DSPF2833X 与 W5500 通信移植说明

以 DSPF28335 与 W5500 基于 SPI 通信为例:

一、特性

W5500 特性

- 支持硬件 TCP/IP 协议：TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE
- 支持 8 个独立端口（Socket）同时通讯
- 支持掉电模式
- 支持网络唤醒
- 支持高速串行外设接口（SPI 模式 0，3）
- 内部 32K 字节收发缓存
- 内嵌 10BaseT/100BaseTX 以太网物理层（PHY）
- 支持自动协商（10/100-Based 全双工/半双工）
- 不支持 IP 分片
- 3.3V 工作电压，I/O 信号口 5V 耐压
- LED 状态显示（全双工/半双工，网络连接，网络速度，活动状态）
- LQFP48 无铅封装（7x7mm,间距 0.5mm）
- 嵌入式服务器

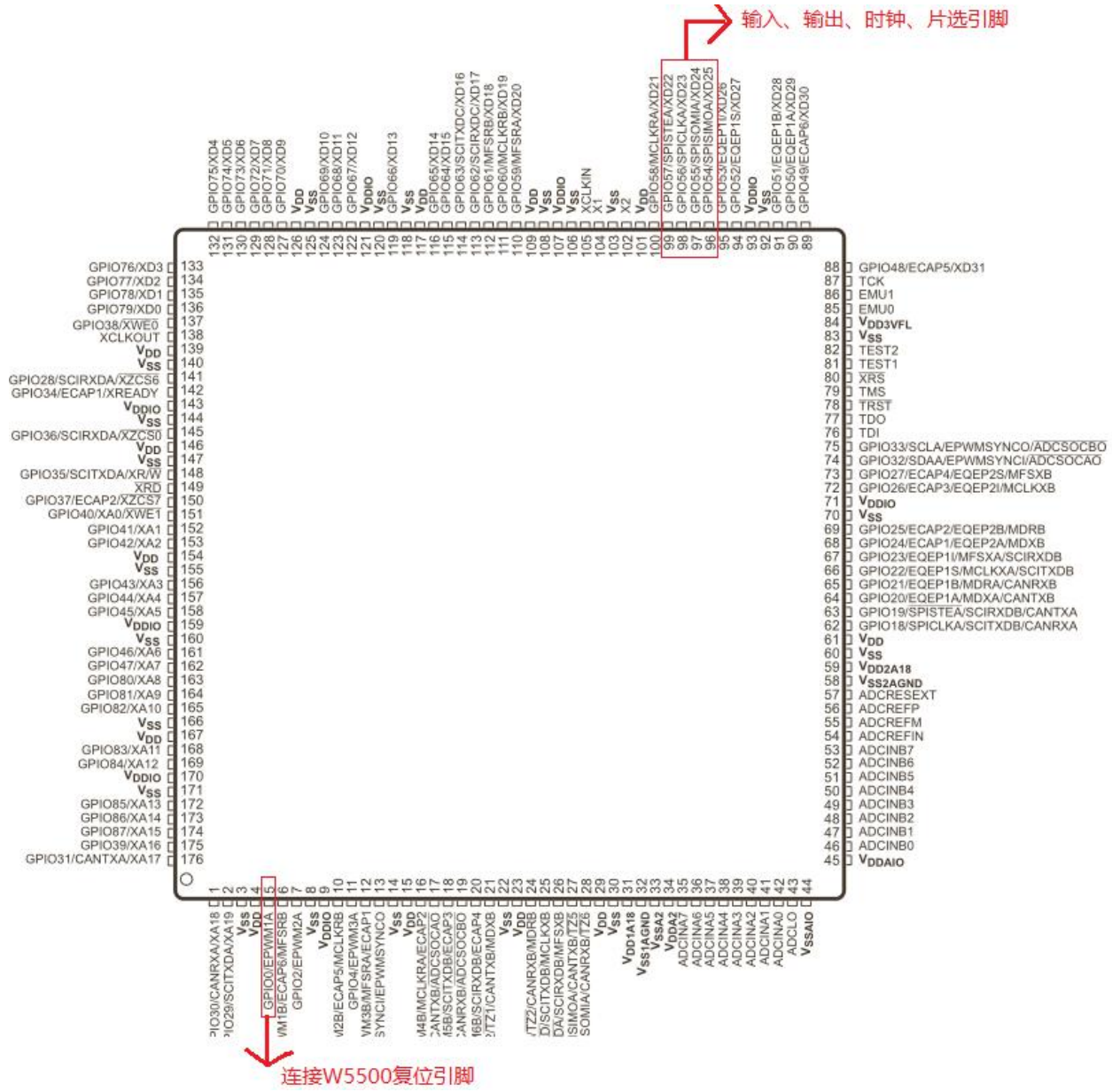
DSPF28335 特性

- 基于 TMS320F28335 浮点 DSP 控制器
- 高性能的静态 CMOS 技术，指令周期为 6.67 女生，主频达 150NHz
- 6 通道的 DMA 控制器
- 8 个外部中断
