR1 = 0x02 IDM_DR0 = 0x80 IDM_DR1 = 0xF0

例2）主机读IR(0x0FE)且将该值存储在val变量中

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')
IDM_AR = 0x0002 val = IDM_DR	IDM_AR0 = 0x00 IDM_AR1 = 0x02 val = IDM_DR0 val = (val << 8) + IDM_DR1

IDM_AR(0x08002/----)	
IDM_AR0(0x08002/----)	IDM_AR1(0x08003/----)
0x00	0x02

IDM_DR(0x08004/----)	
IDM_DR0(0x08004/----)	IDM_DR1(0x08005/----)
MSB(IR0) of IR	LSB(IR1) of IR

4.3 COMMON寄存器

IR (中断寄存器) [R/W] [0x08002/0x002] [0x0000]

IR寄存器表示W5300向主机申请中断的类型。当任何中断产生时，IR相关的位置‘1’，而如果中断屏蔽寄存器的相应位也为‘1’，那么‘/INT’将输出低电平信号。

‘/INT’将保持低电平，直到IR寄存器的所有位都为‘0’。如果IR的所有位都为‘0’，‘/INT’将变为高电平。为了清除IR0中设置为‘1’的位，主机可以向它相应的位写入‘1’。而对于IR1中为‘1’的位，只有当相关的Sn_IR的所有位为‘0’，它将自动清除。

IR0	15	14	13	12	11	10	9	8
0x08002	IPCF	DPUR	PPPT	FMTU	-	-	-	-
0x002	0	0	0	0	0	0	0	0
IR1	7	6	5	4	3	2	1	0
0x08003	S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT
0x003	0	0	0	0	0	0	0	0

IR(15:8)/IR0(7:0)

位	符号	描述
15	IPCF	IP冲突 当IP地址产生冲突时，该位置‘ 1 ’时（当接收到ARP请求数据包的IP地址与W5300本机IP地址相同）。当它置‘ 1 ’时，网络中的另外一个设备使用了相同的IP地址，将造成通信错误。因此需要尽快采取措施解决这个问题
14	DPUR	目标端口无法到达 当收到ICMP（目的端口无法到达）数据包时，该位置‘ 1 ’ 参考UIPR和UPORTR
13	PPPT	PPPoE中止 在PPPoE模式，当与服务器连接关闭时，该位置‘ 1 ’
12	FMTU	分片最大传输单元（MTU） 当收到ICMP（分片最大传输单元）数据包时，该位置‘ 1 ’ 参考FMTUR
11	-	保留
10	-	保留
9	-	保留
8	-	保留

IR(7:0)/IR1(7:0)

位	符号	描述
7	S7_INT	SOCKET7中断 当SOCKET7产生中断时，该位置‘ 1 ’。该中断信息对应于S7_IR1。当S7_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’
6	S6_INT	SOCKET6中断 当SOCKET6产生中断时，该位置‘ 1 ’。该中断信息对应于S6_IR1。当S6_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’
5	S5_INT	SOCKET5中断 当SOCKET5产生中断时，该位置‘ 1 ’。该中断信息对应于S5_IR1。当S5_IR1被主机清‘ 0 ’后，该位自动清‘ 1 ’
4	S4_INT	SOCKET4中断 当SOCKET4产生中断时，该位置‘ 1 ’。该中断信息对应于S4_IR1。当S4_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’
3	S3_INT	SOCKET3中断 当SOCKET3产生中断时，该位置‘ 1 ’。该中断信息对应于S3_IR1。当S3_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’
2	S2_INT	SOCKET2中断 当SOCKET2产生中断时，该位置‘ 1 ’。该中断信息对应于S2_IR1。当S2_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’
1	S1_INT	SOCKET1中断 当SOCKET1产生中断时，该位置‘ 1 ’。该中断信息对应于S1_IR1。当S1_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’
0	S0_INT	SOCKET7中断 当SOCKET0产生中断时，该位置‘ 1 ’。该中断信息对应于S0_IR1。当S0_IR1被主机清‘ 0 ’后，该位自动清‘ 0 ’

IMR (中断屏蔽寄存器) [R/W] [0x08004/0x004] [0x0000]

它配置W5300的中断并报告给主机。IMR的每一个中断屏蔽位对应IR的每一个中断位。当IR的任何一个位为‘1’且相应的IMR位也为‘1’时，将向主机产生中断（‘/INT’输出低电平）。如果相应的IMR位为‘0’，将不产生中断（‘/INT’保持高电平），即使IR位为‘1’。

IMR0	15	14	13	12	11	10	9	8
0x08004	IPCF	DPUR	PPPT	FMTU	-	-	-	-
0x004	0	0	0	0	0	0	0	0
IMR1	7	6	5	4	3	2	1	0
0x08005	S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT
0x005	0	0	0	0	0	0	0	0

IMR(15:8)/IMR0(7:0)

位	符号	描述
15	IPCF	IR(IPCF)中断屏蔽
14	DPUR	IR(DPUR)中断屏蔽
13	PPPT	IR(PPPT)中断屏蔽
12	FMTU	IR(FMTU)中断屏蔽
11	-	保留
10	-	保留
9	-	保留
8	-	保留

IMR(7:0)/IMR1(7:0)

位	符号	描述
7	S7_INT	IR(S7_INT)中断屏蔽
6	S6_INT	IR(S6_INT)中断屏蔽
5	S5_INT	IR(S5_INT)中断屏蔽
4	S4_INT	IR(S4_INT)中断屏蔽
3	S3_INT	IR(S3_INT)中断屏蔽
2	S2_INT	IR(S2_INT)中断屏蔽
1	S1_INT	IR(S1_INT)中断屏蔽
0	S0_INT	IR(S0_INT)中断屏蔽

SHAR (本机硬件地址寄存器) [R/W] [0x08008/0x008] [00.00.00.00.00.00]

它配置本机的硬件地址（MAC地址）。

例）本机硬件地址“ 00.08.DC.01.02.03 ”

SHAR(0x08008/0x008)	
SHAR0(0x08008/0x008)	SHAR1(0x08009/0x009)
0x00	0x08
SHAR2(0x0800A/0x00A)	
SHAR2(0x0800A/0x00A)	SHAR3(0x0800B/0x00B)
0xDC	0x01
SHAR4(0x0800C/0x00C)	
SHAR4(0x0800C/0x00C)	SHAR5(0x0800D/0x00D)
0x02	0x03

GAR (网关IP地址) [R/W] [0x08010/0x010] [00.00.00.00]

它配置网关的IP地址。

例）当网关地址“ 192.168.0.1 ”

GAR(0x08010/0x010)		GAR2(0x08012/0x012)	
GAR0(0x08010/0x010)	GAR1(0x08011/0x011)	GAR2(0x08012/0x012)	GAR3(0x08013/0x013)
192(0xC0)	168(0xA8)	0(0x00)	1(0x01)

SUBR (子网掩码) [R/W] [0x08014/0x014] [00.00.00.00]

它配置子网掩码。

例）子网掩码为“ 255.255.255.0 ”

SUBR(0x08014/0x014)		SUBR2(0x08016/0x016)	
SUBR0(0x08014/0x014)	SUBR1(0x08015/0x015)	SUBR2(0x08016/0x016)	SUBR3(0x08017/0x017)
255 (0xFF)	255 (0xFF)	255 (0xFF)	0 (0x00)

SIPR (本机IP地址) [R/W] [0x08018/0x018] [00.00.00.00]

它用于配置本机的IP地址，或指示W5300通过PPPoE过程获得的IP地址。

例）本机IP地址为“ 192.168.0.3 ”

SIPR(0x08018/0x018)		SIPR2(0x0801A/0x01A)	
SIPR0(0x08018/0x018)	SIPR1(0x08019/0x019)	SIPR2(0x0801A/0x01A)	SIPR3(0x0801B/0x01B)
192(0xC0)	168(0xA8)	0(0x00)	3(0x03)

RTR (重复发送超时寄存器) [R/W] [0x0801C/0x01C] [0x07D0]

它配置重复发送超时周期的值。RTR的标准单位是100us，RTR初始化设置为2000（0x7D0），超时的时间周期为200ms。

例）当设置超时的时间周期为400ms时，RTR = (400ms / 1ms) X 10 = 4000(0x0FA0)

RTR(0x0801C/0x01C)	
RTR0(0x0801C/0x01C)	RTR1(0x0801D/0x01D)
0x0F	0xA0

RCR (重复发送计数寄存器) [R/W] [0x0801E/0x001E] [0x--08]

它配置重复发送的次数。当重复发送的次数达到‘ RCR+1 ’时，将产生超时中断（Sn_IR的‘ TIMEOUT ’位置‘ 1 ’）。

在TCP通信中，在Sn_IR(TIMEOUT)=1的同时Sn_SSR的状态改变为‘ SOCK_CLOSED ’。在UDP通信中，只有Sn_IR(TIMEOUT)=1。

例) RCR = 0x0007

RCR(0x0801E/0x01E)	
RCR0(0x0801E/0x01C)	RCR1(0x0801F/0x01F)
Reserved	0x07

W5300的超时可以通过RTR和RCR进行配置。W5300的超i时有ARP超时和TCP传输超时。

ARP（参考RFC826，http://www.ietf.org/rfc.html）发送超。W5300自动发送ARP请求到对端IP地址，以获得MAC地址信息（用于IP，UDP或TCP通信）。在等待对端ARP响应过程中，如果在RTR设置的时间范围内都没有响应，那么W5300将产生超时，且重复发送ARP请求。它将重复发送‘ RCR+1 ’次。

在ARP重复请求‘ RCR+1 ’次后如果没有ARP相应，那么最终将产生超时且Sn_IR(TIMEOUT)置‘ 1 ’。

ARP请求最终超时的计算如下：

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

TCP数据包传输超时。W5300传输TCP数据包（SYN，FIN，RST，DATA数据包）并在RTR和RCR设置的时间范围内等待响应（ACK）。如果对端没有响应将产生超时，且重复发送先前的TCP数据包。重复发送的次数为‘ RCR+1 ’。即使TCP数据包经过‘ RCR+1 ’次重复发送也没有得到对端的ACK响应，此时将产生最终的超时，在Sn_IR(TIMEOUT)=1时Sn_SSR的值为‘ SOCKET_CLOSED ’。

TCP数据包重复传输的最终超时的值计算如下：

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

其中：

N：重复发送计数，0≤N≤M

M：当$RTR \times 2^{(M+1)} > 65535$或0≤M≤RCR

RTR：$RTR \times 2^M$

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

ARP超时 = 2000 X 0.1ms X 9 = 1800ms = 1.8s

TCP超时 = (0x07D0 + 0x0FA0 + 0x1F40 + 0x3E80 + 0x7D00 + 0xFA00 + 0xFA00 + 0xFA00 + 0xFA00) X 0.1ms

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

= 318000 X 0.1ms = 31.8s

TMSR(TX存储器大小配置寄存器) [R/W] [0x08020/0x020] [08.08.08.08.08.08.08.08]

该寄存器用于设置每个SOCKET的内部TX存储器的大小。每个SOCKET的TX存储器大小的设置范围在0~64K字节之间。每个SOCKET在复位后分配8K字节。所有SOCKET的TX存储器的总合（TMS_{Sum}）应该是8的倍数。TMS_{SUM}和RMS_{Sum}(所有SOCKET的RX存储器的总合) 的总和为128K字节。

TMS01R(SOCKET0/1的TX存储器大小) [R/W] [0x08020/0x020] [0x0808]

它配置内部SOCKET0和SOCKET1的TX存储器的大小。

例1) SOCKET0：4KB，SOCKET1：16KB

TMS01R(0x08020/0x020)	
TMSR0(0x08020/0x020)	TMSR1(0x08021/0x021)
4 (0x04)	16 (0x10)

TMS23R(SOCKET2/3的TX存储器大小) [R/W] [0x08022/0x022] [0x0808]

它配置内部SOCKET2和SOCKET3的TX存储器大小。

例2) SOCKET2：1KB，SOCKET3：20KB

TMS23R(0x08020/0x020)	
TMSR2(0x08022/0x022)	TMSR3(0x08023/0x023)
1 (0x01)	20 (0x14)

TMS45R(SOCKET4/5的TX存储器大小) [R/W] [0x08024/0x024] [0x0808]

它配置内部SOCKET4和SOCKET5的TX存储器大小。

例3) SOCKET4：0KB，SOCKET5：7KB

TMS45R(0x08024/0x024)	
TMSR4(0x08024/0x024)	TMSR5(0x08025/0x025)
0 (0x00)	7 (0x07)

TMS67R(SOCKET6/7的TX存储器大小) [R/W] [0x08026/0x026] [0x0808]

它配置内部SOCKET6和SOCKET7的TX存储器大小。

例4) SOCKET6：12KB，SOCKET7：12KB

TMS67R(0x08026/0x026)	
TMSR6(0x08026/0x026)	TMSR7(0x08027/0x027)
12 (0x0C)	12 (0x0C)

从上面的例1~例4所示，TMS_{SUM}（TMSR0 + TMSR1 + TMSR2 + TMSR3 + TMSR4 + TMSR5 + TMSR6 + TMSR7）是72，它是8的倍数（72%8=0）。

RMSR(RX存储器大小配置寄存器) [R/W] [0x08028/0x028] [08.08.08.08.08.08.08.08]

它设置每个SOCKET内部RX存储器的大小。

每个SOCKET的存储器分配范围在0~64K字节。复位时每个SOCKET分配8K。RMS_{Sum}与TMS_{Sum}之和为128k字节。

RMS01R(SOCKET0/1的RX存储器大小) [R/W] [0x08028/0x028] [0x0808]

它配置SOCKET0和SOCKET1的存储器大小。

例5) SOCKET0：17KB，SOCKET1：3KB

RMS01R(0x08028/0x028)	
RMSR0(0x08028/0x028)	RMSR1(0x08029/0x029)
17 (0x11)	3 (0x03)

RMS23R(SOCKET2/3的RX存储器大小) [R/W] [0x0802A/0x02A] [0x0808]

它配置SOCKET2和SOCKET3的存储器大小。

例6) SOCKET2：5KB，SOCKET3：16KB

RMS23R(0x0802A/0x02A)	
RMSR2(0x0802A/0x02A)	RMSR3(0x0802B/0x02B)
5 (0x05)	16 (0x10)

RMS45R(SOCKET4/5的RX存储器大小) [R/W] [0x0802C/0x02C] [0x0808]

它配置SOCKET4和SOCKET5的存储器大小。

例7) SOCKET4：3KB，SOCKET5：4KB

RMS45R(0x0802C/0x02C)	
RMSR4(0x0802C/0x02C)	RMSR5(0x0802D/0x02D)
3 (0x03)	4 (0x04)

RMS67R(SOCKET6/7的RX存储器大小) [R/W] [0x0802E/0x02F] [0x0808]

它配置SOCKET6和SOCKET7的存储器分配大小。

例8) SOCKET6：4KB，SOCKET7：4KB

RMS67R(0x0802E/0x02E)	
RMSR6(0x0802E/0x02E)	RMSR7(0x0802F/0x02F)
4 (0x04)	4 (0x04)

从上面的例1~例8所示，RMS_{SUM}(RMSR0 + RMSR1 + RMSR2 + RMSR3 + RMSR4 + RMSR5 +RMSR6 + RMSR7)为56，TMS_{SUM} and RMS_{SUM}的总和为128。

MTYPER(存储器单元类型寄存器) [R/W] [0x08030/0x030] [0x00FF]

W5300内部RX/TX数据存储器是由16个8K字节的存储单元组成。MTYPER寄存器配置每个8K字节存储单元的类型——RX或TX存储器。每个8K字节的存储单元对应MTYPER的一个位。当该位为‘1’时，它用于TX存储器，当该位为‘0’时，它用于RX存储器。MTYPER的低位都配置为TX存储器。其余没有配置为TX存储器的都应该设置为‘0’。

MTYPER0	15	14	13	12	11	10	9	8
0x08030	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
0x030	0	0	0	0	0	0	0	0
MTYPER1	7	6	5	4	3	2	1	0
0x08031	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
0x031	1	1	1	1	1	1	1	1

MTYPER(15:8)/MTYPER0(7:0)

位	符号	描述
15	MB16	第16个存储器单元的类型
14	MB15	第15个存储器单元的类型
13	MB14	第14个存储器单元的类型
12	MB13	第13个存储器单元的类型
11	MB12	第12个存储器单元的类型
10	MB11	第11个存储器单元的类型
9	MB10	第10个存储器单元的类型
8	MB9	第9个存储器单元的类型

MTYPER(7:0)/MTYPER1(7:0)

位	符号	描述
7	MB7	第8个存储器单元的类型
6	MB6	第7个存储器单元的类型
5	MB5	第6个存储器单元的类型
4	MB4	第5个存储器单元的类型
3	MB3	第4个存储器单元的类型
2	MB2	第3个存储器单元的类型
1	MB1	第2个存储器单元的类型
0	MB0	第1个存储器单元的类型

例1) TMSSUM = 72, RMSSUM = 56

因为72/8=9，所以从MB8到MB0都设置为TX存储器

MTYPER(0x08030/0x030)	
MTYPER0(0x08030/0x030)	MTYPER1(0x08031/0x031)
0x01	0xFF

例2) TMSSUM = 128, RMSSUM = 0

MTYPER(0x08030/0x030)	
MTYPER0(0x08030/0x030)	MTYPER1(0x08031/0x031)
0xFF	0xFF

例3) TMSSUM = 0, RMSSUM = 128

MTYPER(0x08030/0x030)	
MTYPER0(0x08030/0x030)	MTYPER1(0x08031/0x031)
0x00	0x00

PATR (PPPoE认证类型寄存器) [R] [0x08032/0x032] [0x0000]

它表示W5300与PPPoE服务器连接时的认证方法

W5300支持2种类型的认证方法。

Value	Authentication method
0xC023	PAP
0xC223	CHAP

例) PATR = 'CHAP'

PATR(0x08032/0x032)	
PATR0(0x08032/0x032)	PATR1(0x08033/0x033)
0xC2	0x23

PTIMER(PPP连接控制协议请求定时寄存器) [R/W] [0x08036/0x036] [0x--28]

它配置连接控制协议（LCP）响应请求的传输时钟。值1大约为25ms。

例) PTIMER = 200 (200 * 25ms = 5000ms = 5s)

PTIMER(0x08036/0x037)	
PTIMER0(0x08036/0x036)	PTIMER1(0x08037/0x037)
Reserved	200 (0xC8)

PMAGICR(PPP连接控制协议（LCP）的魔数) [R/W] [0x08038/0x038] [0x--00]

它配置4个字节的魔数，用于连接控制协议（LCP）与PPPoE服务器握手。详细信息请查看“ 怎样在W5300上实现PPPoE ”。

例) PMACIC = 0x01

PMAGICR(0x08036/0x037)	
PMAGICR0(0x08038/0x038)	PMAGICR1(0x08039/0x039)
Reserved	0x01

魔数 = 0x01010101

PSIDR(PPPoE会话ID寄存器)[R][0x0803C/0x03C][0x0000]

它报告PPP会话ID，用于与PPPoE服务器通信（由W5300的PPPoE处理过程获得）

例) PMACIC = 0x0017

PSIDR(0x0803C/0x03C)	
PSIDR0(0x0803C/0x03C)	PSIDR1(0x0803D/0x03D)
0x00	0x17

PDHAR(PPPoE目的硬件地址寄存器)[R][0x08040/0x040][00.00.00.00.00.00]

它报告PPPoE服务器的硬件地址（由W5300的PPPoE处理过程获得）。

例) PDHAR = 00.01.02.03.04.05

PDHAR(0x08040/0x040)	
PDHAR0(0x08040/0x040)	PDHAR1(0x08041/0x041)
0x00	0x01
PDHAR2(0x08042/0x042)	
PDHAR2(0x08042/0x042)	PDHAR3(0x08043/0x043)
0x02	0x03
PDHAR4(0x08044/0x044)	
PDHAR4(0x08044/0x044)	PDHAR5(0x08045/0x045)
0x04	0x05

UIPR (无法到达的IP地址寄存器) [R] [0x08048/0x048] [00.00.00.00]

UPORTR (无法到达的端口号寄存器) [R] [0x0804C/0x04C] [0x0000]

当W5300试着向一个未打开的目的端口发送UDP数据时，W5300会收到ICMP（目的端口不存在）数据包。在这种情况下，IR（DPUR）会变成‘1’，并且通过UIPR和UPORTR可以获得ICMP数据包的目标IP地址和无法到达的端口号。

例1) UIPR = 192.168.0.11

UIPR(0x08048/0x048)		UIPR2(0x0804A/0x04A)	
UIPR0(0x08048/0x048)	UIPR1(0x08049/0x049)	UIPR2(0x0804A/0x04A)	UIPR3(0x0804B/0x04B)
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

例2) UPORT = 5000(0x1388)

UPORTR(0x0804C/0x04C)	
UPORTR0(0x0804C/0x04C)	UPORTR1(0x0804D/0x04D)
0x13	0x18

FMTUR (分片的最大发送单元(MTU)寄存器) [R] [0x0804E/0x04E] [0x0000]

如果本机的MTU和通信对端的MTU不相同时，W5300将收到ICMP(分片MTU)数据包。这时，IR(FMTU)寄存器的位置‘1’， ICMP数据包的目标IP地址和分片MTU的值都可以从UIPR和FMTUR寄存器中获得。为了保持与对端的通信分片具有分片MTU，先在SOCKETn的Sn_MSSR中设置FMTUR，然后再试下一次的通信。

例) FMTUR = 512(0x200)

