W5500

**SPI Frame**

W5500 SPI Frame consists of 16bits Offset Address in Address Phase, 8bits Control Phase and N bytes Data Phase as shown in Figure 7.  
The 8bits Control Phase is reconfigured with Block Select bits (BSB[4:0]), Read/Write Access Mode bit (RWB) and SPI Operation Mode (OM[1:0]).  
Block Select bits select the block to which the Offset Address belongs.

Imagen en blanco y negro

El contenido generado por IA puede ser incorrecto.

**Address Phase**

Addresses are the same for different registers like Common and Socket, the difference of this addresse is that you can select which block to access selecting BSB4:BSB0 in the Control Phase. This is the first differencia with W5100 Chip.

**Control Phase**

Tabla

El contenido generado por IA puede ser incorrecto.The Control Phase specifies the Block to which the Offset Address (set by Address Phase) belongs, the Read/Write Access Mode and the SPI Operation Mode.  
  
Tabla

El contenido generado por IA puede ser incorrecto.

Usaremos BSB4:BSB0 para las direcciones, RWB para lectura o escritura, OM1:OM0 siempre será 0 porque usamos datos variables.

**1 Byte WRITE Access Example**

When the Host writes Data 0xAA to ‘Socket Interrupt Mask Register (SIMR) of Common  
Register Block by using VDM mode, the data is written with the SPI Frame below.  
  
The External Host asserts (High-to-Low) SCSn before transmitting SPI Frame, then the Host transmits 1 bit with synchronizing the Toggle SCLK. The External Host de-asserts (Low-to-High) the SCSn at the end of SPI Frame transmit. (Refer to the Figure 9)

Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

**1 Byte READ Access Example**

When the Host reads the ‘Socket Status Register(S7\_SR) of the Socket 7’s Register Block by using VDM mode, the data is read with the SPI Frame below. Let’s S7\_SR to ‘SOCK\_ESTABLISHED (0x17)’.  
The External Host asserts (High-to-Low) SCSn signal before transmitting SPI Frame, then the Host transmits Address and Control Phase to W5500 through the MOSI signal.   
Then the Host receives Data Phase from the MISO signal. After finishing the Data Phase receives, the Host de-asserts SCSn signal (Low-to-High). (Refer to the Figure 12)

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

**Register and Memory Organization**

W5500 has one Common Register Block, eight Socket Register Blocks, and TX/RX Buffer Blocks allocated to each Socket. Each block is selected by the BSB[4:0](BlockSelect Bit) of SPI Frame.

Figure 20 shows the selected block by the BSB[4:0] and the available offset address range of Socket TX/RX Buffer Blocks. Each Socket’s TX Buffer Block exists in one 16KB TX memory physically and is initially allocated with 2KB.

Also, Each Socket’s RX Buffer Block exists in one 16KB RX Memory physically and is initially allocated with 2KB.  
Regardless of the allocated size of each Socket TX/RX Buffer, it can be accessible within the 16 bits offset address range (From 0x0000 to 0xFFFF).  
  
Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.

**Hardware Initialization**

uint8\_t eth\_init(**struct** W5500\_SPI \* ETH)

{

SPI\_ETH\_RESET(ETH); //Reset W5500 por hardware  
 HAL\_Delay(800);

ETH->T8=0x00  
 SPI\_ETH\_REG(ETH, IMR,COMM\_REG ,SPI\_WRITE, ETH->T8,1);  
 ITM0\_Write("\r\nETH-W5500-INTERRUPT MASK CLEARED\r\n",strlen("\r\nETH-W5500-INTERRUPT MASK CLEARED\r\n"));  
ETH->T8=0x0F;  
 SPI\_ETH\_REG(ETH, RTR,COMM\_REG ,SPI\_WRITE, ETH->T8,1);  
 ETH->T8=0xA0;  
 SPI\_ETH\_REG(ETH, RTR+1,COMM\_REG,SPI\_WRITE, ETH->T8,1);  
 ITM0\_Write("\r\nETH-W5500-RETRY TIME SET\r\n",strlen("\r\nETH-W5500-RETRY TIME SET\r\n"));  
 ETH->T8=0x07;  
 SPI\_ETH\_REG(ETH, RCR,COMM\_REG ,SPI\_WRITE, ETH->T8,1);  
 ITM0\_Write("\r\nETH-W5500-RETRY COUNT SET\r\n",strlen("\r\nETH-W5500-RETRY COUNT SET\r\n"));  
 ETH->T8=0x00;S  
PI\_ETH\_REG(ETH, SIMR,COMM\_REG ,SPI\_WRITE, ETH->T8,1);  
 ITM0\_Write("\r\nETH-W5500-SOCKET INTERRUPT MASK CLEARED\r\n",strlen("\r\nETH-W5500-SOCKET INTERRUPT MASK CLEARED\r\n"));  
 SPI\_ETH\_REG(ETH, SHAR,COMM\_REG,SPI\_WRITE, ETH->SHAR,6);  
 ITM0\_Write("\r\nETH-W5500-MAC SET\r\n",strlen("\r\nETH-W5500-MAC SET"));  
 SPI\_ETH\_REG(ETH, GAR,COMM\_REG,SPI\_WRITE, ETH->GAR,4); //SPI\_ETH\_REG(ETH, GAR\_ADDR\_BASEH,GAR\_ADDR\_BASEL,SPI\_WRITE, ETH->GAR,4); //same for server and client  
ITM0\_Write("\r\nETH-W5500-GATEWAY SET\r\n",strlen("\r\nETH-W5500-GATEWAY SET\r\n"));   
SPI\_ETH\_REG(ETH, SUBR,COMM\_REG,SPI\_WRITE, ETH->SUBR,4);  
M0\_Write("\r\nETH-W5500-SUBNET SET\r\n",strlen("\r\nETH-W5500-SUBNET SET"));  
SPI\_ETH\_REG(ETH, SIPR,COMM\_REG,SPI\_WRITE, ETH->SIPR,4);  
 ITM0\_Write("\r\nETH-W5500-IP SET\r\n",strlen("\r\nETH-W5500-IP SET"))