- 增强型外设模块：18 个 PWM 输出，包含 6 个高分辨率脉宽调制模块（HRPWM）、6 个事件捕获输入、2 通道的正交调制模块（QEP）
- 3 个 32 位的定时器，定时器 0 和定时器 1 用作一般的定时器，定时器 0

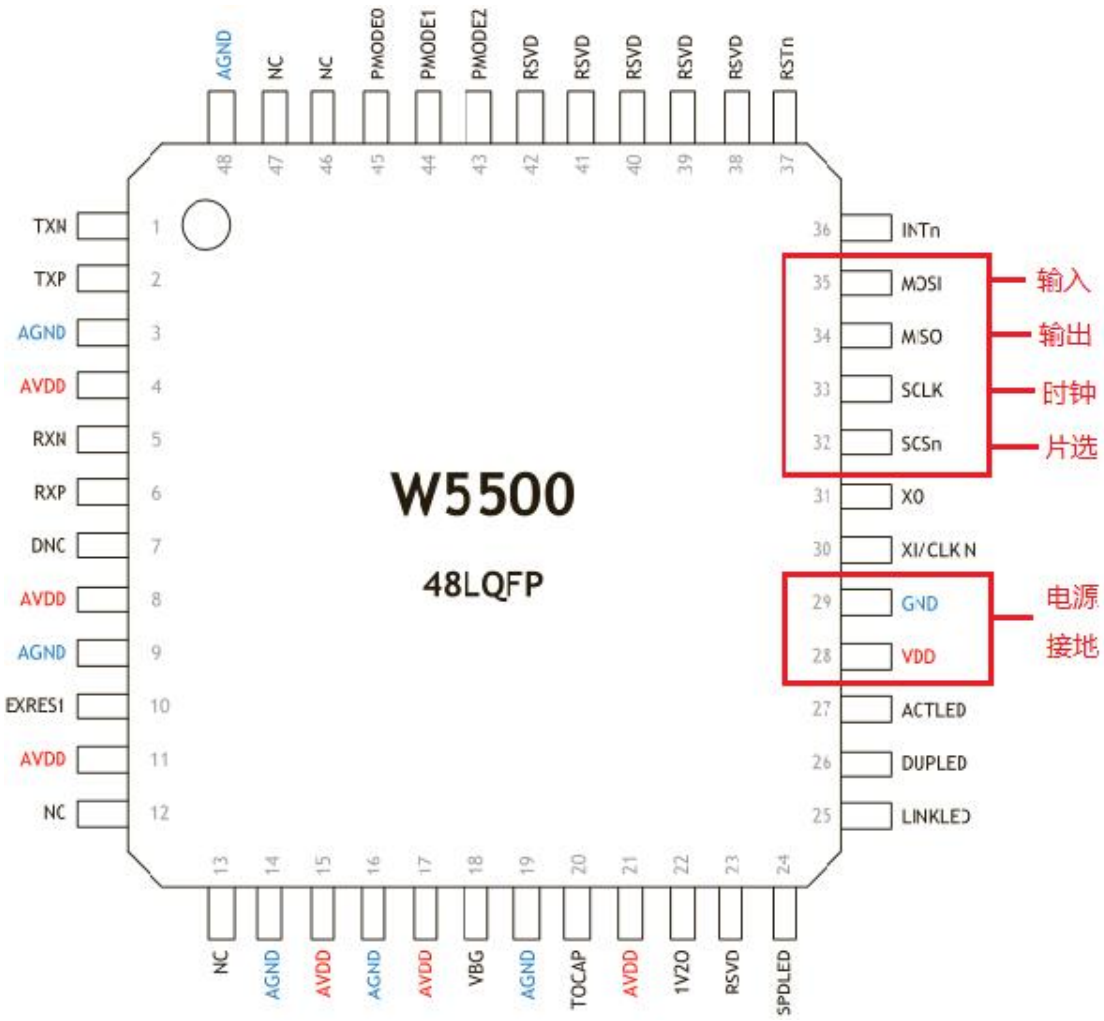
- 接到 PIE 模块，定时器 1 接到中断 INTI3、定时器 2 用于 DSP/BIOS 的片上实时系统，连接到中断 INT14，如果系统不使用 DSP/bios，定时器 2 可用于一般定时器
- 串行外设 2 通道 CAN 模块、3 通道 SCI 模块、2 个 McBSP(多通道缓冲串行接口)模块、1 个 SPI 模块、1 个 I2C 主从兼容的串行总线接口模块
- 12 位的 A/D 转换器具有 16 个转换通道、2 个采样保持器、内外部参考电压、转换速度为 80ns，同时支持多通道转换
- 88 个可编程的复用 GPIO 引脚
- TI 主推高性能 TMS320C28x 系列 DSP 控制器，主频高达 150MHz
- 具备 I2C、SPI、eCAN、ePWM 等总线接口，适用于各种控制类工业设备
- 体积小、性能强、便携性高，同时适用于多种手持设备
- 符合高低温、振动要求，满足工业环境应用