FMTUR(0x0804E/0x04E)	
FMTUR0(0x0804E/0x04E)	FMTUR1(0x0804F/0x04F)
0x02	0x00

Pn_BRDYR (引脚‘BRDYn’的配置寄存器) [R/W] [0x08060+4n/0x060+4n] [0x--00]

该寄存器配置‘BRDYn’引脚，用于监控指定端口的TX/RX存储器的状态。如果发送存储器TX的剩余空间与Pn_BDPTHR相等或更大，或接收存储器RX的空间与Pn_BDPTHR相等或更大，引脚‘BRDYn’输出信号。

Pn_BRDYR0	15	14	13	12	11	10	9	8
0x08060 + 4n	-	-	-	-	-	-	-	-
0x060 + 4n	0	0	0	0	0	0	0	0
Pn_BRDYR1	7	6	5	4	3	2	1	0
0x08061	PEN	PMT	PPL	-	-	SN2	SN1	SN0
0x061	0	0	1	0	0	0	0	0

Pn_BRDYR(7:0)/Pn_BRDYR1(7:0)

位	符号	描述
7	PEN	引脚‘ BRDYn ’有效 0：禁止‘ BRDYn ’ 1：允许‘ BRDYn ’，如果允许‘ BRDYn ’引脚，该位置‘ 1 ’
6	PMT	引脚存储器类型 0：接收存储器 1：发送存储器 它设置要监控的端口存储器的类型
5	PPL	引脚极性 0：低电平激活 1：高电平激活 当剩余接收/发送缓冲区的大小等于或大于Pn_DPTHR时，它将通过‘ BRDYn ’引脚输出指定的逻辑信号给主机
4	-	保留
3	-	保留
2	SN2	端口号 它设置被监控的端口
1	SN1	
0	SN0	

	SN2	SN1	SN0		SN2	SN1	SN0
7	1	1	1	3	0	1	1
6	1	1	0	2	0	1	0
5	1	0	1	1	0	0	1
4	1	0	0	0	0	0	0

P0_BRDYR (引脚‘ BRDY0 ’配置寄存器) [R/W] [0x08060/0x060] [0x--00]

它配置引脚‘ BRDY0 ’。

P1_BRDYR (引脚‘ BRDY1 ’配置寄存器) [R/W] [0x08064/0x064] [0x--00]

它配置引脚‘ BRDY1 ’。

P2_BRDYR (引脚‘ BRDY2 ’配置寄存器) [R/W] [0x08068/0x068] [0x--00]

它配置引脚‘ BRDY2 ’。

P3_BRDYR (引脚‘ BRDY3 ’配置寄存器) [R/W] [0x0806C/0x06C] [0x--00]

它配置引脚‘ BRDY3 ’。

Pn_BDPTHR (引脚‘ BRDYn ’缓冲区门限寄存器) [R/W] [0x08062/0x062] [0xUUUU]

它用于配置‘ BRDYn ’的缓冲区的大小。当监控TX存储器，Sn_TX_FSR等于或大于Pn_DEPTHR时，引脚‘ BRDYn ’输出信号。当监控RX存储器，Sn_RX_FSR等于或大于Pn_DEPTHR时，引脚‘ BRDYn ’输出信号。Pn_DPTHR的值不能超过由TMSR或RMSR定义的TX/RX存储器的大小。

P0_BDPTHR (引脚‘ BRDY0 ’缓冲区门限寄存器) [R/W] [0x08062/0x062] [0xUUUU]

设置引脚‘ BRDY0 ’的缓冲区监控门限。

P1_BDPTHR (引脚‘ BRDY1 ’缓冲区门限寄存器) [R/W] [0x08066/0x066] [0xUUUU]

设置引脚‘ BRDY1 ’的缓冲区监控门限。

P2_BDPTHR (引脚‘ BRDY2 ’缓冲区门限寄存器) [R/W] [0x0806A/0x06A] [0xUUUU]

设置引脚‘ BRDY2 ’的缓冲区监控门限。

P3_BDPTHR (引脚‘ BRDY3 ’缓冲区门限寄存器) [R/W] [0x0806E/0x06E] [0xUUUU]

设置引脚‘ BRDY3 ’的缓冲区监控门限。

例）监控SOCKET5的TX存储器，，当剩余字节数为2048字节时，引脚‘ BRDY3 ’输出高电平，

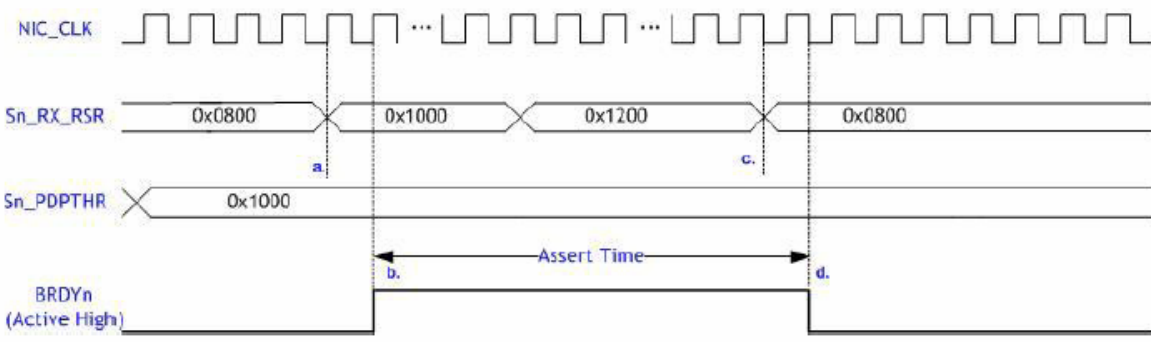
P3_BRDYR = 0x00E5

P3_BRDYR(0x0806C/0x06C)	
P3_BRDYR0(0x0806C/0x06C)	P3_BRDYR1(0x0806D/0x06D)
Reserved	0xE5

P3_BDPTHR = 2048(0x0800)

P3_BDPTHR(0x0806E/0x06E)	
P3_BDPTHR0(0x0806E/0x06E)	P3_BDPTHR1(0x0806F/0x06F)
0x08	0x00

当使用‘ BRDYn ’监控SOCKETn的RX存储器时，‘ BRDYn ’的时序如下：



a. 检测到Sn_RX_RSR > Sn_BDPTHR

b. 过1个NIC_CLK，引脚‘ BRDYn ’输出高电平

c. 主机读取RX存储器时，Sn_RX_RSR减少，检测到Sn_RX_RSR < Sn_BDPTHR，

d. 经过1个NIC_CLK，引脚‘ BRDYn ’输出低电平。

声明时间：BRDYn的作用时间。当Sn_RX_RSR > Sn_BDPTHR时，它一直保持状态不变（最少80ns）

IDR (标识寄存器) [R] [0x080FE/0x0FF] [0x5300]

它标识W5300的ID值

IDR(0x080FE/0x0FE)	
FMTUR0(0x080FE/0x0FE)	FMTUR1(0x080FF/0x0FF)
0x53	0x00

4.4 SOCKET寄存器

Sn_MR (SOCKETn模式寄存器) [R/W] [0x08200+0x40n/0x200+0x40n] [0x0000]

它配置SOCKETn的协议类型和一些选项。

Sn_MR0	15	14	13	12	11	10	9	8
0x08200 + 0x40n	-	-	-	-	-	-	-	ALIGN
0x200 + 0x40n	0	0	0	0	0	0	0	0
Sn_MR1	7	6	5	4	3	2	1	0
0x08201 + 0x40n	MULTI	-	ND/MC	-	P3	P2	P1	P0
0x201 + 0x40n	0	0	1	0	0	0	0	0

Sn_MR(15:8)/Sn_MR0(7:0)

位	符号	描述
15	-	保留
14	-	保留
13	-	保留
12	-	保留
11	-	保留
10	-	保留
9	-	保留
8	ALIGN	队列对齐 0：不使用对齐 1：使用对齐 只有在TCP模式下有效（P3~P0：0001） 在TCP通信过程中，当每次收到的数据包的字节数为偶数时，该位置为‘1’，删去附在接收数据包中的PACKET-INFO（数据的字节数），可以使读取数据的操作大大增强。详细信息请参考“5.2.1.1 TCP服务器”

Sn_MR(7:0)/Sn_MR1(7:0)

位	符号	描述
7	MULTI	多播 0：禁止多播 1：允许多播 只有在UDP模式下有效（P3~P0：0010） 为了实现多播通信，在打开端口命令之前需要分别在Sn_DIPR设置中设置IP地址和在Sn_DPORTR中设置端口号
6	MF	MAC地址过滤 0：禁止MAC地址过滤 1：允许MAC地址过滤 只有在MACRAW模式下有效（P3~P0：0100） 当该位置‘1’，W5300只能接收属于它自己的数据包或广播数据包。如果该位置‘0’，W5300可以接收以太网所有的数据包。当使用混合TCP/IP协议栈，建议将该位置‘1’，以减少主机接收过多的数据

<table>
<tr><td>5</td><td>ND/IGMPv</td><td>使用无延时的ACK
0：禁止延时ACK选项
1：允许延时ACK选项
只有在TCP模式下有效（P3~P0：0001）
当该位置‘1’，收到对端的数据包后立即发送ACK数据包响应。建议将该位置‘1’，以提高TCP通信的性能。当该位置‘0’时，不是在接收到数据包就发送ACK响应，而是只有设置了RTR后才发送ACK数据包响应

IGMP版本
0：使用IGMP的版本2
1：使用IGMP的版本1
只有在MULTI=1和UDP模式(P3~P0：0010)下有效
它配置IGMP的版本，发送IGMP信息，诸如：加入/离开/向多播组报告等</td></tr>
<tr><td>4</td><td>-</td><td>保留</td></tr>
<tr><td>3</td><td>P3</td><td rowspan="3">协议类型。它用于配置每个SOCKET的通信协议（TCP、UDP、IPRAW，MACRAW等）或PPPoE SOCKET与PPPoE服务器之间的操作
<table>
<tr><th>Symbol</th><th>P3</th><th>P2</th><th>P1</th><th>P0</th><th>Meaning</th></tr>
<tr><td>Sn_MR_CLOSE</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Closed</td></tr>
<tr><td>Sn_MR_TCP</td><td>0</td><td>0</td><td>0</td><td>1</td><td>TCP</td></tr>
<tr><td>Sn_MR_UDP</td><td>0</td><td>0</td><td>1</td><td>0</td><td>UDP</td></tr>
<tr><td>Sn_MR_IPRAW</td><td>0</td><td>0</td><td>1</td><td>1</td><td>IP RAW</td></tr>
<tr><td>S0_MR_MACRAW</td><td>0</td><td>1</td><td>0</td><td>0</td><td>MAC RAW</td></tr>
<tr><td>S0_MR_PPPoE</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr>
</table>
S0_MR_MACRAW和S0_MR_PPPOE只对SOCKET0有效
S0_MR_PPPOE只在与PPPoE服务器连接/断开时暂时使用。PPPoE连接建立以后，SOCKET0则可以用于其它的协议。</td></tr>
<tr><td>2</td><td>P2</td></tr>
<tr><td>1</td><td>P1</td></tr>
<tr><td>0</td><td>P0</td></tr>
</table>

Sn_CR (SOCKETn命令寄存器) [R/W] [0x08202+0x40n/0x202+0x40n] [0x--00]

它设置命令类型，诸如：打开端口、关闭端口、端口建立连接、端口发送数据及接收数据等等。当W5300接收到任何命令时，Sn_CR自动清为0x00。即使Sn_CR被清0x00，命令仍然在执行。命令执行的结果可以在Sn_IR或Sn_SSR中查询。

<table>
<tr><td colspan="2">Sn_CR(0x08202+0x40n/0x202+0x40n)</td></tr>
<tr><td>Sn_CR0(0x08202+0x40n/0x202+0x40n)</td><td>Sn_CR1(0x08203+0x40n/0x203+0x40n)</td></tr>
<tr><td>Reserved</td><td>Command</td></tr>
</table>

Sn_CR(7:0)/Sn_CR1(7:0)

<table>
<tr><th>值</th><th>命令</th><th>描述</th></tr>
<tr><td>0x01</td><td>打开端口
OPEN</td><td>它根据Sn_MR(P3~P0)所定义的协议类型初始化端口并打开端口
<table>
<tr><th>Sn_MR(P3:P0)</th><th>Sn_SSR</th></tr>
<tr><td>Sn_MR_CLOSE</td><td>-</td></tr>
<tr><td>Sn_MR_TCP</td><td>SOCK_INIT</td></tr>
<tr><td>Sn_MR_UDP</td><td>SOCK_UDP</td></tr>
<tr><td>Sn_MR_IPRAW</td><td>SOCK_IPRAW</td></tr>
<tr><td>S0_MR_MACRAW</td><td>SOCK_MACRAW</td></tr>
<tr><td>S0_MR_PPPoE</td><td>SOCK_PPPoE</td></tr>
</table></td></tr>
<tr><td>0x02</td><td>侦听
LISTEN</td><td>只有在TCP模式下有效(Sn_MR(P3:P0)=Sn_MR_TCP)
它将SOCKETn设置为TCP服务器模式。它将改变Sn_SSR寄存器的SOCK_INIT为SOCK_LISTEN，以等待其它TCP客户端的连接请求（SYN数据包）
当Sn_SSR为SOCK_LISTEN且成功处理了其它TCP客户端的连接请求时，Sn_IR(0)将置‘1’，而Sn_SSR变为SOCK_ESTABLISHED。如果没有处理连接请求(SYN/ACK传输失败)，TCP产生超时（Sn_IR(3)=1）且Sn_SSR变为SOCK_CLOSED
另外，如果TCP客户端连接请求的端口号不存在，W5300将发送RST数据包且Sn_SSR的状态不改变</td></tr>
<tr><td>0x04</td><td>连接
CONNECT</td><td>只有在TCP模式下有效
它将端口设置为TCP客户端模式
它发送连接请求到由Sn_DIPR和Sn_DPORTR指定的TCP服务器
当连接请求被成功处理(收到SYN/ACK数据包)，Sn_IR(0)置‘1’，且Sn_SSR的状态变为SOCK_ESTABLISHED
如果连接失败，可能有三种情况
1．ARP产生超时，因为目标硬件地址无法获得
2．没有收到SYN/ACK数据包而产生超时(Sn_IR(3)=1)
3．收到RST数据包而不是SYN/ACK数据包
以上三种情况Sn_SSR都将变为SOCK_CLOSED状态</td></tr>
<tr><td>0x08</td><td>断开连接
DISCON</td><td>只有在TCP模式下有效
不论是TCP服务器还是客户端，它都将执行断开连接的处理。
1．主动关闭：它发送断开连接的请求(FIN数据包)到连接的对端
2．被动关闭：当收到对端的断开连接请求(FIN数据包)时，它发送FIN数据包
如果断开连接成功(收到对端的FIN/ACK数据包)，Sn_SSR的状态将变为SOCK_CLOSED
如果断开连接失败，产生TCP超时(Sn_IR(3)=1)且Sn_SSR的状态变为SOCK_CLOSED
另外，如果直接使用CLOSE命令而不是DISCON命令，只有Sn_SSR的状态变为SOCK_CLOSED，不产生断开连接的处理(断开连接的请求)。如果在通信过程中收到对端发送来的RST数据包，Sn_SSR无条件变为SOCK_CLOSED状态</td></tr>
<tr><td>0x10</td><td>端口关闭
CLOSE</td><td>它关闭端口，Sn_SSR的状态变为SOCK_CLOSED</td></tr>
</table>

0x20	发送数据 SEND	它发送启动数据发送，发送的字节长度由Sn_TX_WRSR确定 在TCP或UDP模式，如果Sn_TX_WRSR大于最大分片字节数(MSS)，W5300将自动根据裁分为MSS单元，且发送已经裁分的数据 然而这个功能不支持IPRAW和MACRAW模式。主机应该将数据裁分为MSS单元再传送。 当发送过程结束，Sn_IR(SENDOK)将置1，主机检测到Sn_IR(SENDOK)=1后，可以进行下一次的传输 如果通过SEND命令数据包成功传输到对端（当收到对端的DATA/ACK数据包），Sn_TX_FSR根据传输的数据长度自动增加。如果没有传输成功（没有收到DATA/ACK的数据包），将产生超时（Sn_IR(3)=1），且Sn_SSR进入SOCK_CLOSED状态 另外，主机在使用SEND命令发送数据之前，首先通过Sn_TX_FIFOR寄存器将数据写入到TX存储器，然后写入要发送数据的字节数到Sn_TX_WRSR
0x21	发送数据到MAC地址 SEND_MAC	只有在UDP模式下 (Sn_MR(P3:P0)=Sn_MR_UDP) 或 IPRAW(Sn_MR(P3:P0)=Sn_MR_IPRAW)模式下有效 基本操作与SEND相同 在通过ARP过程获得了目标硬件地址后SEND命令传输数据，而SEND_MAC命令则将Sn_DHAR作为目标硬件地址发送数据。在发送UDP或IPRAW数据到目的地址时，由于减少了ARP的处理，SEND_MAC可以减少网络阻塞。
0x22	发送保持 SEND_KEEP	只有在TCP模式下有效 为了检查与对端的TCP连接状态，发送一个KEEP ALIVE（KA）数据包。 SEND_KEEP只有在‘Sn_KPALVTR=0’时有效，在‘Sn_KPALVTR>0’时无效。当‘Sn_KPALVTR>0’时，KA数据包在没有数据时自动发送 如果KA数据包成功发送（当收到对端的KA/ACK数据包时），Sn_SSR保持SOCK_ESTABLISHED状态。如果KA数据包发送不成功（对端已经断开连接或美元发送KA/ACK），将产生超时中断（Sn_IR(3)=1），且Sn_SSR改变为SOCK_CLOSED状态 另外，KA数据包可以在一个或更多的数据通信处理后发送
0x40	接收数据 RECV	它表示主机接收到SOCKETn的数据 在使用RECV命令前，主机需要通过Sn_RX_FIFOR寄存器从RX存储器读取接收的数据。

下面的命令只对SOCKET0有效且S0_MODE(P3~P0)=S0_MR_PPPOE。详细资料请查看“怎样在W5300上实现PPPoE”。

值	命令	描述
0x23	PCON	通过传输PPPoE搜索数据包开始PPPoE的连接
0x24	PDISCON	关闭PPPoE连接
0x25	PCR	在每一个过程，它传输REQ消息
0x26	PCN	在每一个过程，它传输NAK消息
0x27	PCJ	在每一个过程，他传输REJECT消息

Sn_IMR (SOCKETn中断屏蔽寄存器)[R/W] [0x08204+0x40n/0x204+0x40n] [0x--FF]

它配置SOCKETn向主机产生的中断。

Sn_IMR的中断屏蔽位与Sn_IR是对应的。当任何SOCKET产生中断并且中断标志位被置‘1’，相应的Sn_IR的对应位被置‘1’，当Sn_IMR和Sn_IR的位都为‘1’，IR(n)变为‘1’。这时，W5300将向主机发出中断请求（‘/INT’变为低电平）。

Sn_IMR0	15	14	13	12	11	10	9	8
0x08204 + 0x40n	-	-	-	-	-	-	-	-
0x204 + 0x40n	0	0	0	0	0	0	0	0
Sn_IMR1	7	6	5	4	3	2	1	0
0x08205 + 0x40n	PRECV	PFAIL	PNEXT	SENDOK	TIMEOUT	RECV	DISCON	CON
0x205 + 0x40n	1	1	1	1	1	1	1	1

Sn_IMR(15:8)/Sn_IMR0(7:0)：所有都保留

Sn_IMR(7:0)/Sn_IMR1(7:0)

位	符号	描述
7	PRECV	Sn_IR(PRECV)中断屏蔽 只有在SOCKET0和S0_MR(P3:P0)=S0_MR_PPPOE模式下有效
6	PFAIL	Sn_PFAIL(PFAIL)中断屏蔽 只有在SOCKET0和S0_MR(P3:P0)=S0_MR_PPPOE模式下有效
5	PNEXT	Sn_PNEXT(PNEXT)中断屏蔽 只有在SOCKET0和S0_MR(P3:P0)=S0_MR_PPPOE模式下有效
4	SENDOK	Sn_IR(SENDOK) 中断屏蔽
3	TIMEOUT	Sn_IR(TIMEOUT) 中断屏蔽
2	RECV	Sn_IR(RECV) 中断屏蔽
1	DISCON	Sn_IR(DISCON) 中断屏蔽
0	CON	Sn_IR(CON) 中断屏蔽

Sn_IR (SOCKETn中断寄存器) [R/W] [0x08206+0x40n/0x206+0x40n] [0x--00]

Sn_IR寄存器向主机指示SOCKETn的中断类型（建立连接，中断，接收数据，超时等状态）。

当任何中断产生且中断屏蔽Sn_IMR相应的位为‘1’，中断寄存器Sn_IR相应的位变为‘1’。

为了清除Sn_IR中置‘1’的位，主机需要向相应的位写‘1’。当所有的Sn_IR位都为‘0’，IR(n)自动清除。

Sn_IR0	15	14	13	12	11	10	9	8
0x08206 + 0x40n	-	-	-	-	-	-	-	-
0x206 + 0x40n	0	0	0	0	0	0	0	0
Sn_IR1	7	6	5	4	3	2	1	0
0x08207 + 0x40n	PRECV	PFAIL	PNEXT	SENDOK	TIMEOUT	RECV	DISCON	CON
0x207 + 0x40n	0	0	0	0	0	0	0	0

Sn_IR(15:8)/Sn_IR0(7:0)：全部保留

Sn_IR(7:0)/Sn_IR1(7:0)