2、引脚分配图：

DSPF28335 引脚分配图：


输入、输出、时钟、片选引脚
连接W5500复位引脚

W5500 引脚分配图:



3、环境：

软件环境：Windows xp/Windows 7/Windows 8/Windows 10

硬件环境：DSPF28335 、 W5500

开发工具：Code Composer studio 6.0.0 、 USR-TCP232-Test（调试工具）

二、步骤

1、连接 DSPF28335 与 W5500（选择引脚）

首先确认 W5500 与 DSPF28335 开发板上分别需要连接的引脚

（1）W5500：引脚描述

SCSn	Pull-up	I	**片选（Chip Select for SPI bus）** 选用 W5500 的 SPI 接口，该引脚低电平有效： 低电平：选用； 高电平：不选用；
SCLK	-	I	**SPI 时钟输入（SPI clock input）** 该引脚用于接收 SPI 主机的 SPI 时钟信号
MISO	-	O	**SPI 主机输入从机（W5500）输出**
MOSI	-	I	**SPI 主机输出从机（W5500）输入**

（2）DSPF28335 引脚描述:

GPIO54 SPISIMOA XD25	96	L13	J14	通用输入/输出 54 (I/O/Z) SPI-A 从器件输入，主器件输出 (I/O) 外部接口数据线路 25 (I/O/Z)
GPIO55 SPISOMIA XD24	97	L14	J13	通用输入/输出 55 (I/O/Z) SPI-A 从器件输出，主器件输入 (I/O) 外部接口数据线路 24 (I/O/Z)
GPIO56 SPICLKA XD23	98	K11	J12	通用输入/输出 56 (I/O/Z) SPI-A 时钟 (I/O) 外部接口数据线路 23 (I/O/Z)
GPIO57 SPISTEA XD22	99	K13	H13	通用输入/输出 57 (I/O/Z) SPI-A 从器件发送使能 (I/O) 外部接口数据线路 22 (I/O/Z)

选择好两块开发板上的引脚后，对两块开发板进行连接：

型号 / 序号	W5500	DSPF28335	连接方式
1	MISO	54	从输入主输出
2	MOSI	55	从输出主输入
3	SCK	56	时钟
4	CS	57	片选
5	3.3v	3.3v	电源
6	GND	GND	地线

2、引脚初始化

（1）设置 DSPF28335 上所选引脚全部设置为上拉

```
GpioCtrlRegs.GPBPUD.bit.GPIO54 = 0;     // Enable pull-up on GPIO54 (SPISIMOA)
```

```
GpioCtrlRegs.GPBPUD.bit.GPIO55 = 0;    // Enable pull-up on GPIO55 (SPISOMIA)
GpioCtrlRegs.GPBPUD.bit.GPIO56 = 0;    // Enable pull-up on GPIO56 (SPICLKA)
GpioCtrlRegs.GPBPUD.bit.GPIO57 = 0;    // Enable pull-up on GPIO57 (SPISTEA)
```

（2）设置 DSPF28335 上所选引脚为异步输入

```
GpioCtrlRegs.GPBQSEL2.bit.GPIO54 = 3; // Asynch input GPIO54 (SPISIMOA)


GpioCtrlRegs.GPBQSEL2.bit.GPIO55 = 3;  // Asynch input GPIO55 (SPISOMIA)
GpioCtrlRegs.GPBQSEL2.bit.GPIO56 = 3; // Asynch input GPIO56 (SPICLKA)
GpioCtrlRegs.GPBQSEL2.bit.GPIO57 = 3; // Asynch input GPIO57 (SPISTEA)
```

（3）配置 DSPF28335 上引脚的功能

```
GpioCtrlRegs.GPBMUX2.bit.GPIO54 = 1; // 配置 GPIO54 引脚为 SPISIMOA
GpioCtrlRegs.GPBMUX2.bit.GPIO55 = 1;  // 配置 GPIO55 引脚为 SPISOMIA
GpioCtrlRegs.GPBMUX2.bit.GPIO56 = 1; // 配置 GPIO56 引脚为 SPICLKA
GpioCtrlRegs.GPBMUX2.bit.GPIO57 = 0; // 配置 GPIO57 引脚为 SPISTEA
```

3、SPI 配置

根据开发板的数据手册进行配置，对每个引脚的参数进行相应的配置

```
SSpiaRegs.SPICCR.bit.SPISWRESET = 0;
SpiaRegs.SPICCR.all = 0x0047;//The SPI software resets the polarity bit
                        //to 1 (sending data along the falling edge),
                        //moving in and out of the 8 bit word length each
time,
                        //and  prohibiting  the  SPI  internal  loopback
(LOOKBACK) function;
    SpiaRegs.SPICTL.all = 0x0006;     // Enable master mode, normal phase, // enable
talk, and SPI int disabled.
    SpiaRegs.SPISTS.all = 0x0000;     //溢出中断，禁止 SPI 中断；
    SpiaRegs.SPIBRR = 0x001F;            //SPI 波特率=37.5M/24=1.5MHZ；
    SpiaRegs.SPIPRI.bit.FREE = 1;     //Set so breakpoints don't disturb xmission
    SpiaRegs.SPICCR.bit.SPISWRESET = 1;
```

根据下图进行对 SPI 进行配置并了解每个引脚的参数：

（1）、配置控制寄存器

使 SPI 处于复位方式、并设为下降沿触发、设为 8 位字符长度

SPI Configuration Control Register (SPICCR)

SPICCR controls the setup of the SPI for operation.

Figure 2-1. SPI Configuration Control Register (SPICCR) — Address 7040h

7	6	5	4	3	2	1	0
SPI SW Reset	CLOCK POLARITY	Reserved	SPILBK	SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0
R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-1. SPI Configuration Control Register (SPICCR) Field Descriptions