位	符号	描述
7	PRECV	PPP接收中断 接收到不支持的可选数据（Option Data）时，该位置位
6	PFAIL	PPP失败中断 PAP认证失败时该位置位
5	PNEXT	PPP下一过程中断 在PPPoE连接过程中，该过程改变时置位
4	SENDOK	发送完成中断 SEND命令完成后置位
3	TIMEOUT	超时中断 在ARP和TCP过程中超时置位
2	RECV	接收数据中断 端口从对端接收到数据时置位
1	DISCON	断开连接中断 接收到从对端来的FIN或FIN/ACK数据包时置位
0	CON	连接中断 与对端成功建立连接时置位

Sn_SSR (SOCKETn 状态寄存器) [R] [0x08208+0x40n/0x208+0x40n] [0x--00]

它指示SOCKETn的状态。SOCKETn的状态可以由Sn_CR命令或数据包的收发而改变。

Sn_SSR(0x08208+0x40n/0x208+0x40n)	
Sn_SSR0(0x08208+0x40n/0x208+0x40n)	Sn_SSR1(0x08209+0x40n/0x209+0x40n)
Reserved	SOCKET Stauts

Sn_SSR(15:8)/Sn_IR0(7:0)：全部保留

Sn_SSR(7:0)/Sn_SSR1(7:0)

值	符号	描述
0x00	SOCK_CLOSED	SOCKETn端口资源释放状态 当执行DISCON或CLOSE命令，或产生ARP、TCP超时，不管以前是什么状态，此时它都变为SOCK_CLOSED状态
0x13	SOCK_INIT	SOCKETn以TCP模式打开时的状态 当Sn_MR(P3~P0)为Sn_MR_TCP且执行OPEN命令时，它变为SOCK_INIT状态。它是建立TCP连接的第一步 这时可以使用LISTEN命令设置TCP服务器模式，或CONNECT命令设置TCP客户端模式

0x14	SOCK_LISTEN	它是SOCKETn在TCP服务器状态，等待TCP客户端的连接请求(SYN数据包) 当运行LISTEN命令时，它改变为SOCK_LISTEN状态 当成功处理了TCP客户端的连接请求(SYN数据包)，SOCK_LISTEN变为SOCK_ESTABLISHED。如果失败，将产生超时中断(Sn_IR(TIMEOUT)= 1)，且状态改变为SOCK_CLOSED
0x17	SOCK_ESTABLISHED	它是TCP建立连接的状态 在SOCK_LISTEN状态，收到TCP客户端SYN数据包并成功处理，它将变成SOCK_ESTABLISHED，或CONNECT命令成功运行。在这种状态，可以进行数据传输，即可以运行SEND或RECV命令
0x1C	SOCK_CLOSE_WAIT	该状态是收到对端断开连接请求(FIN数据包) 由于TCP连接处于半关闭状态，但可以进行数据传输。为了彻底断开TCP连接，必须执行DISCON命令。如果关闭SOCKEn而没有断开连接的处理，可以只运行CLOSE命令
0x22	SOCK_UDP	SOCKETn处于UDP模式 当Sn_MR(P3:P0)为Sn_MR_UDP且执行OPEN命令，它将变为SOCK_UDP。端口不需要像TCP那样连接即可以进行数据通信
0x32	SOCK_IPRAW	SOCKETn处于IPRAW模式 当Sn_MR(P3:P0)为Sn_MR_IPRAW且执行OPEN命令，它将变为SOCK_IPRAW。可以进行IP数据包传输。IP数据包不需要像SOCK_TCP那样连接即可以进行传输
0x42	SOCK_MACRAW	SOCKET0以MACRAW模式打开 当Sn_MR(P3:P0)为Sn_MR_MACRAW且执行OPEN命令，它将变为SOCK_MACRAW。MAC数据包（以太网数据帧）的传输与SOCK_UDP相同
0x5F	SOCK_PPPoE	SOCKET0以PPPoE模式打开 当Sn_MR(P3:P0)为Sn_MR_PPPOE且执行OPEN命令，它将暂时用于PPPoE的连接 详细信息请参考“ 怎样在W5100上实现PPPoE ”

以下是Sn_SSR状态改变过程中出现的临时状态。

值	符号	描述
0x00	SOCK_SYNSENT	该状态表示连接请求(SYN数据包)发送到TCP服务器 该状态显示CONNECT命令从SOCK_INIT到SOCK_ESTABLISHED的状态改变过程 在这种状态，如果收到TCP服务器允许连接信息(SYN/ACK数据包)，状态自动转换为SOCK_ESTABLISHED。如果在产生TCP超时（Sn_IR(TIMEOUT)=1）之前没有收到TCP服务器的SYN/ACK数据包，那么它自动转变为SOCK_CLOSED
0x16	SOCK_SYNRECV	该状态表示收到TCP客户端的连接请求(SYN数据包) 当W5300向TCP客户端发出允许连接(SYN/ACK数据包)信息后，它自动变换为SOCK_ESTABLISHED。如果失败，将产生超时（Sn_IR(TIMEOUT)=1），且改变为SOCK_CLOSED。
0x18	SOCK_FIN_WAIT	SOCKETn被关闭的状态 当SOCKET完成主动关闭或被动关闭的断开连接处理时出现这种状态。当完成断开连接处理或TCP超时（Sn_IR (TIMEOUT)=1），它的状态将改变为SOCK_CLOSED
0X1B	SOCK_TIME_WAIT	
0X1D	SOCK_LAST_ACK	

0x01	SOCK_ARP	该状态表示已发送ARP请求，以获取目的硬件地址 在SOCK_UDP或SOCK_IPRAW状态执行了SEND命令，或在SOCK_INIT状态执行了CONNECT命令，将产生该状态 如果成功获得了目的硬件地址（收到ARP响应），它的状态将改变为SOCK_UDP、SOCK_IPRAW或SOCK_SYNSENT。如果不成功并产生超时(Sn_IR(TIMEOUT)=1)，在UDP或IPRAW模式，它将回到先前的状态（SOCK_UDP或SOCK_IPRAW），在TCP模式，将回到SOCK_CLOSED状态 另外，在SOCK_UDP和SOCK_IPRAW状态，当SEND命令所执行的当前Sn_DIPR与上一次不相同，也要执行ARP操作。如果当前的Sn_DIPR与上一次的相同，则使用前面已经得到的硬件地址

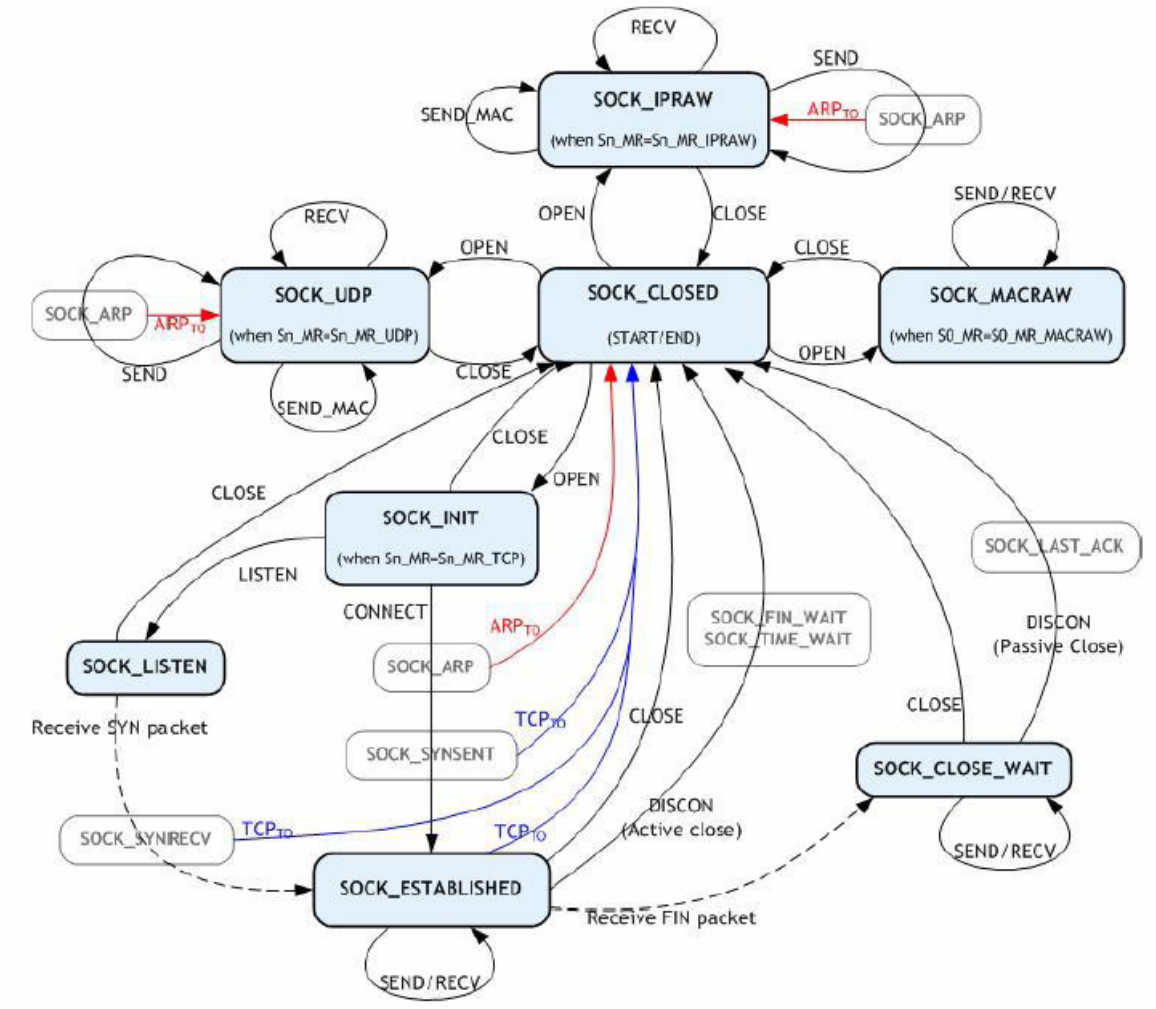


Fig 5. SOCKETn状态变化

Sn_PORTR(SOCKETn源端口号寄存器) [R/W] [0x0820A+0x40n/0x20A+0x40A] [0x0000]

它设置源端口的端口号

SOCKETn工作在TCP或UDP模式下有效，在其它模式下将忽略。它必须在OPEN命令之前设置。

例) Sn_PORTR = 5000(0x1388)

Sn_PORTR(0x0820A+0x40n/0x20A+0x40n)	
Sn_PORTR0(0x0820A+0x40n/0x20A+0x40n)	Sn_PORTR1(0x0820B+0x40n/0x20B+0x40n)
0x13	0x88

Sn_DHAR (SOCKETn目的硬件地址寄存器) [R/W] [0x0820C+0x40n/0x20C+0x40n] [FF.FF.FF.FF.FF.FF]

主机通过该寄存器设置SOCKETn的目的硬件地址，或通过某些过程获得SOCKETn的目的硬件地址。如果SOCKET0用于PPPoE模式， S0_DHAR就是已经得到的PPPoE服务器的硬件地址。

当在UDP或IPRAW模式使用SEND_MAC命令时，它设置SOCKETn的目的硬件地址。在TCP、UDP或IPRAW模式，通过CONNECT或SEND命令的ARP处理，Sn_DHAR获得了目的硬件地址。主机在成功运行CONNECT或SEND命令后，可以通过Sn_DHAR得到目的硬件地址。

当使用W5300的PPPoE处理时，PPPoE服务器的硬件地址不需要设置。然而，即使不使用W5300的PPPoE处理，而是自己通过MACRAW模式来实现，但为了发送和接收PPPoE数据包， PPPoE服务器硬件地址（PPPoE处理过程中获得）、PPPoE服务器IP地址和PPP对话的IP地址都需要设置，且MR(PPPoE)也需要设置为‘1’。

S0_DHAR在运行OPEN命令之前设置PPPoE的硬件地址。运行OPEN命令后，通过S0_DHAR设置的PPPOE服务器的硬件地址应用到PDHAR。即使运行了CLOSE命令，PPPoE的配置信息在内部一直有效。

例) Sn_DHAR = 00.08.DC.01.02.10

Sn_DHAR(0x0820C+0x40n/0x20C+0x040n)	
Sn_DHAR0(0x0820C+0x40n/0x20C+0x040n)	Sn_DHAR1(0x0820D+0x40n/0x20D+0x040n)
0x00	0x08
Sn_DHAR2(0x0820E+0x40n/0x20E+0x040n)	
Sn_DHAR2(0x0820E+0x40n/0x20E+0x040n)	Sn_DHAR3(0x0820F+0x40n/0x20F+0x040n)
0xDC	0x01
Sn_DHAR4(0x08210+0x40n/0x210+0x040n)	
Sn_DHAR4(0x08210+0x40n/0x210+0x040n)	Sn_DHAR5(0x08211+0x40n/0x211+0x040n)
0x02	0x10

Sn_DPORTR (SOCKETn目的端口号寄存器) [RO] [0x08212+0x40n/0x212+0x40n] [0x0000]

它用于设置SOCKETn的目的端口号，或在通信过程中获得SOCKETn的目的端口号。如果SOCKET0为PPPoE模式，S0_DPORTR设置为已经获得的PPP的会话ID。

只有在TCP、UDP或PPPoE模式下有效，在其它模式下无效。

在TCP客户端模式下运行CONNECT命令之前，需要将它设置为处于TCP服务器模式下的侦听端口的端口号。而在TCP服务器模式，当成功建立连接以后，它被设置为TCP客户端的端口号。

在UDP模式下运行SEND或SEND_MAC命令之前，Sn_DPORTR设置为UDP数据包发送的目的端口号。

在PPPoE模式下，S0_DPORTR设置为已经获得的PPP会话ID。在运行OPEN命令之后，PPP会话ID应用在PSIDR。

例) Sn_DPORTR = 5000(0x1388)

Sn_PORTR(0x08212+0x40n/0x212+0x40n)	
Sn_PORTR0(0x08212+0x40n/0x212+0x40n)	Sn_PORTR1(0x08213+0x40n/0x213+0x40n)
0x13	0x88

Sn_DIPR (SOCKETn目的IP地址寄存器) [R/W][0x08214+0x40n/0x214+0x40n] [00.00.00.00]

它设置为目的IP地址或在通信过程中被设置为目的IP地址。如果SOCKET0用于PPPoE模式，S0_DIPR设置为已经获得的PPPoE服务器的IP地址。只有在TCP、UDP或PPPoE模式下有效，在其它模式下无效。

在TCP客户端模式下，运行CONNECT命令之前，必须将它设置为TCP服务器的IP地址。而在TCP服务器模式，当成功建立连接以后，它被设置为TCP客户端的IP地址。

在UDP模式下运行SEND或SEND_MAC命令之前，Sn_DIPR设置为UDP数据包发送的目的IP地址。

在PPPoE模式下，S0_DIPR设置为已经获得的PPPoE服务器的IP地址。

例) Sn_DIPR = 192.168.0.11

Sn_DIPR(0x08214+0x40n/0x214+0x040n)	
Sn_DIPR0(0x08214+0x40n/0x214+0x040n)	Sn_DIPR1(0x08215+0x40n/0x215+0x040n)
192 (0xC0)	168 (0xA8)
Sn_DHAR2(0x08216+0x40n/0x216+0x040n)	
Sn_DIPR2(0x08216+0x40n/0x216+0x040n)	Sn_DIPR3(0x08217+0x40n/0x217+0x040n)
0 (0x00)	11 (0x0B)

Sn_MSSR (SOCKETn最大分片字节数寄存器) [R/W][0x08218+0x40n/0x218+0x40n] [0x0000]

它设置SOCKETn的MTU（最大传输单元），或指示已经设置的MTU。

如果主机不设置Sn_MSSR，它将使用默认的设置。

它只支持TCP或UDP模式。当使用PPPoE（MR（PPPoE）=1）时，TCP或UDP的MTU设置为PPPoE的MTU。

在IPRAW或MACRAW模式，内部不处理MTU，但使用默认的MTU。因此，当传输的数据长度超过默认的MTU时，主机应当将数据按照默认的MTU进行分片。

在TCP或UDP模式，如果传输的数据长度超过了MTU，W5300将数据自动按照MTU进行分片。

MTU在TCP模式下又称之为MSS。根据主机写入的值和对端的MSS，W5300在连接过程中选择一个较小的MSS值。

在UDP模式下，没有TCP的连接过程，因此只使用主机设置的值。当与不相同的MTU的对端通信时，W5300可以接收ICMP（分片的MTU）数据包。在这种情况下，IR（FMTU）设置为‘1’，主机可以通过FMTUR和UIPR分别获得分片的MTU和目的IP地址。在IF（FMTU）= 1时，不能与对端进行UDP通信。所以必须关闭SOCKET，设置FMTU为Sn_MSS，然后试着使用OPEN命令启动通信。

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

例) Sn_MSSR = 1460 (0x05B4)

Sn_MSSR(0x08218+0x40n/0x218+0x040n)	
Sn_MSSR0(0x08218+0x40n/0x218+0x040n)	Sn_MSSR1(0x08219+0x40n/0x219+0x040n)
0x05	0xB4

Sn_KPALVTR(SOCKETn保持激活时间寄存器[R/W][0x0821A+40n/0x21A+0x40n][0x00]

这是一个1字节的寄存器，设置SOCKETn的数据包保持激活（KA）的传输定时。只有在TCP模式下有效，在其它模式下无效。时间单位是5秒。

在Sn_SSR变换为SOCK_ESTABLISHED后，并经过最多一次数据包的发送和接收后，KA数据包就

可以传输。在Sn_KPALVTR>0时，W5300经过一段时间间隔后自动发送一个KA数据包，以检验TCP的连接（自动保持激活操作）。在Sn_KPALVTR=0时，不进行自动保持激活的操作，KA数据包可以由主机通过发送SEND_KEEP命令来实现（手动保持激活操作）。当Sn_KPALVTR>0时忽略手动激活操作。

例) 当'Sn_KPALVTR = 10'，KA数据包每隔50秒传输一次。

Sn_PROTOR(0x0821A+0x40n/0x21A+0x040n)	
Sn_KPALVTR(0x0821A+0x40n/0x21A+0x040n)	Sn_PROTOR (0x0821B+0x40n/0x21B+0x040n)
10 (0x0A)	Sn_PROTOR

Sn_PROTOR (SOCKETn协议号寄存器)[R/W][0x0821B+40n/0x21B+0x40n] [0x00]

它是一个1字节的寄存器，设置IP层数据包IP头中的协议号字段。

它只是在IPRAW模式有效，而在其它模式无效。Sn_PROTOR必须在运行OPEN命令之前设置。SOCKETn以IPRAW模式打开，发送和接收的数据由Sn_PROTOR协议定义。Sn_PROTOR设置的范围是0x00~0xFF，但W5300不支持TCP(0x06)和UDP(0x11)协议号。

协议号由IANA（Internet assigned numbers authority)）中定义。详细信息请参考在线文件：http://www.iana.org/assignments/protocol-numbers

例) Sn_PROTOR = 0x01 (ICMP)

Sn_PROTOR(0x0821A+0x40n/0x21A+0x040n)	
Sn_KPALVTR(0x0821A+0x40n/0x21A+0x040n)	Sn_PROTOR (0x0821B+0x40n/0x21B+0x040n)
Sn_KPALVTR	0x01

Sn_TOSR (SOCKETn的服务类型寄存器) [R/W] [0x0821C+40n/0x21C+40n] [0x00]

它设置IP层数据包IP头中的服务类型(TOS) 字段。它必须在OPEN命令之前设置。参考：http://www.iana.org/assignments/ip-parameters

例) Sn_TOSR = 0x00

Sn_TOSR(0x0821C+0x40n/0x21C+0x040n)	
Sn_TOSR0(0x0821C+0x40n/0x21C+0x040n)	Sn_TOSR1(0x0821D+0x40n/0x21D+0x040n)
Reserved	0x00

Sn_TTLR (SOCKETn的TTL寄存器) [R/W] [0x0821E+40n/0x21E+40n] [0x80]

它设置IP层数据包IP头中的生存期（TTL）字段。它必须在OPEN命令之前设置。参考：http://www.iana.org/assignments/ip-parameters

例) Sn_TTLR = 128 (0x80)

Sn_TTLR(0x0821E+0x40n/0x21E+0x040n)	
Sn_TTLR0(0x0821E+0x40n/0x21E+0x040n)	Sn_TTLR1(0x0821F+0x40n/0x21F+0x040n)
Reserved	0x80

Sn_TX_WRSR (SOCKETn发送数据字节长度寄存器) [R/W][0x08220+40n/0x220+40n] [0x00000000]

它设置通过Sn_TX_FIFOR寄存器写入到内部TX存储器的数据字节长度。

它必须在SEND或SEND_MAC命令之前进行设置，该值不能比TMSRn设置的内部TX存储器空间大。

在TCP或UDP模式，如果Sn_TX_WRSR > Sn_MSSR，W5300自动将数据按照Sn_MSSR进行分片。在其它模式，Sn_TX_WRSR不能比Sn_MSSR大。

例1) Sn_TX_WRSR = 64KB = 65536 = 0x00010000

Sn_TX_WRSR(0x08220+0x40n/0x220+0x040n)								
Sn_TX_WRSR0(0x08220+0x40n/0x220+0x040n)	Sn_TX_WRSR1(0x08221+0x40n/0x221+0x040n)							
Reserved	-	-	-	-	-	-	-	'1'
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)								
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)	Sn_TX_WRSR3(0x08223+0x40n/0x21D+0x040n)							
0x00	0x00							

例2) Sn_TX_WRSR = 2017 = 0x000007E1

Sn_TX_WRSR(0x08220+0x40n/0x220+0x040n)								
Sn_TX_WRSR0(0x08220+0x40n/0x220+0x040n)	Sn_TX_WRSR1(0x08221+0x40n/0x221+0x040n)							
Reserved	-	-	-	-	-	-	-	'0'
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)								
Sn_TX_WRSR2(0x08222+0x40n/0x222+0x040n)	Sn_TX_WRSR3(0x08223+0x40n/0x223+0x040n)							
0x07	0xE1							

Sn_TX_FSR (SOCKETn剩余存储空间寄存器) [R][0x08224+40n/0x224+40n] [0x00002000]

它指示SOCKETn内部TX存储器的剩余空间的字节数（传输数据的字节数）。当发送数据的字节数超过Sn_TX_FSR时，主机不能通过Sn_TX_FIFOR写入数据。因此，在发送数据之前，必须检验Sn_TX_FSR寄存器。而如果发送的字节数小于或等于Sn_TX_FSR，那么就可以向W5300写入数据，并使用SEND或SEND_MAC命令发送数据。

在TCP模式，如果对端检查发送的数据包（收到对端的DATA/ACK数据包），Sn_TX_FSR根据发送数据包的大小自动增加。在其它模式，当IR(SENDOK)为‘1’时，Sn_TX_FSR根据发送数据的多少自动增加。

例1) Sn_TX_FSR = 64KB = 65536 = 0x00010000

Sn_TX_FSR(0x08224+0x40n/0x224+0x040n)								
Sn_TX_FSR0(0x08224+0x40n/0x214+0x040n)					Sn_TX_FSR1(0x08225+0x40n/0x225+0x040n)			
Reserved	-	-	-	-	-	-	-	‘1’
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)								
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)					Sn_TX_FSR3(0x08227+0x40n/0x227+0x040n)			
0x00					0x00			

例2) Sn_TX_FSR = 33332 = 0x00008234

Sn_TX_FSR(0x08224+0x40n/0x224+0x040n)								
Sn_TX_FSR0(0x08224+0x40n/0x224+0x040n)					Sn_TX_FSR1(0x08225+0x40n/0x225+0x040n)			
Reserved	-	-	-	-	-	-	-	‘0’
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)								
Sn_TX_FSR2(0x08226+0x40n/0x226+0x040n)					Sn_TX_FSR3(0x08227+0x40n/0x227+0x040n)			
0x82					0x34			

Sn_RX_RSR (OCKETn接收数据的字节长度寄存器) [R][0x08228+40n/0x228+40n] [0x00000000]

它指示SOCKETn内部RX存储器接收数据的字节长度。

主机不能通过Sn_RX_FIFOR读取比Sn_RX_RSR多的数据。因此，检查Sn_RX_RSR之后，主机才可以通过Sn_RX_FIFOR读取Sn_RX_RSR字节长度的数据，或比Sn_RX_RSR少的数据，然后将数据保存到主机系统存储器中。然后，主机通过RECV命令通知W5300数据读取完成。

主机在读Sn_RX_FIFOR寄存器时，Sn_RX_RSR自动增加2字节。

当Sn_RX_RSR>0时，有一个或多个数据包在W5300的内部RX存储器中。而接收的数据必须按照数据包的单元进行处理。参考Sn_RX_FIFOR.

例1) Sn_RX_RSR = 64KB = 65536 = 0x00010000

Sn_RX_RSR(0x08228+0x40n/0x228+0x040n)								
Sn_RX_RSR0(0x08228+0x40n/0x21C+0x040n)	Sn_RX_RSR1(0x08229+0x40n/0x229+0x040n)							
Reserved	-	-	-	-	-	-	-	'1'
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)								
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)	Sn_RX_RSR3(0x0822B+0x40n/0x22B+0x040n)							
0x00	0x00							

例2) Sn_RX_RSR = 3800 = 0x00000ED8

Sn_RX_RSR(0x08228+0x40n/0x228+0x040n)								
Sn_RX_RSR0(0x08228+0x40n/0x21C+0x040n)	Sn_RX_RSR1(0x08229+0x40n/0x229+0x040n)							
Reserved	-	-	-	-	-	-	-	'0'
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)								
Sn_RX_RSR2(0x0822A+0x40n/0x22A+0x040n)	Sn_RX_RSR3(0x0822B+0x40n/0x22B+0x040n)							
0x0E	0xD8							

Sn_FRAGR (SOCKETn分片寄存器) [R/W] [0x0822C+40n/0x22C+40n] [0x40]

它设置IP层数据包IP头中的分片字段。W5300不支持IP层的数据包分片。即使配置了Sn_FRAGR，IP数据也不分片。不推荐使用该配置。该寄存器需要在运行OPEN命令之前进行配置。

例) Sn_FRAGR = 0x40 (不分片)

Sn_FRAGR(0x0822C+0x40n/0x22C+0x040n)	
Sn_FRAGR0(0x0822C+0x40n/0x22C+0x040n)	Sn_FRAGR1(0x0822D+0x40n/0x22D+0x040n)
Reserved	0x40

Sn_TX_FIFOR (SOCKETn的TX FIFO寄存器) [R/W] [0x0822E+40n/0x22E+40n] [0xUUUU]

它直接访问SOCKETn的内部TX存储器。

主机不能直接访问内部TX存储器，只能通过Sn_TX_FIFOR。如果MR（MT）=0，主机只能通过Sn_TX_FIFOR对内部TX存储器写操作。但如果MR（MT）=1，主机可以对内部TX存储器进行读/写操作。在检查W5300和主机之间的接口之后，必须将MR（MT）置0。（参考‘怎样测试内部TX/RX存储器’）

如果主机系统使用的是8位数据总线，Sn_TX_FIFOR0和Sn_TX_FIFOR1应该成对使用。当要写入1

字节到TX存储器时，将该字节写入Sn_TX_FIFOR0，而写入Sn_TX_FIFOR1的是一个哑字节数据。

Sn_TX_FIFOR必须访问2个字节。首先访问低地址寄存器Sn_TX_FIFOR0，然后访问高地址寄存器Sn_TX_FIFOR1。访问Sn_TX_FIFOR0后，不能立即访问W5300的其它寄存器，除非访问了Sn_TX_FIFOR1。

当主机通过Sn_TX_FIFOR写入任何数据时，数据将按顺序写入到内部TX存储器。Sn_TX_FIFOR0的数据和Sn_TX_FIFOR1的数据分别存储在内部TX存储器的低地址和高地址。通过SEND或SEND_MAC命令，内部TX存储器的数据将按照先低地址后高地址的顺序发送。

例1) Sn_TX_FIFOR = 0x1122

Sn_TX_FIFOR(0x0822E+0x40n/0x22E+0x040n)	
Sn_TX_FIFOR0(0x0822E+0x40n/0x22E+0x040n)	Sn_TX_FIFOR1(0x0822F+0x40n/0x22F+0x040n)
0x11	0x22

例2) When transmitting 5 Byte String Data “abcde” (abcde - 0x61 0x62 0x63 0x64 0x65)

16 Bit Data Bus Width (MR(DBW) = ‘1’)	8 Bit Data Bus Width (MR(DBW) = ‘0’)
Sn_TX_FIFOR = 0x6162	Sn_TX_FIFOR0 = 0x61
Sn_TX_FIFOR = 0x6364	Sn_TX_FIFOR1 = 0x62
Sn_TX_FIFOR = 0x6500	Sn_TX_FIFOR0 = 0x63
Sn_TX_WRSR0 = 0x0000	Sn_TX_FIFOR1 = 0x64
Sn_TX_WRSR1 = 0x0005	Sn_TX_FIFOR0 = 0x65
Sn_CR = 0x0020 (SEND command)	Sn_TX_FIFOR1 = 0x00
	Sn_TX_WRSR0 = 0x00
	Sn_TX_WRSR1 = 0x00
	Sn_TX_WRSR2 = 0x00
	Sn_TX_WRSR2 = 0x05
	Sn_CR1 = 0x20 (SEND command)

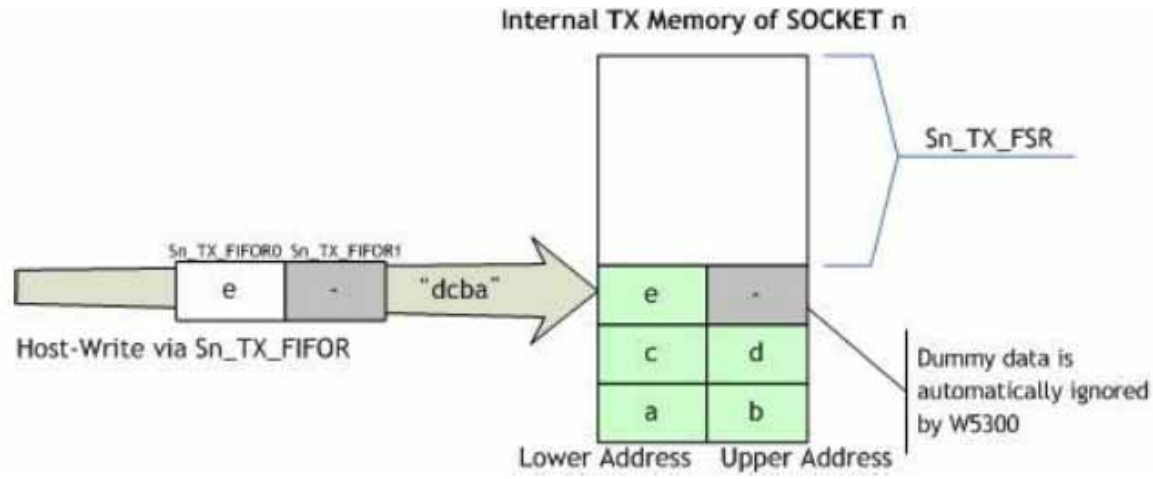


图6. 访问内部TX存存器

Sn_RX_FIFOR (SOCKETn RX FIFO寄存器) [R/W] [0x08230+40n/0x230+40n] [0xUUUU]

它直接范围SOCKETn的内部RX存储器。

主机不能直接访问内部RX存储器，只能通过Sn_RX_FIFOR。如果MR（MT）=0，主机只能通过Sn_RX_FIFOR对内部RX存储器进行读操作。但如果MR（MT）=1，主机可以进行读/写操作。在检查W5300和主机之间的接口之后，必须将MR（MT）置0。（参考‘怎样测试内部TX/RX存储器’）。

如果主机系统是8位数据总线，与Sn_TX_FIFOR一样，Sn_RX_FIFOR0和Sn_RX_FIFOR1需要成对访问。在访问了Sn_RX_FIFOR0和Sn_RX_FIFOR1之后，是不能立即访问Sn_TX_FIFOR0和Sn_TX_FIFOR1，这样将造成不正确的读取数据。为了避免这种情况的发生，在读取了Sn_TX_FIFOR0和Sn_TX_FIFOR1后，主机要读取其它任何寄存器，比如Sn_MR寄存器，然后再读取Sn_RX_FIFOR寄存器。

当主机通过Sn_RX_FIFOR从内部RX存存器读取2字节的接收的数据包时，低字节和高字节分别为Sn_RX_FIFOR0和Sn_RX_FIFOR1。主机读取内部RX存储器的数据完成后才可以运行RECV命令。

根据Sn_MR(P3:P0)的设置，在数据包的前面追加有PACKET-INFO，PACKET-INFO包含接收数据包的信息，如数据包的大小。主机必须首先处理PACKET-INFO，然后再处理数据。如果收到的数据长度字节数为奇数，那么将添加一个哑字节。主机必须首先读取这个哑字节并丢弃它。从PACKET-INFO的字节长度信息可以判断数据包的最后一个字节是否为哑字节。

通过Sn_RX_FIFOR寄存器，主机顺序处理内部RX存储器中的PACKET-INFO信息和DATA数据。

在TCP和MACRAW模式，PACKET-INFO信息固定为2字节，在UDP模式为8个字节，在IPRAW模式为6个字节。PACKET-INFO的详细信息请参考“第五章. 功能描述部分”。

例1) Sn_RX_FIFOR = 0x3344

Sn_RX_FIFOR(0x08230+0x40n/0x230+0x040n)	
Sn_RX_FIFOR0(0x08230+0x40n/0x230+0x040n)	Sn_RX_FIFOR1(0x08231+0x40n/0x231+0x040n)
0x33	0x44

例2) 在TCP模式，接收了5个字节的字符串"abcde"，并保存在"str"变量中。

16 Bit Data Bus Width (MR(DBW) = '1')	8 Bit Data Bus Width (MR(DBW) = '0')
INT16 pack_size, idx,temp	**INT16** pack_size, idx,temp
INT8 str[5]	**INT8** str[5], dummy
pack_size = Sn_RX_FIFOR	pack_size = Sn_RX_FIFOR0
idx = 0	pack_size = (pack_size << 8)
LOOP pack_size/2	pack_size = pack_size + Sn_RX_FIFOR1
temp = Sn_RX_FIFOR	idx = 0
str[idx] = (INT8)(temp >> 8)	**LOOP** pack_size/2
idx = idx + 1	str[idx] = Sn_RX_FIFOR0
str[idx] = (INT8)(temp & 0x00FF)	idx = idx + 1
idx = idx + 1	str[idx] = Sn_RX_FIFOR1
END LOOP	idx = idx + 1
IF pack_size is odd ? **THEN**	**END LOOP**
temp = Sn_RX_FIFOR	**IF** pack_size is odd ? **THEN**
str[idx] = (INT8)(temp >> 8)	str[idx] = Sn_RX_FIFOR0
END IF	dummy = Sn_RX_FIFOR1
Sn_CR = 0x0040 (RECV command)	**END IF**
	Sn_CR1 = 0x40 (RECV command)

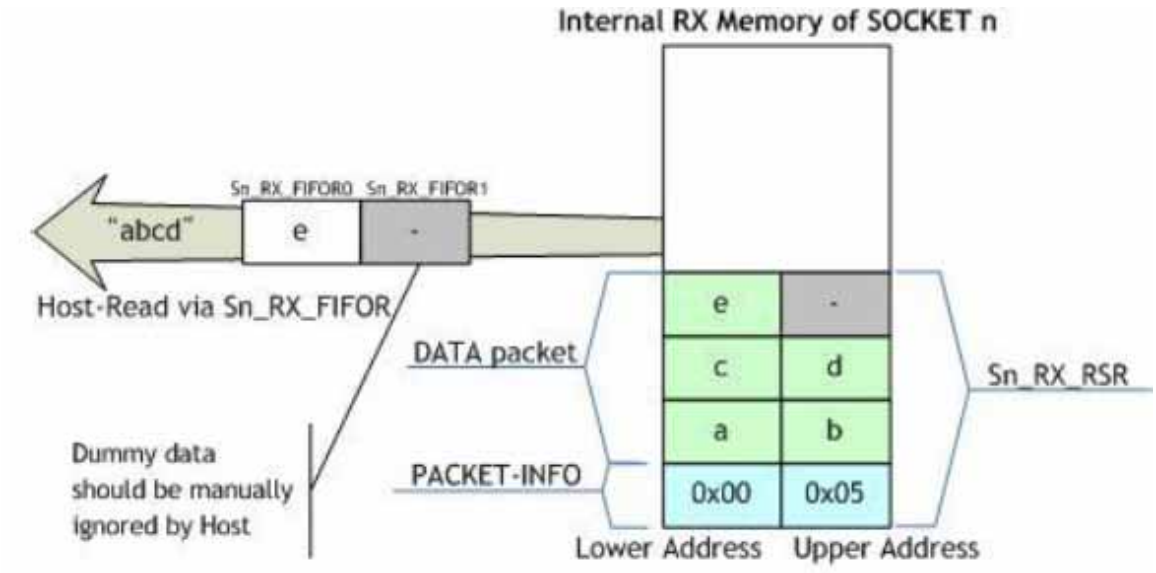


图7 访问内部RX存存器

5. 功能描述

W5300可以通过寄存器的设置使Internet的连接变得非常简单。在本章将学习怎样初始化W5300，通过学习一些代码，怎样根据协议类型（TCP、UDP、IPRAW和MACRAW）实现网络通信。

5.1 初始化

W5300的初始化分三个步骤：主机接口设置，网络信息设置和内部TX/RX存储器的分配。

第1步：设置主机接口

1. 设置数据总线宽度，主机接口模式和时序（参考MR寄存器）
2. 设置主机中断（参考IMR）

第2步：设置网络信息

1. 设置数据通信的基本网络信息（参考：SHAR、GAS、SBUR和SIPR）
2. 设置重复发送的时间间隔和重复发送的次数，用于数据包发送失败时的重复发送（参考RTR和RCR）

本机硬件地址通过SHAR设置，这是一个以太网设备的唯一的硬件地址（以太网的MAC地址），应用于以太网的MAC层

由IEEE分配和管理MAC地址。生产厂商需要从IEEE申请网络设备的MAC地址。参考 http://www.ieee.org/, http://standards.ieee.org/regauth/oui/index.shtml

第3步：分配SOCKETn的内部TX/RX存储器空间

1. 定义内部TX/RX存储器大小（参考MYTPER）
2. 定义SOCKETn的TX/RX存储器大小（参考TMR和RMSR）

W5300内部包含16个8K字节的存储单元。这些存储单元依次映射在128K字节的存储器空间。128K存储器分为发送存储器（TX）和接收存储器（RX）。内部TX和RX存储器以8K字节为单元分布在128K字节空间。内部TX/RX存储器可以在0~64K字节空间以1K字节为单元从新分配给每个SOCKET。

下面的图显示72K字节分配给内部TX存储器，56K字节分配给内部RX存储器。内部TX存储器在72K字节范围内，为SOCKET0到SOCKET7分别再分配4、16、1、20、0、7、12、12K字节。内部RX存储器在56K字节范围内为每个SOCKET再分配17、3、5、16、3、4、4K字节。SOCKET4不能发送数据，因为没有给它分配TX存储器。

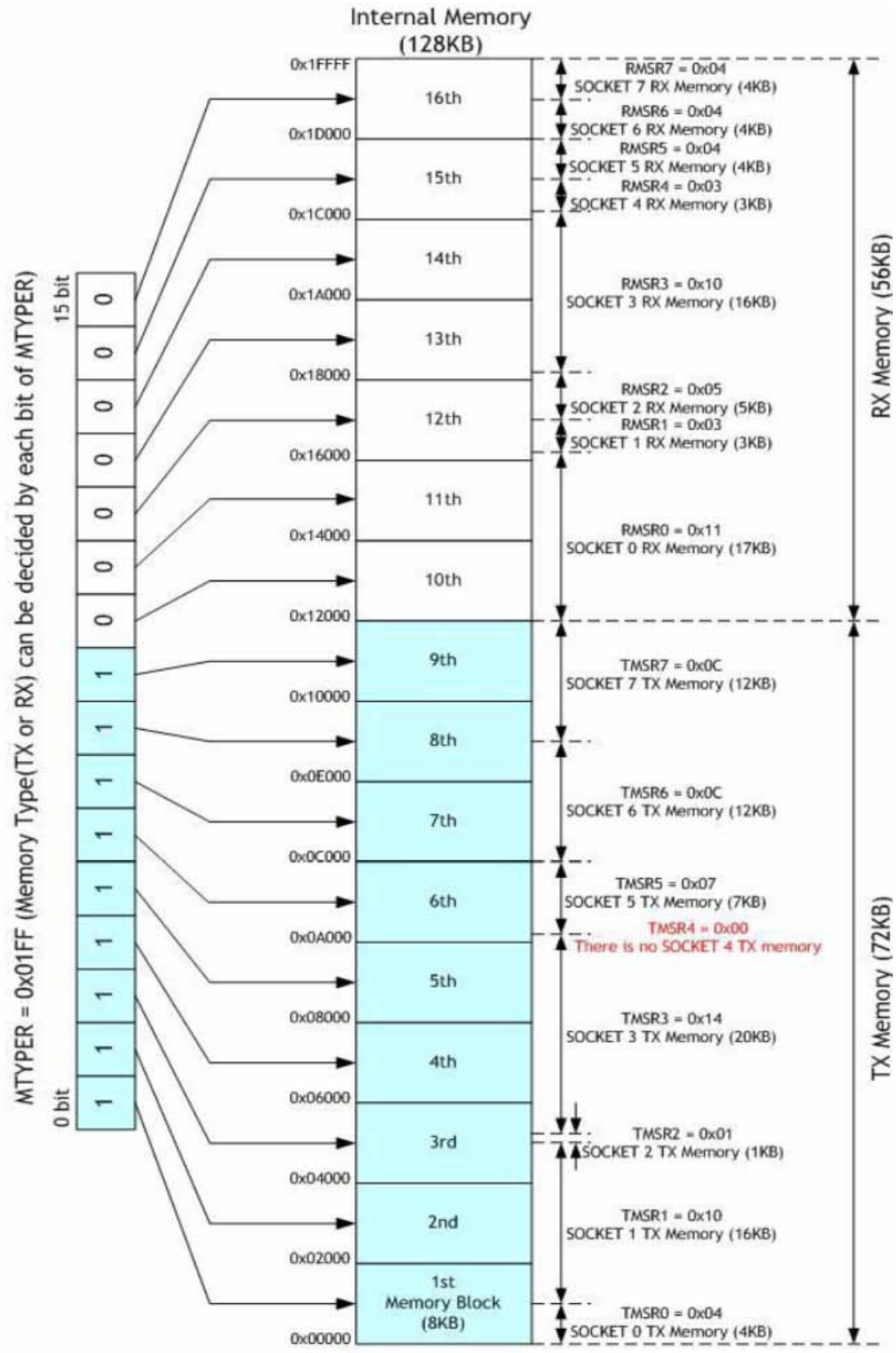


图 8 SOCKETn的内部TX/RX存储器分配

当第3步初始化设置完成后，就可以使用W5300通过以太网传输数据了。这时，W5300可以响应Ping请求（自动Ping响应）。

5.2 数据通信

完成初始化设置以后，W5300可以以TCP、UDP、IPRAW或MACRAW的方式打开SOCKET发送或接收数据。W5300支持8个SOCKET同时且独立地工作。在本章将描述每种通信方式的工作方法。