Bit	Field	Value	Description
7	SPI SW RESET		SPI software reset. When changing configuration, you should clear this bit before the changes and set this bit before resuming operation.
		0	Initializes the SPI operating flags to the reset condition. Specifically, the RECEIVER OVERRUN Flag bit (SPISTS.7), the SPI INT FLAG bit (SPISTS.6), and the TXBUF FULL Flag bit (SPISTS.5) are cleared. The SPI configuration remains unchanged. If the module is operating as a master, the SPICLK signal output returns to its inactive level.
		1	SPI is ready to transmit or receive the next character. When the SPI SW RESET bit is a 0, a character written to the transmitter will not be shifted out when this bit is set. A new character must be written to the serial data register.
6	CLOCK POLARITY		Shift Clock Polarity. This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin. See Section 1.4.3.
		0	Data is output on rising edge and input on falling edge. When no SPI data is sent, SPICLK is at low level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: • CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal. • CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal.
		1	Data is output on falling edge and input on rising edge. When no SPI data is sent, SPICLK is at high level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: • CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal. • CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal.
5	Reserved		Reads return zero; writes have no effect.
4	SPILBK		SPI loopback. Loop back mode allows module validation during device testing. This mode is valid only in master mode of the SPI.
		0	SPI loop back mode disabled – default value after reset
		1	SPI loop back mode enabled, SIMO/SOMI lines are connected internally. Used for module self tests.
3-0	SPI CHAR3 – SPI CHAR0		Character Length Control Bits 3-0. These four bits determine the number of bits to be shifted in or out as a single character during one shift sequence. Table 2-2 lists the character length selected by the bit values.

通过配置控制寄存器为选择字符长度为 8 位，就是 0x47，转换成二进制就是 0100 0111

Table 2-2. Character Length Control Bit Values

SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

（2）、配置操作寄存器

启用主模式、正常相位、启用对话和禁用 SPI

SPI Operation Control Register (SPICTL)

SPICTL controls data transmission, the SPI's ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

Figure 2-2. SPI Operation Control Register (SPICTL) — Address 7041h

7	6	5	4	3	2	1	0
Reserved			OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-3. SPI Operation Control Register (SPICTL) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved		Reads return zero; writes have no effect.
4	Overrun INT ENA		Overrun Interrupt Enable. Setting this bit causes an interrupt to be generated when the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by the RECEIVER OVERRUN Flag bit and the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.
		0	Disable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts
		1	Enable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts
3	CLOCK PHASE		SPI Clock Phase Select. This bit controls the phase of the SPICLK signal. CLOCK PHASE and CLOCK POLARITY (SPICCR.6) make four different clocking schemes possible (see Figure 1-4). When operating with CLOCK PHASE high, the SPI (master or slave) makes the first bit of data available after SPIDAT is written and before the first edge of the SPICLK signal, regardless of which SPI mode is being used.
		0	Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6)
		1	SPICLK signal delayed by one half-cycle; polarity determined by the CLOCK POLARITY bit
2	MASTER / SLAVE		SPI Network Mode Control. This bit determines whether the SPI is a network master or slave. During reset initialization, the SPI is automatically configured as a network slave.
		0	SPI configured as a slave.
		1	SPI configured as a master.

Bit	Field	Value	Description
1	TALK		Master/Slave Transmit Enable. The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset.
		0	Disables transmission: • Slave mode operation: If not previously configured as a general-purpose I/O pin, the SPISOMI pin will be put in the high-impedance state. • Master mode operation: If not previously configured as a general-purpose I/O pin, the SPISIMO pin will be put in the high-impedance state.
		1	Enables transmission For the 4-pin option, ensure to enable the receiver's $\overline{\text{SPISTE}}$ input pin.
0	SPI INT ENA		SPI Interrupt Enable. This bit controls the SPI's ability to generate a transmit/receive interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit.
		0	Disables interrupt
		1	Enables interrupt

（3）、配置状态寄存器

溢出中断，禁止 SPI 中断

SPI Status Register (SPIST)

Figure 2-3. SPI Status Register (SPIST) — Address 7042h

7	6	5	4	0
RECEIVER OVERRUN FLAG[1][2]	SPI INT FLAG[1][2]	TX BUF FULL FLAG[2]	Reserved	
R/C-0	R/C-0	R/C-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

(1) The RECEIVER OVERRUN FLAG bit and the SPI INT FLAG bit share the same interrupt vector.
(2) Writing a 0 to bits 5, 6, and 7 has no effect.