5.2.1 TCP

TCP是一种连接通信的协议，在TCP模式，首先要根据IP地址和端口号与对端建立SOCKET连接。通过连接的SOCKET发送和接收数据。

建立SOCKET的连接有“ TCP服务器 ”和“ TCP客户端 ”之分。区分它们的方法是谁首先发送连接请求（SYS数据包）。“ TCP服务器 ”等待对端的连接请求，当收到连接请求时建立SOCKET连接（被动打开）。“ TCP客户端 ”主动发出连接请求，与对端建立连接（主动打开）

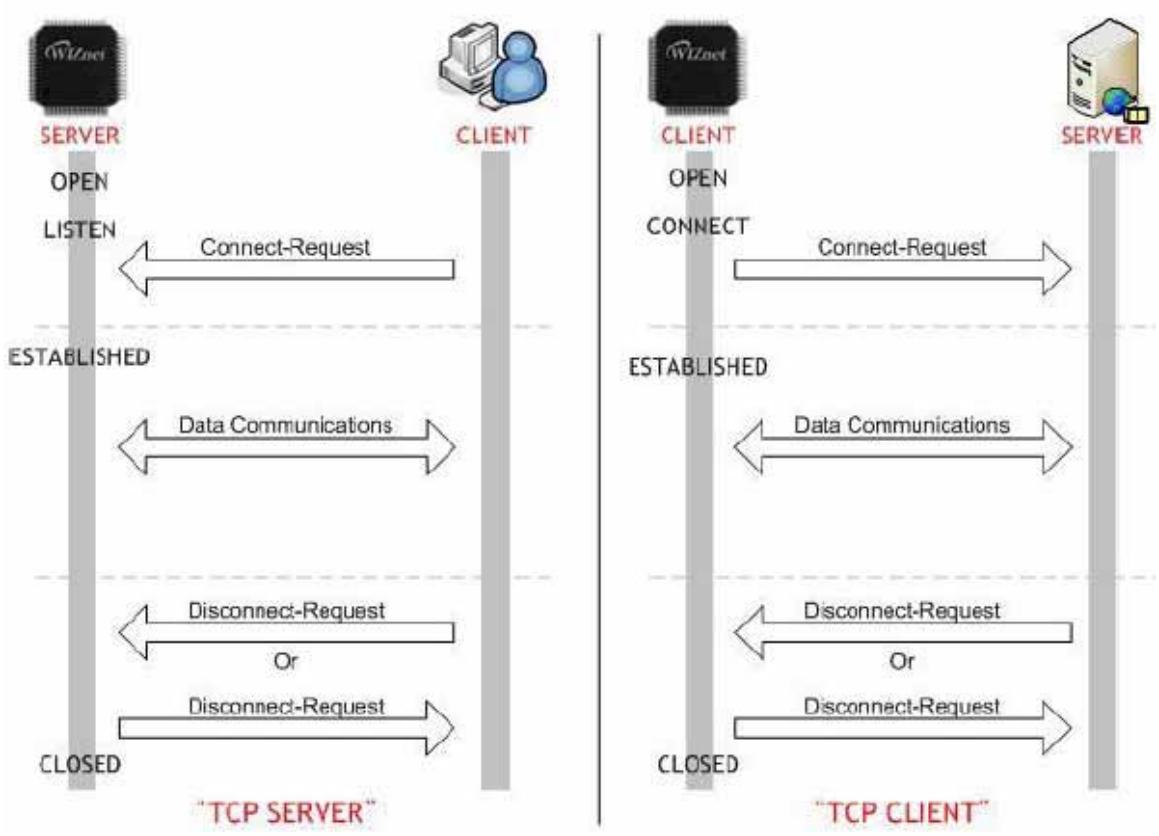


图9 TCP服务器和TCP客户端

5.2.1.1 TCP SERVER TCP服务器

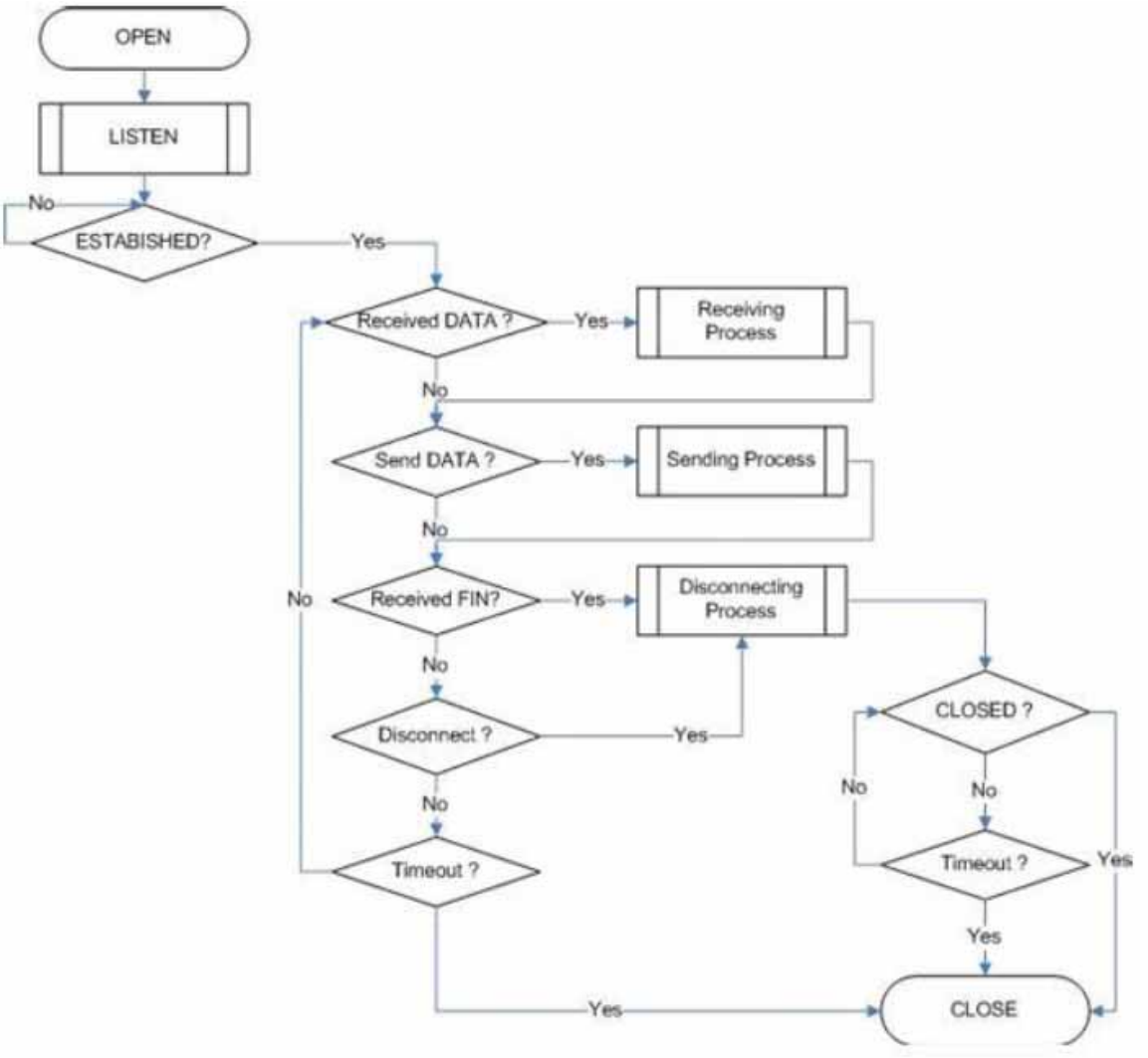


图10 TCP服务器操作流程

SOCKET初始化

为了实现TCP通信，需要对SOCKET进行初始化设置并打开SOCKET。为了打开SOCKET，选择其中的一个SOCKET（被选择的SOCKET称之为SOCKETn），通过Sn_MR(P3:P0)和Sn_PORTR分别设置通信协议和本机端口号（在TCP服务器模式，称之为侦听端口号），然后执行OPEN命令。执行完OPEN命令后，如果Sn_SSR改变为SOCK_INIT，则SOCKET的初始化设置完成。

在TCP服务器和TCP客户端模式，SOCKET初始化的过程都是相同的。下面的方法是将SOCKET初始化为TCP模式。

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

```
    Sn_PORTR = source_port;        /* sets source port number */

    Sn_CR = OPEN;                  /* sets OPEN command */


    /* wait until Sn_SSR is changed to SOCK_INIT */

    if (Sn_SSR != SOCK_INIT) Sn_CR = CLOSE; goto START;

}
```

如果接收到对端的数据字节数为偶数，Sn_MR(ALIGN)置‘1’。当Sn_MR(ALIGN)=1时，W5300不需要增加TCP模式的PACKET-INFO，而只将数据保存在SOCKETn的内部RX存储器。这样将减少主机对SOCKET_INFO的额外处理，提高系统性能。(在前面的代码中，Sn_MR=0x0101可能会被Sn_MR=0x0001取代)

侦听

运行LISTEN命令将W5300设置为TCP服务器模式

```
{
    /* listen SOCKET */

    Sn_CR = LISTEN;

    /* wait until Sn_SSR is changed to SOCK_LISTEN */

    If (Sn_SSR != SOCK_LISTEN) Sn_CR = CLOSE; goto START;

}
```

建立连接？

当Sn_SSR改变为SOCK_LISTEN状态时，如果收到SYN数据包，那么Sn_SSR将改变为SOCK_SYNRECV。发送了SYN/ACK数据包后，SOCKETn就建立了连接。SOCKETn建立连接以后就可以进行数据通信。有两种方法可以检验是否建立SOCKETn的连接。

第1种方法

```
{
    if (Sn_IR(CON) == ‘1’) Sn_IR(CON) = ‘1’; goto ESTABLISHED stage;

    /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR

    Sn_IMR and Sn_IR. */

}
```

第2种方法

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

建立连接：接收到数据？

检查是否接收到对端发送来的数据。

第1种方法

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

第2种方法

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

第1种方法，当SOCKETn接收到数据包时，Sn_IR(RECV)将置‘1’。这时如果主机还没有处理上次接收数据包的Sn_IR(RECV)，而W5300又收到下一次的数据包，主机保持前一次的Sn_IR(RECV)，不能识别下一次数据包的Sn_IR(RECV)，因此，如果主机没有能力处理所有数据包的Sn_IR(RECV)，建议不采用这种方法。

建立连接：接收数据处理

它处理内部RX存储器的TCP数据。接收到的TCP数据格式如下：

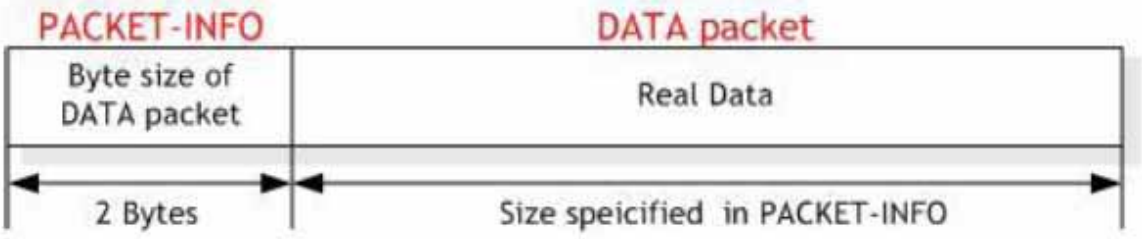


图11 接收的TCP数据格式

当Sn_MR(ALIGN)=0时，数据包中包含PACKET-INFO和数据的数据。在Sn_ME(ALIGN)=1时，TCP数据只有数据包而没有PACKET-INFO。

在TCP模式，如果对端发送的数据的大小超过SOCKETn的RX存储器的剩余空间，那么W5300将不能接收数据，这时的W5300将继续保持连接，并等待RX存储器的剩余空间大于数据长度。

```
{
    /* first, check Sn_MR(ALIGN) */
    if (Sn_MR(ALIGN) == '0')
    {
        pack_size = Sn_RX_FIFOR; /* extract size of DATA packet from internal RX memory */
    }
    else
    {
        pack_size = Sn_RX_RSR; /* check the total received data size */
    }
    /* calculate the read count of Sn_RX_FIFOR */
    if (pack_size is odd ?) read_cnt = (pack_size + 1) / 2;
    read_cnt = pack_size / 2;
    /* extract DATA packet from internal RX memory */
    for( i = 0; i < read_cnt; i++)
    {
        data_buf[i] = Sn_RX_FIFOR; /* data_buf is array of 16bit */
    }
    /* set RECV command */
    Sn_CR = RECV;
}
```

注意：当SOCKETn只用于接收数据而不发送数据时，主机不能快速接收处理数据将导致使内部RX存储器溢出。

在这种情况下，即使W5300的窗口尺寸（接收数据的最大尺寸）不为0，对端误以为窗口尺寸为0而不

再发送数据，等待窗口尺寸增加。这样就会降低W5300接收数据的性能。为了解决这个问题，主机首先处理内部RX存储器接收的数据，同时通知对端，W5300的窗口尺寸增加了已经接收数据的大小。针对上面代码，在RECV命令之后增加以下的代码。

```
/* set RECV command */

Sn_CR = RECV;

/* Add the code that notifies the update of window size to the peer */

/* check the received data process to finish or not */

if(Sn_RX_RSR == 0) /* send the window-update packet when the window size is full */

{ /* Sn_RX_RSR can be compared with another value instead of „0", according to the host performance of receiving data */

    Sn_TX_WRSR = 0x00000001;          /* set Dummy Data size to Sn_TX_WRSR */

    Sn_CR = SEND;                     /* set SEND command */

    while(Sn_CR != 0x00);             /* check SEND command completion */

    while(Sn_IR(SENDOK) == 0);        /* wait for SEND OK */

    Sn_IR(SENDOK) =1;             /* Clear SENDOK bit */

}
```

建立连接：发送数据？/发送处理

将数据通过Sn_TX_FIFOR写入到内部TX存储器后，W5300将试着把数据发送到对端。发送数据的大小不能比分配给该SOCKETn的内部TX存储器空间大。如果发送数据的尺寸比MSS大，W5300将自动根据MSS分片，然后再发送。

为了下一次数据的发送，主机必须检查上次SEND命令是否执行完毕。如果上一次的SEND命令还没有执行完而又开始下一次的SEND命令，将可能产生各种各样的错误。数据越大，执行SEND命令所需要的时间就会越长。所以要想提高发送效率，适当将数据分为合适的大小发送。

```
{

    /* first, get the free TX memory size */

    FREESIZE:

    get_free_size = Sn_TX_FSR;

    if (Sn_SSR != SOCK_ESTABLISHED && Sn_SSR != SOCK_CLOSE_WAIT) goto CLOSED state;
```

```
    if (get_free_size < send_size) goto FREESIZE;

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;
    /* copy data to internal TX memory */
    for (i = 0; i < write_cnt; i++)
    {
        Sn_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
    }

    /* check previous SEND command completion */
    if (is first send ?) ; /* skip check Sn_IR(SENDOK) */
    else
    {
        while(Sn_IR(SENDOK)=='0')
        {
           if(Sn_SSR == SOCK_CLOSED) goto CLOSED state; /* check connection establishment */
        }
        Sn_IR(SENDOK) = '1'; /* clear previous interrupt of SEND completion */
    }

    /* sets transmission data size to Sn_TX_WRSR */
    Sn_TX_WRSR = send_size;
    /* set SEND command */
    Sn_CR = SEND;
}
```

建立连接：接收到FIN

它检查是否接收到断开连接的请求（FIN数据包）。检查如下：

第1种方法

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

第2种方法

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

建立连接：断开连接？/断开连接处理

如果不再需要进行数据通信，或收到FIN数据包，那么SOCKET的连接应该断开。

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

建立连接：关闭端口？

它检查SOCKETn是否通过DISCON或CLOSE命令断开连接或关闭端口。

第1种方法

```
{
    if (Sn_IR(DISCON) == ‘1’) goto CLOSED stage;
```

```
    /* In this case, if the interrupt of SOCKETn is activated, interrupt occurs. Refer to IR, IMR
    Sn_IMR and Sn_IR. */
}
```

第2种方法

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

建立连接：超时

超时可能发生在TCP数据包传输过程中，如连接请求（SYN数据包）或其响应数据包（SYN/ACK数据包）、数据（DATA数据包）或其响应数据包（DATA/ACK数据包）、断开连接请求（FIN数据包）或其响应数据包（FIN/ACK数据包）等等。如果以上的数据包在RTR和RCR设定的时间内没有发送出去，那么将产生TCP超时，且Sn_SSR将改变为SOCK_CLOSED状态。

TCP超时检查如下：

第1种方法：

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

第2种方法：

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

SOCKET关闭

SOCKETn经过断开连接处理或由于超时而断开连接，它用于关闭断开连接的SOCKETn。主机也可以不进行断开连接处理而直接关闭SOCKETn。

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

5.2.1.2 TCP客户端

除了连接状态以外，其它都与TCP服务器完全相同。参考5.2.1.1 TCP服务器

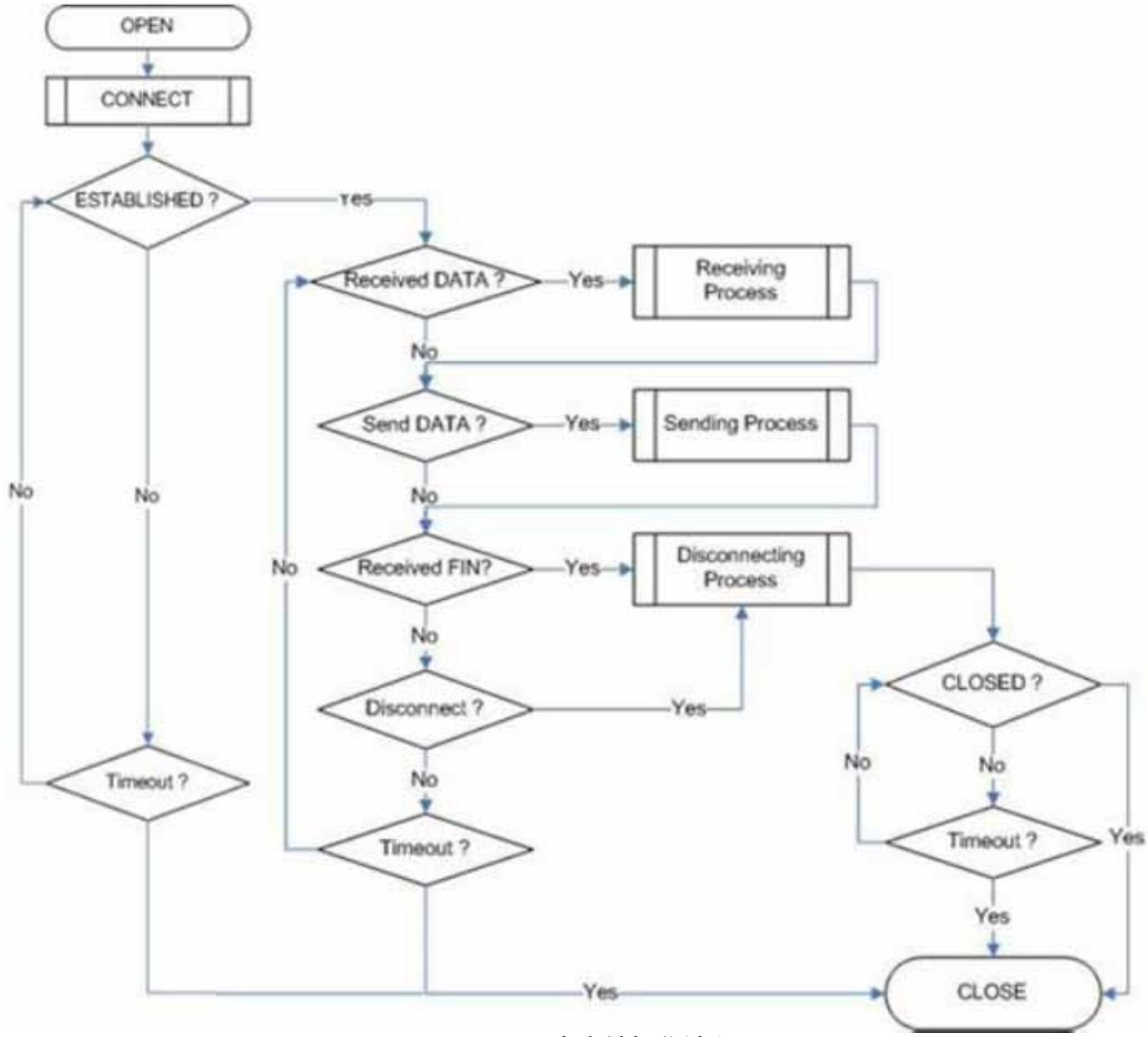


图12　TCP客户端操作流程

连接

它发送连接请求（SYN数据包）到对端。在与对端SOCKET建立连接的过程中可能会出现ARP超时，或TCP连接超时。

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

5.2.2 UDP

UDP是一种无连接的通信协议。UDP发送和接收数据不需要像TCP那样建立SOCKET连接。TCP保证数据可靠传输，但UDP没有这种保证。UDP是一种报文通信协议。由于UDP不需要建立连接端口，因此它可以允许同多个已知的IP地址和端口号的对端进行通信。这种报文通信可以通过一个SOCKET与多个对端进行通信，但可能出现的问题是丢失数据，或从不希望的对端接收到数据。为了防止这种问题的产生，主机本身应当从新处理丢失的数据，或丢弃不希望对端发送来的数据。UDP支持单播、广播和多播通信。通信的流程如下所示：

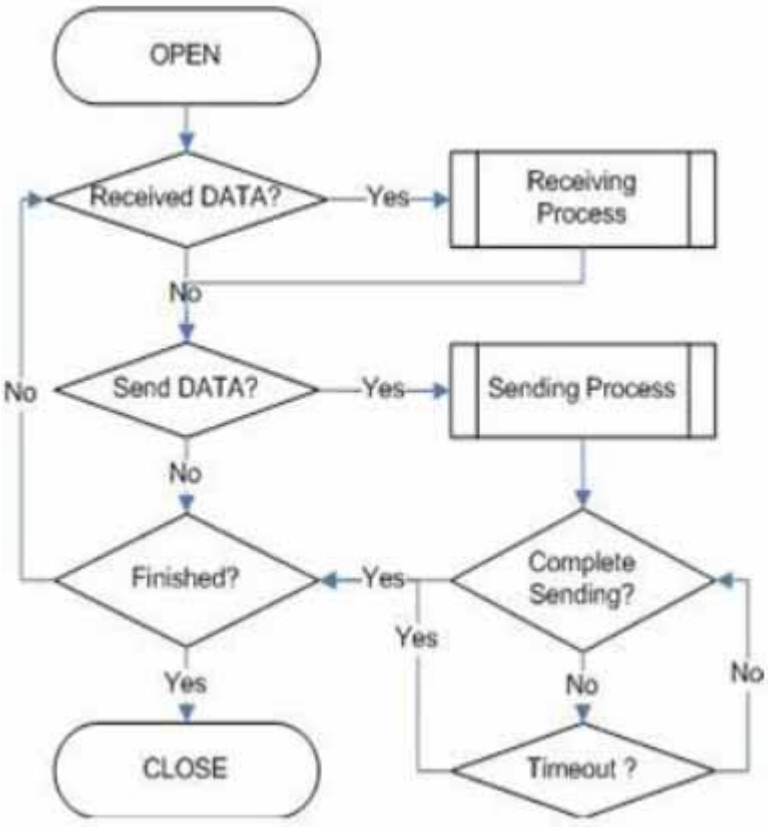


Fig 13. UDP操作流程

5.2.2.1 单播和广播

单播是UDP通信中使用最多的一种方法，它一次将数据发送到一个对端。广播是采用广播地址（255.255.255.255）将数据一次发送到所有接收的对端。

例如，有三个对端A，B，C，单播是将数据发送到A或B或C。这时通过ARP获取A、B、C的硬件地址

时可能产生ARP超时。它不可能将数据发送到ARP超时的对端。广播是使用广播地址（255.255.255.255）将数据同时发送到A、B和C。与单播不同，广播不需要通过ARP来获得A、B和C的硬件地址，也不会发生ARP超时。

SOCKET初始化

为了UDP数据通信，必须对SOCKET初始化，打开一个SOCKET。

为了打开SOCKET，选择其中的一个SOCKET（被选择的SOCKET称之为SOCKETn），通过Sn_MR(P3:P0)和Sn_PORTR分别设置协议类型和本机端口号。然后执行OPEN命令。执行OPEN命令后，如果SOCKET的状态改变为SOCK_UDP，则完成SOCKET的初始化。

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

接收数据？

它检查是否从对端接收到UDP数据。与TCP通信的检查方法相同。不建议采用第1种方法。详细信息请参考“ 5.2.1.1 TCP服务器 ”。

第1种方法

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

第2种方法

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

接收数据处理

它处理内部RX存储器中接收到的UDP的数据。接收到的UDP数据格式如下：

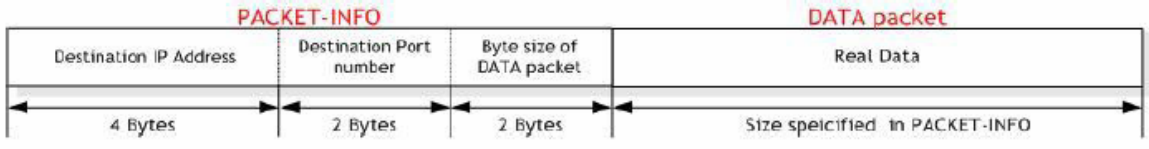


图14 接收的UDP数据格式

UDP数据包含有8个字节的PACKET-INFO，PACKET-INFO包含发送端的信息（IP地址、端口号等等）和数据包字节长度。UDP可以接收多个发送者发送的UDP数据。主机可以从PACKET-INFO中得到发送者的IP地址和端口号。W5300也可以收到发送者使用广播地址“ 255.255.255.255 ”发送的数据。主机必须通过PACKET-INFO信息判断哪些是不需要的数据包。

如果数据长度大于SOCKETn的RX存储器剩余空间，那么W5300将不能接收数据。也无法接收分片数据。

```
{
    /* process PACKET-INFO read from internal RX memory */
    temp = Sn_RX_FIFOR; /* extract destination IP address from internal RX memory */
    dest_ip[0] = ((temp & 0xFF00) >> 8);
    dest_ip[1] = (temp & 0x00FF);
    temp = Sn_RX_FIFOR;
    dest_ip[2] = ((temp & 0xFF00) >> 8);
    dest_ip[3] = (temp & 0x00FF);
    dest_port = Sn_RX_FIFOR; /* extract destination port number from internal RX memory */
    pack_size = Sn_RX_FIFOR; /* extract length of DAT packet from internal RX memory */

    /* calculate the read count of Sn_RX_FIFOR */
    if (pack_size is odd ?) read_cnt = (pack_size + 1) / 2;
    read_cnt = pack_size / 2;
    for ( i = 0 ; i < read_cnt ; i++ )
    {
```

```
        data_buf[i] = Sn_RX_FIFOR; /* data_buf is array of 16bit */
    }
    /* set RECV command */
    Sn_CR = RECV;
}
```

发送数据？/发送处理

设置对端的IP地址和端口号，将要发送的数据通过Sn_TX_FIFOR写入到内部TX存储器，然后试着将数据发送到对端。

发送数据的大小不能超过SOCKETn的TX存储器的大小。如果数据字节长度比MTU大，W5300将根据MTU自动分片，然后将分片的数据传输到对端。

广播发送时，Sn_DIPR设置为“ 255.255.255.255 ”

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

    /* Set the destination information */
    Sn_DIPR0 = dest_ip[0]; //or 255; /* Set the 4 bytes destination IP address to Sn_DIPR */
    Sn_DIPR1 = dest_ip[1]; //or 255;
    Sn_DIPR2 = dest_ip[2]; //or 255;
    Sn_DIPR3 = dest_ip[3]; //or 255;
    Sn_DPORTR = dest_port; /* Set the 2 bytes destination port number to Sn_DPORTR */

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
```

```
else write_cnt = send_size / 2;

/* copy data to internal TX memory */

for (i = 0; i < write_cnt; i++)

{

    Sn_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */

}

/* sets transmission data size to Sn_TX_WRSR */

Sn_TX_WRSR = send_size;

/* set SEND command */

Sn_CR = SEND;

}
```

完成发送？/超时

为了发送下一包数据，需要先确认上一包数据已经发送完毕。数据包越大，执行SEND命令所需要的时间就越长。可以将数据包适当分小使数据发送效率更高。

传输UDP数据时，可能会产生ARP超时。这种情况下UDP数据发送失败。

```
{

/* check SEND command completion */

while(Sn_IR(SENDOK)==‘0’) /* wait interrupt of SEND completion */

{

/* check ARPTO */

if (Sn_IR(TIMEOUT)==‘1’) Sn_IR(TIMEOUT)=‘1’; goto Next stage;

}

Sn_IR(SENDOK) = ‘1’; /* clear previous interrupt of SEND completion */
}
```

结束任务/关闭SOCKET

如果不再需要通信，可以关闭SOCKETn

```
{
```

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

5.2.2.2 Multicast多播

广播是与无法确认的多个对端进行通信的一种方法，但多播是与可以确定的多个对端进行通信，这些确定的对端是一组已经注册的多播组。

例如，A，B和C是已经注册的多播分组。如果A传输数据到多播分组，B和C可以收到数据。为了多播通信，使用IGMP协议注册为多播分组中的一格成员。所有的多播分组都是由分组硬件地址、分组IP地址和分组端口号进行区分的。

分组硬件和分组IP地址可以使用已经分配的地址，但分组端口号可以任意使用。

对分组硬件地址而言，可选择的范围在“ 01:00:5E:00:00:00 ”到“ 01:00:5E:FF:FF:FF ”之间。对分组IP地址而言，它是在D类IP地址范围（“ 224.0.0.0 ”~“ 239.255.255.255 ”）。这时，分组硬件地址（6字节）的低23位和IP地址（4字节）应该相同。例如，如果分组IP地址设置为“ 224.1.1.11 ”，那么分组硬件地址就应该为“ 01:00:5E:01:01:0B ”。参考“ RFC1112 ”（http://www.ietf.org/rfc.html） 。

在W5300中，自动处理需要注册多播组的IGMP。当以多播模式打开SOCKETn时，自动发送IGMP的“ Join ”信息。当关闭SOCKET时，发送“ Leave ”信息。打开SOCKET后，自动定时间隔发送“ Report ”信息。

W5300支持IGMP的版本1和2。如果需要更高的版本，主机可以在SOCKET的IPRAW模式下手动处理IGMP协议。

SOCKET初始化

为了实现多播通信，选择其中一个SOCKET（8个中被选中的端口称之为SOCKETn）。然后设置Sn_DHAR为多播分组硬件地址，Sn_DIPR为多播分组IP地址。Sn_PORTR和Sn_DPORTR设置为多播分组端口号。设置Sn_MR（P3:P0）为UDP且Sn_MR（MULTI）为“ 1 ”，然后运行OPEN命令。运行OPEN命令后，当SOCKET的状态改变为SOCK_UDP时，SOCKET初始化完成。

```
{
    START:
    /* set Multicast-Group information */
    Sn_DHAR0 = 0x01; /* set Multicast-Group H/W address(01:00:5e:01:01:0b) */
```

```
    Sn_DHAR1 = 0x00;
    Sn_DHAR2 = 0x5E;
    Sn_DHAR3 = 0x01;
    Sn_DHAR4 = 0x01;
    Sn_DHAR5 = 0x0B;

    Sn_DIPR0 = 211; /* set Multicast-Group IP address(211.1.1.11) */
    Sn_DIPR1 = 1;
    Sn_DIPR2 = 1;
    Sn_DIRP3 = 11;

    Sn_DPORTR = 0x0BB8; /* set Multicast-Group Port number(3000) */
    Sn_PORTR = 0x0BB8; /* set Source Port number(3000) */
    Sn_MR = 0x0002 | 0x0080; /* set UDP mode & Multicast on SOCKETn Mode Register */
    Sn_CR = OPEN; /* set OPEN command */

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

接收数据？

接收数据处理

参考“ 5.2.2.1 单播和广播 ”。

发送数据？/发送处理

因为多播分组的信息已经在SOCKET初始化过程中设置了，不需要再设置目的IP地址和目的端口号，这一点与单播方式不同。因此只需要将数据写入到内部TX存储器，然后运行SEND命令。

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

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;

    /* copy data to internal TX memory */
    for (i = 0; i < write_cnt; i++)
    {
        Sn_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
    }

    /* sets transmission data size to Sn_TX_WRSR */
    Sn_TX_WRSR = send_size;
    /* set SEND command */
    Sn_CR = SEND;
}
```

完成发送及超时

由于是与以前设定的多播分组通信，因此不需要进行ARP处理，因此也不会产生ARP超时。

```
{
    /* check SEND command completion */
```

```
    while(Sn_IR(SENDOK)==‘0’); /* wait interrupt of SEND completion */

    Sn_IR(SENDOK) = ‘1’; /* clear interrupt of SEND completion */

  }
```

完成发送？/关闭SOCKET

参考“ 5.2.2.1 单播和广播 ”。

5.2.3 IPRAW

IPRAW是在比TCP和UDP更低一层的IP层进行通信。IPRAW支持IP层的协议，如ICMP（0x01）或IGMP（0x02），可以通过协议号定义。

ICMP的PING命令或IGMP的版本1/版本2已经内部硬件逻辑实现。然而主机仍然可以通过将SOCKETn以IPRAW模式打开手动实现。

在使用SOCKET的IPRAW模式时，协议是通过IP字段的协议号定义的，协议号由IANA定义（参考http://www.iana.org/assignments/protocolnumbers）。协议号必须在打开SOCKET之前设置好。

W5300不支持TCP（0x06）或UDP（0x11）协议号。IPRAW模式通信的SOCKET只允许Sn_PROTOR设置协议号。例如，如果SOCKET的Sn_PROTOR设置为ICMP，就不能接收非ICMP协议的数据。

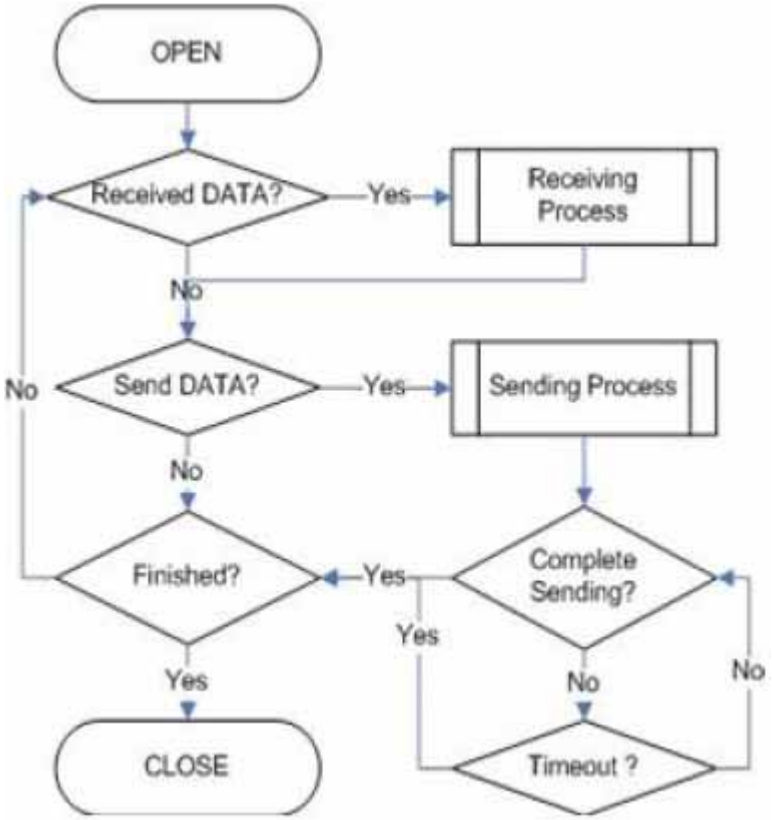


图15 IPRAW操作流程

SOCKET初始化

它选择其中的一个端口设置协议号，设置Sn_MR(P3:P0)为IPRAW模式，然后执行OPEN命令。OPEN命令后，如果SOCKET状态为SOCK_IPRAW，那么SOCKET初始化完成。

```
{
    START:
    /* sets Protocol number */
    /* The protocol number is used in Protocol Field of IP Header. */
    Sn_PROTO = protocol_num;
    /* sets IP raw mode */
    Sn_MR = 0x03;
    /* sets OPEN command */
    Sn_CR = OPEN;
    /* wait until Sn_SSR is changed to SOCK_IPRAW */
    if (Sn_SSR != SOCK_IPRAW) Sn_CR = CLOSE; goto START;
}
```

接收数据？

参考“ 5.2.2.1 单播和广播 ”。

接收数据处理

它处理内部RX存储器中收到的IPRAW数据。IPRAW数据格式如下：

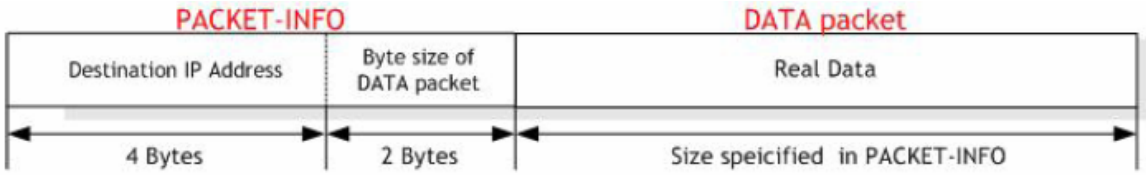


图16. 接收的IPRAW数据格式

IPRAW的数据包括6个字节的PACKET-INFO和数据字节。在PACKET-INFO中包含发送者的信息(IP地址)和数据字节长度。在IPRAW模式，数据处理过程与UDP相同，只是在PACKET-INFO中没有端口号

信息。参考“ 5.2.2.1 单播和广播 ”。

如果发送数据的字节长度比SOCKET的RX存储器的剩余空间大，SOCKET则不能接收数据。也不能接收分片数据。

发送数据？/发送处理

发送数据的字节长度不能比该SOCKET的TX存储器大，也不能大于默认的MTU。在IPRAW模式下发送数据的处理过程与UDP相同，只是不需要配置目的端口号。参考“ 5.2.2.1 单播和广播 ”。

发送结束和超时

结束？/关闭SOCKET

与UDP通信相同，参考“ 5.2.2 UDP ”。

5.2.4 MACRAW

MACRAW是基于比IP层更低的以太网的MAC层的通信，只有SOCKET0可以在MACRAW模式下通信。即使SOCKET0工作在MACRAW模式，SOCKET1~SOCKET7仍然可以同时工作在TCP/IP的其它模式。这时，SOCKET0作为NIC(网络接口控制器)，通过它可以实现软件的TCP/IP协议。

这也就是W5300的混合TCP/IP的概念——支持硬件的TCP/IP和软件的TCP/IP。使用混合TCP/IP，可以克服W5300的SOCKET的限制。如果需要高性能的数据传输，可以使用硬件的TCP/IP的SOCKET来实现。对于那些一般要求的数据通信，可以通过MACRAW模式用软件的TCP/IP来实现。除了SOCKET1~SOCKET7使用的协议，SOCKET0的MACRAW模式可以处理所有的协议。因为MACRAW是纯以太网的数据包的通信方法，因此工程师必须具有软件TCP/IP协议栈的相关知识。

由于MACRAW是基于以太网MAC层的数据，它必须有6字节的源硬件地址、目的硬件地址和2字节的以太网的类型。

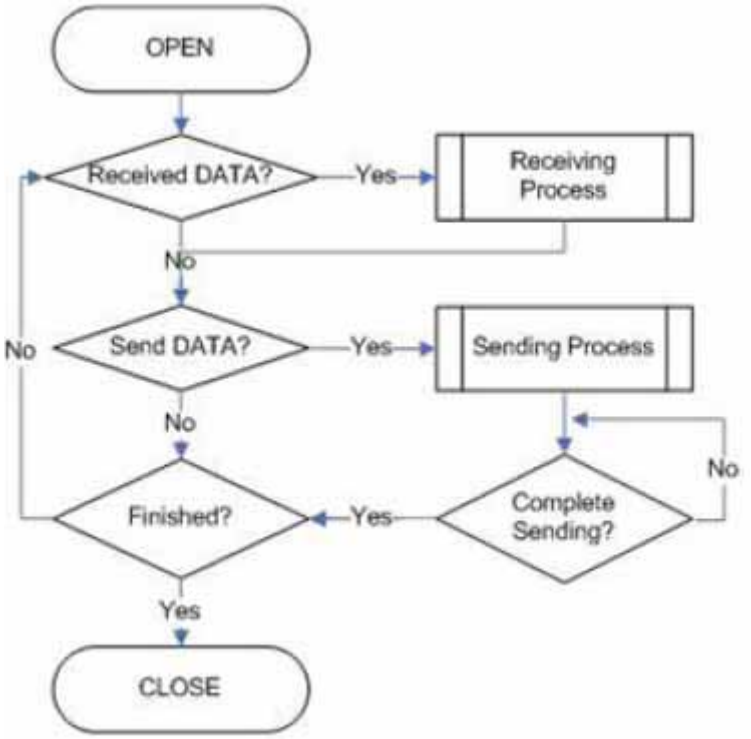


图17 MACRAW操作流程

SOCKET初始化

选择SOCKET0，设置S0_MR（P3:P0）为MACRAW模式，然后运行OPEN命令。

运行OPEN命令后，当SOCKET0的状态改变为SOCK_MACRAW时，SOCKET0的初始化完成。由于所有的通信信息（本机硬件地址、本机IP地址、本机端口号、目的硬件地址、目的IP地址、目的端口号以及所有协议头的类型等）都包含在数据中，因此不需要设置相关的寄存器。

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

接收数据？

参考“ 5.2.2.1 单播和广播 ”。

接收处理

它处理SOCKET0内部RX存储器的MACRAW数据。接收到的MACRAW数据格式如下：

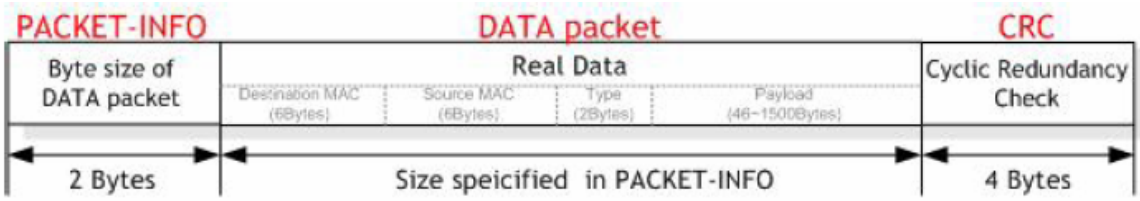


图18 接收的MACRAW数据格式

MACRAW数据包括2个字节的PACKET-INFO、数据包和4个字节的CRC。

PACKET-INFO包括2个字节的数据包长度信息。而数据包则包含6个字节的目的MAC地址，6个字节的源地址，2个字节的协议类型以及46~1500个字节的有效数据。数据包的有效数据是Internet协议，如ARP或IP。详细信息参考：http://www.iana.org/assignments/ethernet-numbers

MACRAW的CRC必须由主机通过S0_RX_FIFOR读取，可以忽略。