Table 2-4. SPI Status Register (SPIST) Field Descriptions

Bit	Field	Value	Description
7	RECEIVER OVERRUN FLAG		SPI Receiver Overrun Flag. This bit is a read/clear-only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost (when the SPIRXBUF was overwritten by the SPI module before the previous character was read by the user application). The SPI requests one interrupt sequence each time this bit is set if the OVERRUN INT ENA bit (SPICTL.4) is set high. The bit is cleared in one of three ways: • Writing a 1 to this bit • Writing a 0 to SPI SW RESET (SPICCR.7) • Resetting the system If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt upon the first occurrence of setting the RECEIVER OVERRUN Flag bit. Subsequent overruns will not request additional interrupts if this flag bit is already set. This means that in order to allow new overrun interrupt requests the user must clear this flag bit by writing a 1 to SPISTS.7 each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN Flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately re-entered when the interrupt service routine is exited.
		0	Writing a 0 has no effect
		1	Clears this bit. The RECEIVER OVERRUN Flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN Flag bit and SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.

Bit	Field	Value	Description
6	SPI INT FLAG		SPI Interrupt Flag. SPI INT FLAG is a read-only flag. The SPI hardware sets this bit to indicate that it has completed sending or receiving the last bit and is ready to be serviced. The received character is placed in the receiver buffer at the same time this bit is set. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0) is set.
		0	Writing a 0 has no effect
		1	This bit is cleared in one of three ways: • Reading SPIRXBUF • Writing a 0 to SPI SW RESET (SPICCR.7) • Resetting the system
5	TX BUF FULL FLAG		SPI Transmit Buffer Full Flag. This read-only bit gets set to 1 when a character is written to the SPI Transmit buffer SPITXBUF. It is cleared when the character is automatically loaded into SPIDAT when the shifting out of a previous character is complete.
		0	Writing a 0 has no effect
		1	This bit is cleared at reset.
4-0	Reserved	1 0	Reads return zero; writes have no effect.

（4）、配置波特率

根据以下公式配置波特率

SPI Baud Rate Register (SPIBRR)

SPIBRR contains the bits used for baud-rate selection.

Figure 2-4. SPI Baud Rate Register (SPIBRR) — Address 7044h

7	6	5	4	3	2	1	0
Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 2-5. Field Descriptions

Bit	Field	Value	Description
7	Reserved		Reads return zero; writes have no effect.
6-0	SPI BIT RATE 6–SPI BIT RATE 0		SPI Bit Rate (Baud) Control. These bits determine the bit transfer rate if the SPI is the network master. There are 125 data-transfer rates (each a function of the CPU clock, LSPCLK) that can be selected. One data bit is shifted per SPICLK cycle. (SPICLK is the baud rate clock output on the SPICLK pin.) If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master; therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's SPICLK signal divided by 4. In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the following formula: For SPIBRR = 3 to 127: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$ For SPIBRR = 0, 1, or 2: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$ where: LSPCLK = Function of CPU clock frequency X low-speed peripheral clock of the device SPIBRR = Contents of the SPIBRR in the master SPI device

4、W5500 的配置

首先设定传输缓冲区的大小，数据写入命令和写入数据长度 ，数据写入（写入 1byte 数据）等，再通过配置初始化 IP 信息并打印，初始化 8 个 Socket， （ **void**

set_network(void)）函数的设置，控制数据的接受与发送。

例：

```
    void IINCHIP_WRITE( uint32 addrbsb,  uint8 data)
{

   //IINCHIP_ISR_DISABLE();              // Interrupt Service Routine Disable
   IINCHIP_CSoff();                          // CS=0, SPI start
   IINCHIP_SpiSendData( (addrbsb & 0x00FF0000)>>16);// Address byte 1
   IINCHIP_SpiSendData( (addrbsb & 0x0000FF00)>> 8);// Address byte 2
   IINCHIP_SpiSendData( (addrbsb & 0x000000F8) + 4);// Data write command and
Write data length 1
   IINCHIP_SpiSendData(data);                // Data write (write 1byte data)
   IINCHIP_CSon();                           // CS=1,  SPI end
   //IINCHIP_ISR_ENABLE();                // Interrupt Service Routine Enable
    }
```