```
{
    /* extract size of DATA packet from internal RX memory */
    pack_size = S0_RX_FIFOR;

    /* calculate the read count of Sn_RX_FIFOR */
    if (pack_size is odd ?) read_cnt = (pack_size + 1) / 2;
    read_cnt = pack_size / 2;

    /* extract DATA packet from internal RX memory */
    for( i = 0; i < read_cnt; i++)
    {
        data_buf[i] = S0_RX_FIFOR; /* data_buf is array of 16bit */
```

```
    }

    /* extract 4 bytes CRC from internal RX memory and then ignore it */
    dummy = S0_RX_FIFOR;
    dummy = S0_RX_FIFOR;

    /* set RECV command */
    S0_CR = RECV;
  }
```

注意：

当内部RX存储器的剩余空间比接收的MACRAW数据小，在内部RX存储器中可能会保存一些不希望接收到的PACKET-INFO和数据，这样，在分析PACKET-INFO（如上面的代码）和接收正确的数据时将产生错误。这种错误在内部RX存储器快存满时很容易发生。要解决这个问题，可以忽略一些丢失的MACRAW数据。

- 尽快处理内部RX存储器中的数据，以防止存储器空间被填满。
- 只接收本机MACRAW的数据，减少主机处理接收数据的负荷。

按照下面SOCKET0初始化的示例代码设置S0_MR的MF位，

```
  {
    START:
    /* sets MAC raw mode with enabling MAC filter */
    S0_MR = 0x44;

    /* sets OPEN command */
    S0_CR = OPEN;
    /* wait until Sn_SSR is changed to SOCK_MACRAW */
    if (Sn_SSR != SOCK_MACRAW) S0_CR = CLOSE; goto START;
  }
```

当内部RX存储器的剩余空间小于1526字节时（默认的MTU为1514+PACKET-INFO(2)+ DATA packet(8) +CRC(2)），关闭SOCKET0。关闭SOCKET0后，处理接收到的所有MACRAW数据然后重新打

开SOCKET0。

```
{
    /* check the free size of internal RX memory */
    if((TMSR0 * 1024) - Sn_RX_RSR < 1526)
    {
        recved_size = Sn_RX_RSR; /* backup Sn_RX_RSR */
        Sn_CR = CLOSE; /* SOCKET0 Closed */
        while(Sn_SSR != SOCK_CLOSED); /* wait until SOCKET0 is closed */
        /* process all data remained in internal RX memory */
        while(recved_size > 0)
        {
            /* extract size of DATA packet from internal RX memory */
            pack_size = S0_RX_FIFOR;
            /* calculate the read count of Sn_RX_FIFOR */
            if (pack_size is odd ?) read_cnt = (pack_size + 1) / 2;
            read_cnt = pack_size / 2;
            /* extract DATA packet from internal RX memory */
            for( i = 0; i < read_cnt; i++)
            {
                data_buf[i] = S0_RX_FIFOR; /* data_buf is array of 16bit */
            }
            /* extract 4 bytes CRC from internal RX memory and then ignore it */
            dummy = S0_RX_FIFOR;
            dummy = S0_RX_FIFOR;
            /* calculate the size of remained data in internal RX memory*/
            recved_size = recved_size – 2 – pack_size – 4;
        }
```

```
        /* Reopen the SOCKET0 */
        /* sets MAC raw mode with enabling MAC filter */
        S0_MR = 0x44; /* or S0_MR = 0x04 */

        /* sets OPEN command */
        S0_CR = OPEN;
        /* wait until Sn_SSR is changed to SOCK_MACRAW */
        while (Sn_SSR != SOCK_MACRAW);
    }
    else /* process normally the DATA packet from internal RX memory */
    {
    /* This block is same as the code of “Receiving process” stage*/
    }
}
```

发送数据？/发送处理

要发送的数据字节长度不能比SOCKET0的内部TX存储器大，也不能大于MTU。主机按照上面介绍的“接收数据”的格式创建MACRAW的数据。如果主机的数据长度小于60个字节，内部要填充0，以使实际发送的以太网数据包字节数为60个字节。

```
{
    /* first, get the free TX memory size */
    FREESIZE:
    get_free_size = S0_TX_FSR;
    if (get_free_size < send_size) goto FREESIZE;

    /* calculate the write count of Sn_TX_FIFOR */
    if (send_size is odd ?) write_cnt = (send_size + 1) / 2;
    else write_cnt = send_size / 2;
```

```
    /* copy data to internal TX memory */
    for (i = 0; i < write_cnt; i++)
    {
        S0_TX_FIFOR = data_buf[i]; /* data_buf is array of 16bit */
    }