5、数据的发送与接收

使用 SPITXBUF 和 SPIRXBUF 来实现

SPITXBUF　　发送读取数据

配置串口传输缓冲寄存器

2.1.7 *SPI Serial Transmit Buffer Register (SPITXBUF)*

SPITXBUF stores the next character to be transmitted. Writing to this register sets the TX BUF FULL Flag bit (SPISTS.5). When transmission of the current character is complete, the contents of this register are automatically loaded in SPIDAT and the TX BUF FULL Flag is cleared. If no transmission is currently active, data written to this register falls through into the SPIDAT register and the TX BUF FULL Flag is not set.

In master mode, if no transmission is currently active, writing to this register initiates a transmission in the same manner that writing to SPIDAT does.

Figure 2-7. SPI Serial Transmit Buffer Register (SPITXBUF) — Address 7048h

15	14	13	12	11	10	9	8
TXB15	TXB14	TXB13	TXB12	TXB11	TXB10	TXB9	TXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 2-8. SPI Serial Transmit Buffer Register (SPITXBUF) Field Descriptions

Bit	Field	Value	Description
15	TXB15 – TXB0		Transmit Data Buffer. This is where the next character to be transmitted is stored. When the transmission of the current character has completed, if the TX BUF FULL Flag bit is set, the contents of this register is automatically transferred to SPIDAT, and the TX BUF FULL Flag is cleared. Writes to SPITXBUF must be left-justified.

SPIRXBUF　　接收数据
配置串口接收缓冲寄存器

2.1.6 SPI Serial Receive Buffer Register (SPIRXBUF)

SPIRXBUF contains the received data. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6).

Figure 2-6. SPI Serial Receive Buffer Register (SPIRXBUF) — Address 7047h

15	14	13	12	11	10	9	8
RXB15	RXB14	RXB13	RXB12	RXB11	RXB10	RXB9	RXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-7. SPI Serial Receive Buffer Register (SPIRXBUF) Field Descriptions

Bit	Field	Value	Description
15	RXB15 – RXB0		Received Data. Once SPIDAT has received the complete character, the character is transferred to SPIRXBUF, where it can be read. At the same time, the SPI INT FLAG bit (SPISTS.6) is set. Since data is shifted into the SPI's most significant bit first, it is stored right-justified in this register.

在同一个函数中实现数据的发送与接收

6、测试：

在 dos 端进行 ip 地址的 ping 操作，若 ping 通了 则代表 spi 通信成功，反之则失败，要对程序端进行检查。

三、常见问题及注意事项

1、通过仿真器连接 pc 机与板子间的通信时，先创建一个.ccmxl 文件（用于连接 pc 机与板子），选择相应的仿真器与板子的型号-->设置名字，勾选选项-->save-->Test Connection--> 最后显示 success 表示连接成功

2、SPI 通信不成功：首先检查 DSPF28335 与 W5500 连接是否正确、引脚是否接对，其次检查 SPI、GPIO 配置是否正确（可以参考数据手册）

3、若在运行时程序卡在数据接收的循环
while(SpiaRegs.SPISTS.bit.INT_FLAG != 1);{}中不动时，应该检查在程序中是否使用了 fifo 函数，若使用则需要在使用后关闭（可以不使用 fifo）

4、因为在本次例程中把 dspf28335 设置成在读取数据时，每次读 8 位，但发送的数据一共有 16 位，若读只读前 8 位，但有效数据在后 8 位，所以在写数据的时候要向左移 8 位（SpiaRegs.SPITXBUF = (byte << 8);），以便都到有效数据。

5、发送和接收应该写在一个函数中，不能写在两个函数中

6、SPI 通信成功后，如果在发送接收时有乱码出现，是因为在 recv_data_prccessing()和 send_data_prccessing()中的 ptr（存储偏移地址的变量），一开始是 16 位的，因为移位溢出，可以设置成 32 位，或者在移位时，进行强制类型转换（ptr =>uint32）.PTR 是存储偏移地址的变量