    /* sets transmission data size to Sn_TX_WRSR */
    S0_TX_WRSR = send_size;
    /* set SEND command */
    S0_CR = SEND;
}
```

结束发送？

数据通信的所有协议都是由主机进行处理，因此不会产生超时。

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

结束？/关闭SOCKET

参考“ 5.2.2.1 单播和广播 ”。

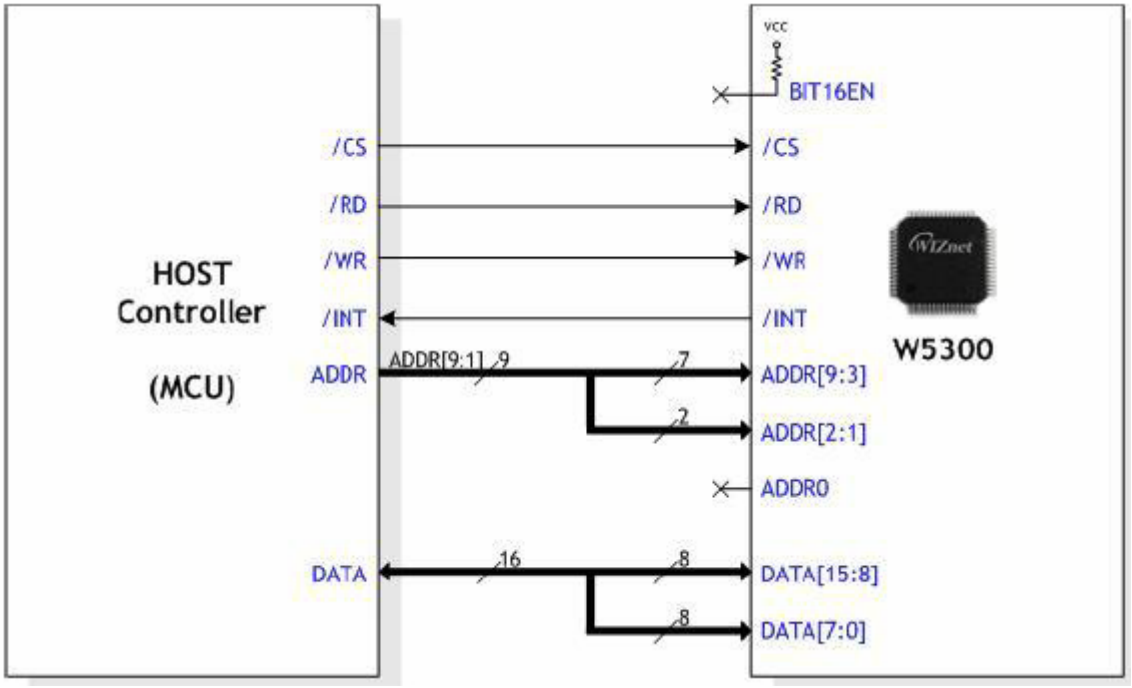
6. 外部接口

W5300与主机的接口有直接地址模式和间接地址模式两种，总线又分为16位和8位两种。另外，W5300可以根据TEST_MODE[3:0]的不同配置选择内部PHY和外部PHY。

6.1 直接地址模式

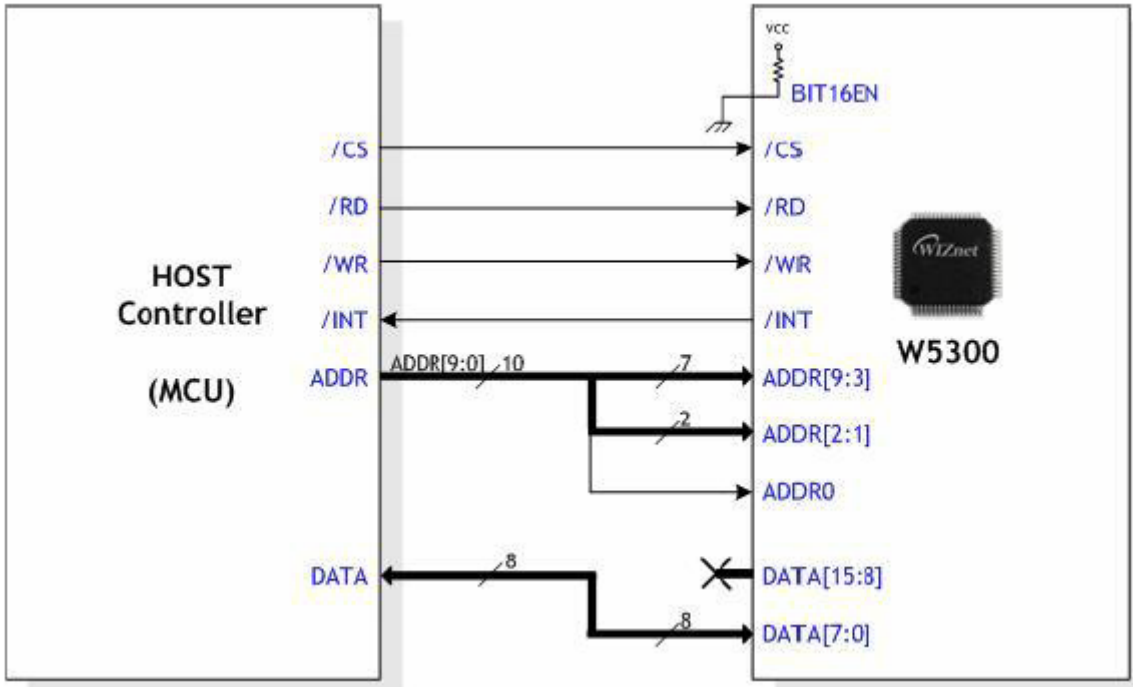
6.1.1 16位数据总线

在16位数据总线模式下，只使用ADDR[9:1]，而ADDR0接地或悬空。因为‘ BIT16EN ’有内部上拉，所以可以悬空。



6.1.2 8位数据总线

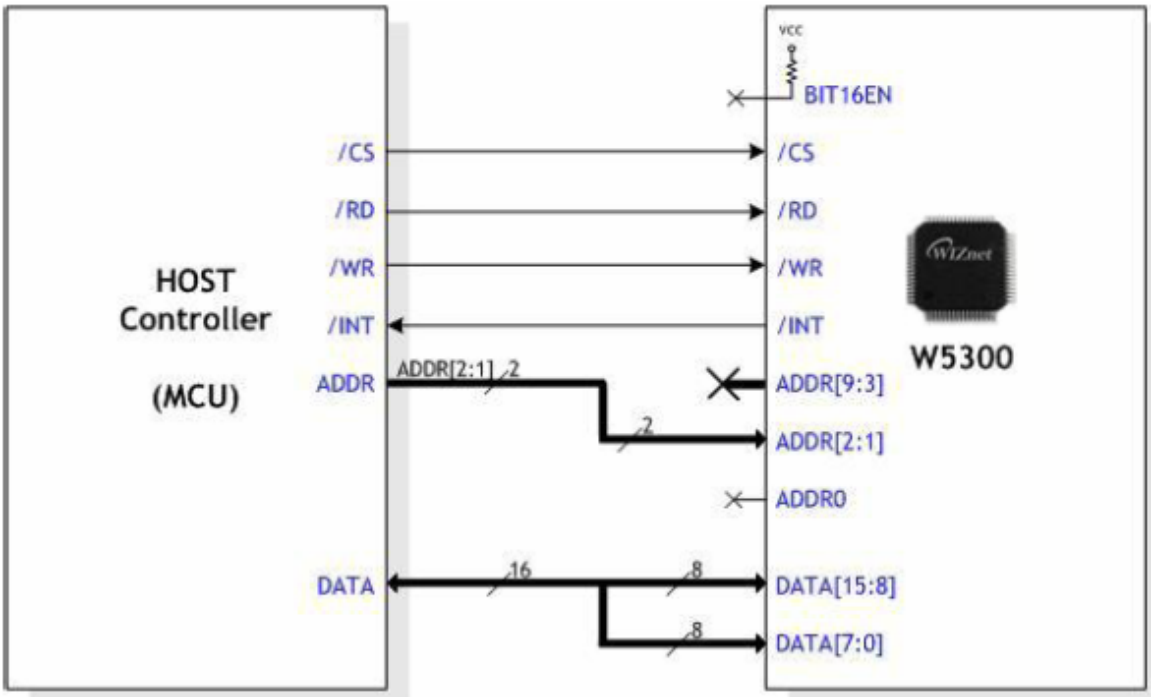
在8位数据总线模式下，ADDR[9:0]都有效，‘ BIT16EN ’必须接低电平（接地）。数据线DATA[15:8]悬空。



6.2 间接地址模式

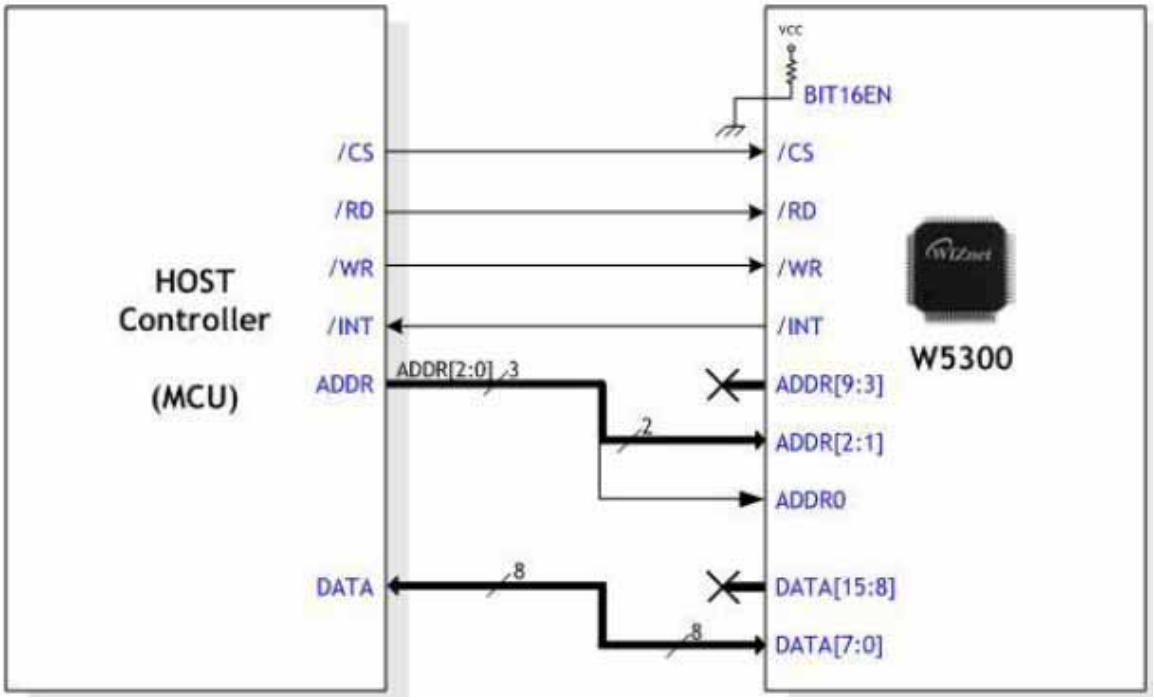
6.2.1 16位数据总线

在16位数据总线模式下，只使用ADDR[2:1]，而ADDR[9:3]及ADDR0接地或悬空。因为‘ BIT16EN ’有内部上拉，所以可以悬空。



6.2.2 8位数据总线

在8位数据总线模式下，只使用ADDR[1:0]，而ADDR[9:2]接地或悬空。‘ BIT16EN ’必须接地。而DATA[15:8]没有使用，可以悬空。



6.3 内部PHY模式

如果使用W5300内部PHY，TEST_MODE[3:0]需要接低电平或悬空。

根据内部PHY的运行模式配置OP_MODE[2:0]，详细信息请参考“ 1.1 配置信号 ”。

为了使内部PHY与隔离变压器之间很好地阻抗匹配，需要一个50ohm(±1%)的电阻和一个0.1uF的电容。

内部PHY支持6个网络指示LED，包括连接状态、速度等。不用的LED信号悬空。将RXLED和TXLED经过AND逻辑可以产生一个ACT_LED（数据传输LED指示）。详细信息参考“ 1.6 网络LED信号指示 ”。

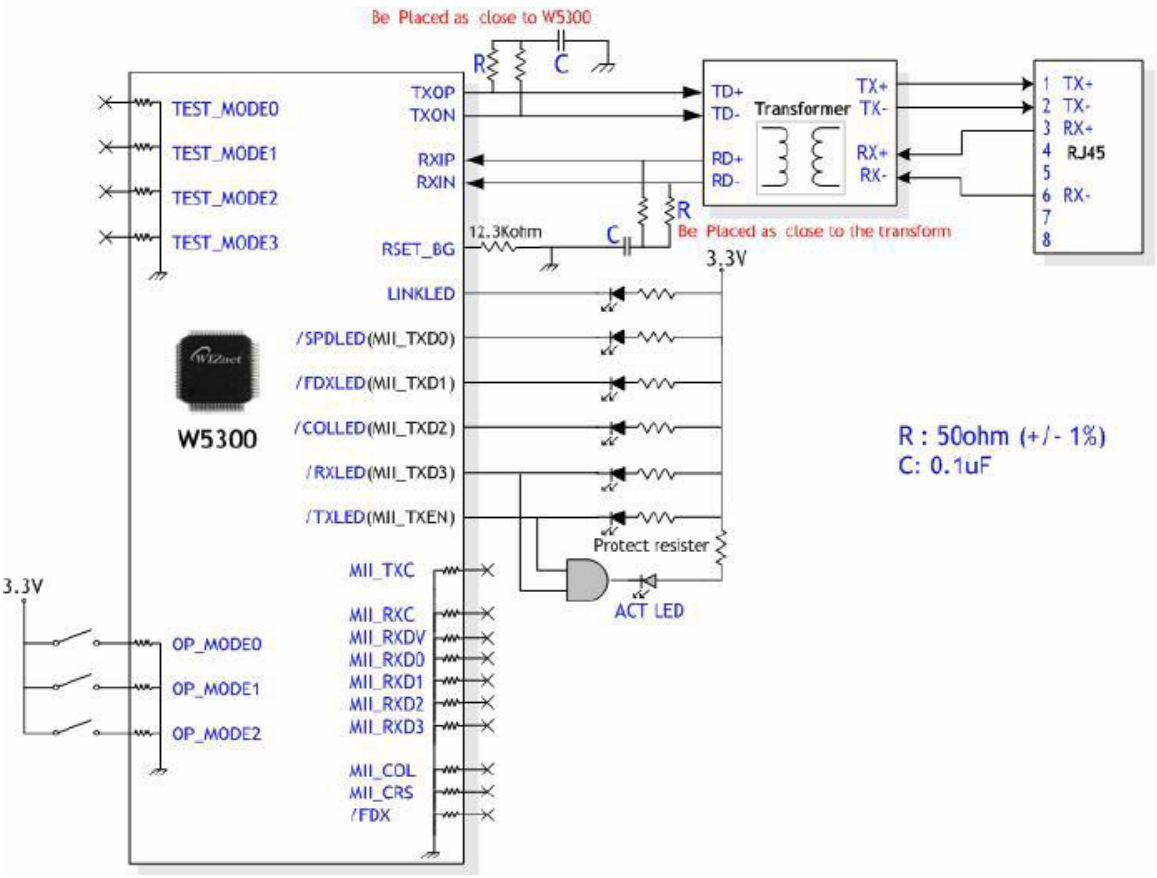


图19. 内部PHY及LED信号

6.4 外部PHY模式

如果内部PHY不能满足用户的需求，可以选用第三方的PHY与W5300接口。在使用外部PHY模式，需要选用W5300的时钟源。当TEST_MODE0是逻辑高电平时使用晶体，当TEST_MODE1为高电平时，使用振荡器。

详细信息请参考“ 1.1 配置信号 ”和“ 1.7 时钟信号 ”。

为了使外部PHY 与隔离变压器之间很好地阻抗匹配，请参看PHY厂家提供的技术手册。

W5300的‘ /FDX ’引脚与外部PHY的双工指示器信号连接。

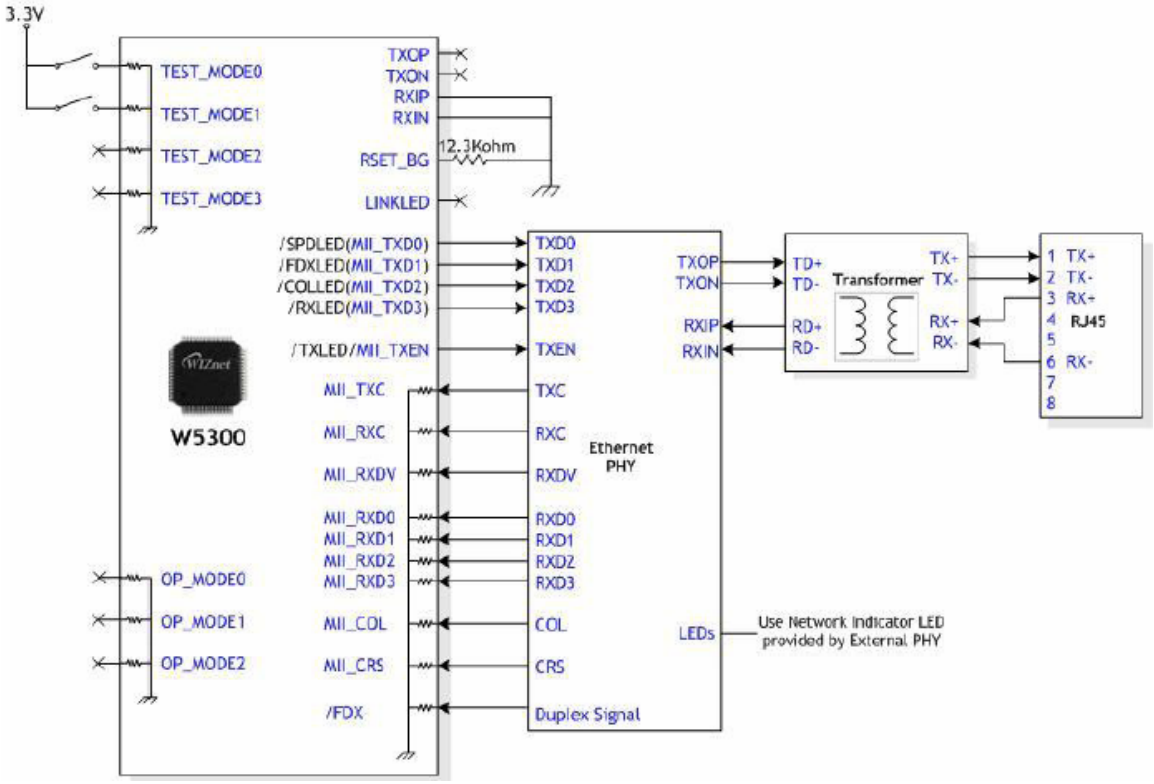


图20 外部PHY与W5300的MII接口

7. Electrical Specifications

极限参数

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

注意：器件在超过极限参数使用时可能会造成永久的损坏。

直流参数

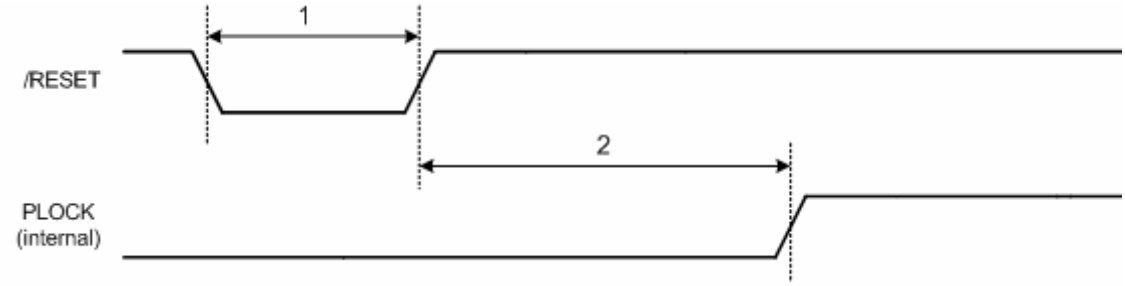
Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
V_{DD}	DC Supply voltage	Junction temperature is from -55°C to 125°C	3.0	3.3	3.6	V
V_{IH}	High level input voltage		2.0		5.5	V
V_{IL}	Low level input voltage		- 0.5		0.8	V
V_{OH}	High level output voltage	I_{OH} = 2, 4, 8, 12, 16, 24 mA	2.0	3.3	3.63	V
V_{OL}	Low level output voltage	I_{OL} = -2, -4, -8, -12, -16, -24 mA	0.0		0.4	V
I_I	Input Current	$V_{IN} = V_{DD}$			±5	µA
I_O	Output Current	$V_{OUT} = V_{DD}$	2		8	mA

功耗

Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
P_{IA}	Power consumption when using the auto-negotiation of internal PHY mode	Vcc 3.3V Temperature 25°C	-	180	250	mA
P_{IM}	Power consumption when using manual configuration of internal PHY mode	Vcc 3.3V Temperature 25°C	-	175	210	mA
P_E	Power consumption when using external PHY mode	Vcc 3.3V Temperature 25°C		65	150	mA

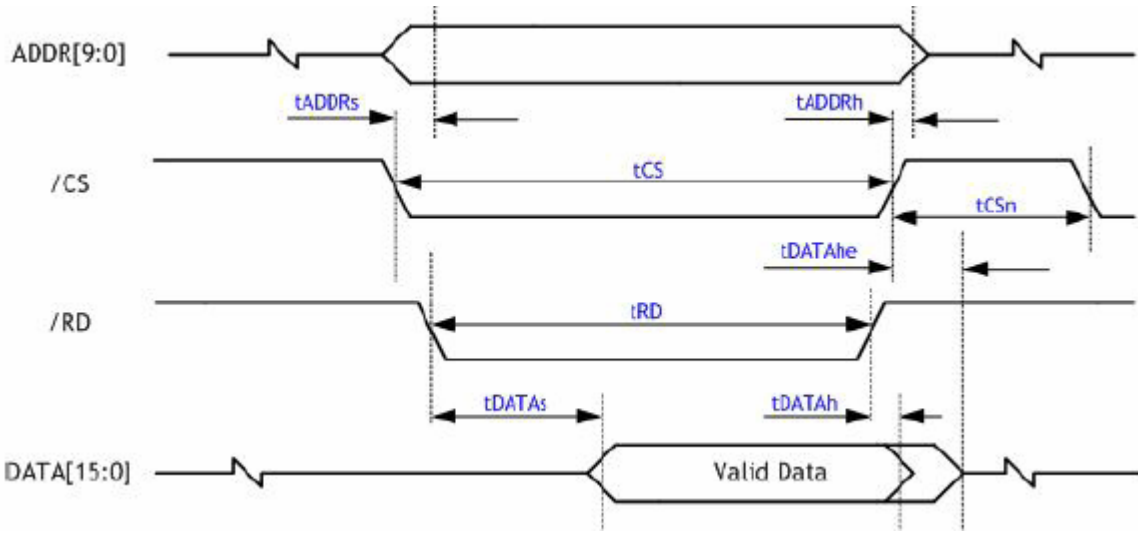
交流特性

复位时序



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

寄存器读时序

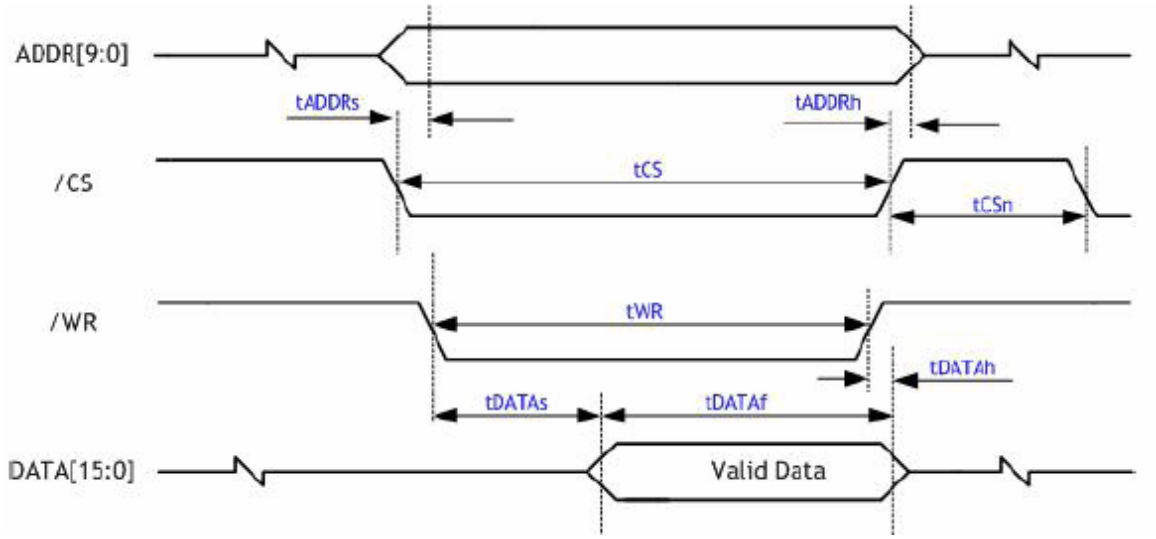


Description		Min	Max
tADDRs	Address Setup Time after /CS and /RD low	-	7 ns
tADDRh	Address Hold Time after /CS or /RD high	-	-
tCS	/CS Low Time	65 ns	-
tCSn	/CS Next Assert Time	28 ns	-
tRD	/RC Low Time	65 ns	-
tDATAs	DATA Setup Time after /RD low	42 ns	-
tDATAh	DATA Hold Time after /RD and /CS high	-	7 ns
tDATAhe	DATA Hold Extension Time after /CS high	-	2XPLL_CLK

注意：

‘ tDATAhe ’在MR(RDH)为‘ 1 ’时是数据保持时间。在这段时间里，/CS为低电平后的2XPLL_CLK时间里数据总线被驱动。因此注意数据总线上的冲突。

寄存器写时序



Description		Min	Max
tADDRs	Address Setup Time after /CS and /WR low	-	7 ns
tADDRh	Address Hold Time after /CS or /RD high	-	-
tCS	/CS low Time	50 ns	-
tCSn	/CS next Assert Time	28 ns	
tWR	/WR low time	50 ns	
tDATAs	Data Setup Time after /WR low	7 ns	7ns + 7XPLL_CLK
tDATAf	Data Fetch Time	14 ns	tWR-tDATAs
tDATAh	Data Hold Time after /WR high	7 ns	-

注意：

‘ tDATAs ’是主机写入数据的保持时间。根据MR(WDF2~WDF0)的设置值设置在7个PLL_CLK之间。

由于‘ tDATAf ’是W5300取主机写入数据的时间，如果/WR在这之前没有变为高电平，主机写入的数据在/WR没变为高电平时被W5300取走，而不考虑‘ tDATAf ’。

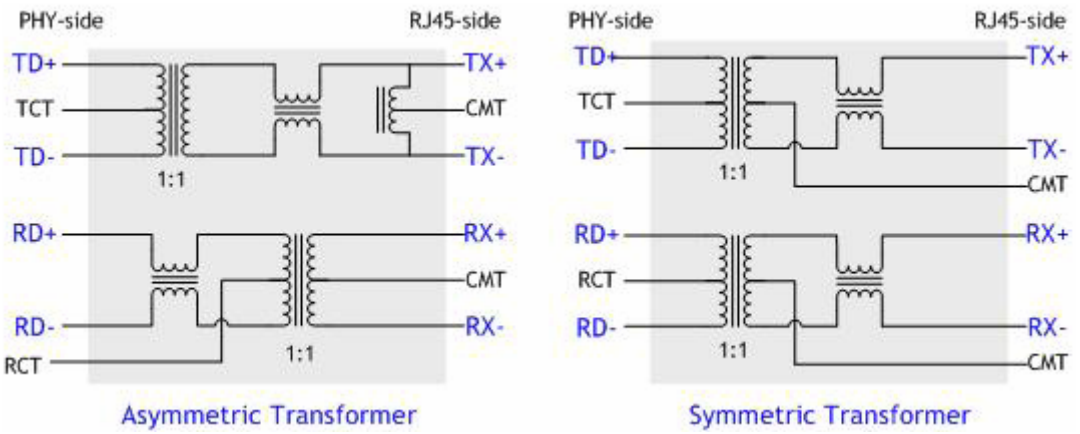
为了保证这时读数的有效性，必须保证‘ tDATAh ’保持时间。

晶体特性

Parameter	Range
Frequency	25 MHz
Frequency Tolerance (at 25℃)	±30 ppm
Shunt Capacitance	7pF Max
Drive Level	1 ~ 500uW (100uW typical)
Load Capacitance	27pF
Aging (at 25℃)	±3ppm / year Max

变压器特性

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



如果使用内部PHY模式，应该使用对称的变压器，以便支持MDI/MDIX（交叉）。

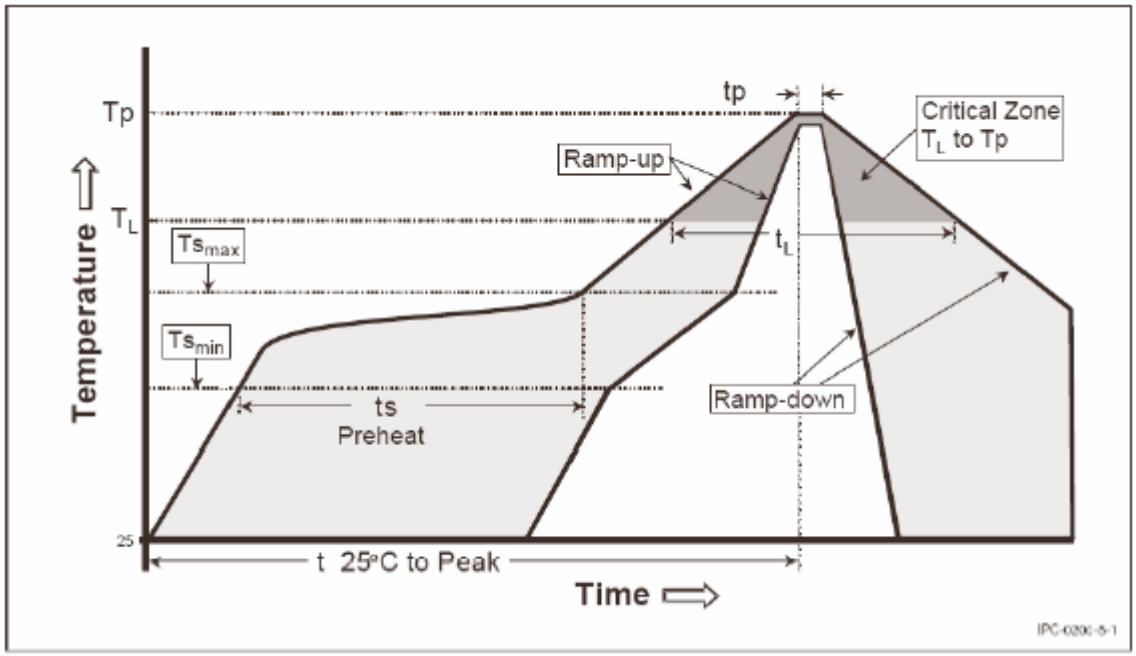
如果使用外部PHY模式，变压器应该与外部的PHY相适应。

8. IR Reflow Temperature Profile (Lead-Free)

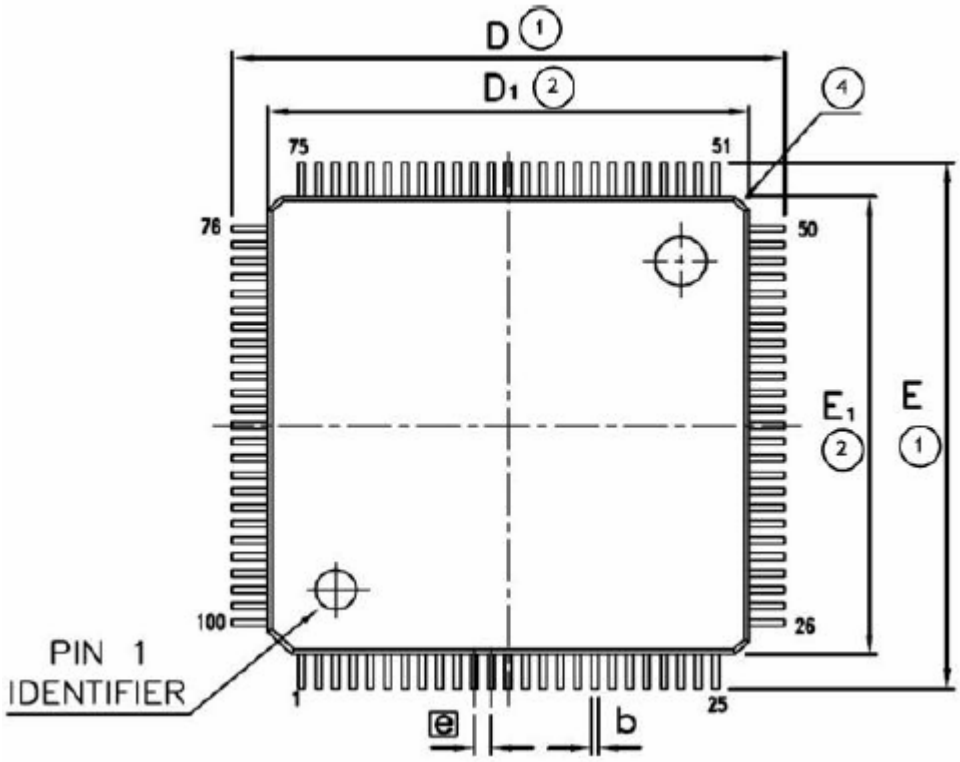
Moisture Sensitivity Level : 3

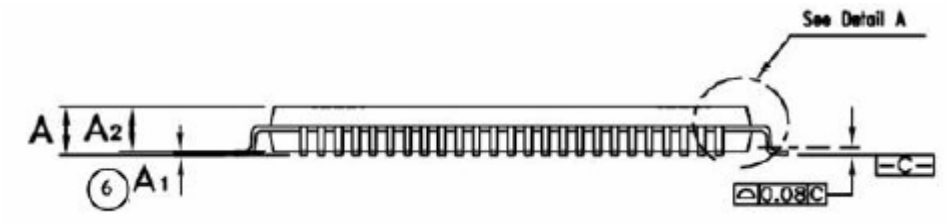
Dry Pack Required : Yes

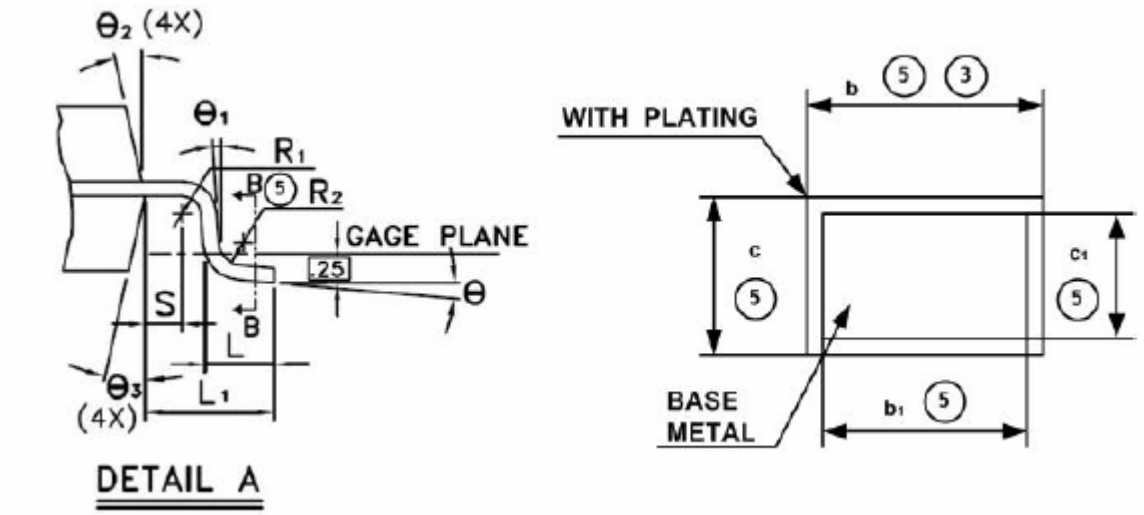
Average Ramp-Up Rate (Ts_{max} to Tp)	3° C/second max.
Preheat - Temperature Min (Ts_{min}) - Temperature Max (Ts_{max}) - Time (ts_{min} to ts_{max})	 150 °C 200 °C 60-180 seconds
Time maintained above: - Temperature (TL) - Time (tL)	 217 °C 60-150 seconds
Peak/Classification Temperature (Tp)	260 +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



9. 封装描述







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

<NOTE>

① To be determined at seating plane -C-.

② Dimensions 'D_1' and 'E_1' do not include mold protrusion.
'D_1' and 'E_1' are maxium plastic body size dimensions including mold mismatch.

③ Dimension 'b' does not include dambar protrusion.
Dambar can not be located on the lower radius or the foot.

④ Exact shape of each corner is optional

⑤ These Dimensions apply to the flat section of the lead between 0.10mm and 0.25mm from the lead tip.

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

7 Controlling dimension : Millimeter

8 Reference Document : JEDEC MS-026 , BED.