}

**Socket Initalization**

uint8\_t eth\_socket\_init(**struct** W5500\_SPI \* ETH, uint8\_t socket)

{

ETH->T8=0x02;

SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,socket,SPI\_WRITE, ETH->T8,1);  
 SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,socket,SPI\_WRITE, ETH->T8,1);  
 SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S1\_REG,SPI\_WRITE, ETH->T8,1);  
SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S1\_REG,SPI\_WRITE, ETH->T8,1);  
 SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S2\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S2\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S3\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S3\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S4\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S4\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S5\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S5\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S6\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S6\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_RXBUF\_SIZE,S7\_REG,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TXBUF\_SIZE,S7\_REG,SPI\_WRITE, ETH->T8,1);

ITM0\_Write("\r\nETH-W5500-SOCKETS BUFFERS TX-RX SET\r\n",strlen("\r\nETH-W5500-SOCKETS BUFFERS TX-RX SET\r\n"));

ETH->T8=0x00;

SPI\_ETH\_REG(ETH, S\_TX\_WR,socket,SPI\_WRITE, ETH->T8,1);

SPI\_ETH\_REG(ETH, S\_TX\_WR+1,socket,SPI\_WRITE, ETH->T8,1);

ITM0\_Write("\r\nETH-W5500-SOCKET0 WR POINTER CLEAR\r\n",strlen("\r\nETH-W5500-SOCKET0 WR POINTER CLEAR\r\n"));  
 SPI\_ETH\_REG(ETH, S\_PORT, socket,SPI\_WRITE, ETH->S\_PORT,2);  
 ITM0\_Write("\r\nETH-W5500-SOCK0 SOURCE PORT SET\r\n",strlen("\r\nETH-W5500-SOCK0 SOURCE PORT SET\r\n"));  
 SPI\_ETH\_REG(ETH, S\_DIPR,socket,SPI\_WRITE, ETH->S\_DIPR,4);  
 ITM0\_Write("\r\nETH-W5500-SOCK0 DESTINTAION IP ADDRESS SET\r\n",strlen("\r\nETH-W5100-SOCK0 DESTINTAION IP ADDRESS SET\r\n"));  
 SPI\_ETH\_REG(ETH, S\_DPORT,socket,SPI\_WRITE, ETH->S\_DPORT,2);  
 ITM0\_Write("\r\nETH-W5500-SOCK0 DESTINTAION PORT SET\r\n",strlen("\r\nETH-W5100-SOCK0 DESTINTAION PORT SET\r\n"));  
eth\_wr\_SOCKET\_MODE(ETH,socket, MODE\_TCP);  
TM0\_Write("\r\nETH-W5500-SOCK0 TCP MODE SET\r\n",strlen("\r\nETH-W5100-SOCK0 TCP MODE SET\r\n"));  
eth\_wr\_SOCKET\_CMD(ETH,socket, OPEN);  
 ITM0\_Write("\r\nETH-W5500-OPEN SOCKET\r\n",strlen("\r\nETH-W5100-OPEN SOCKET\r\n"));  
**f**(ETH->S\_ENserver == 1)  
 { eth\_wr\_SOCKET\_CMD(ETH,socket, LISTEN);  
 ITM0\_Write("\r\nETH-W5500-LISTEN SOCKET\r\n",strlen("\r\nETH-W5100-LISTEN SOCKET\r\n"));  
 }  
 **else** {  
 eth\_wr\_SOCKET\_CMD(ETH,socket, CONNECT);   
ITM0\_Write("\r\nETH-W5500-CONNECT\r\n",strlen("\r\nETH-W5100-CONNECT\r\n"));  
 }

**Physical layer Interrupts**

**IR (Interrupt Register) [R/W] [0x0015] [0x00]**

IR indicates the interrupt status. Each bit of IR can be cleared when the host writes ‘1’ value to each bit. If IR is not equal to ‘0x00’, INTn PIN is asserted low until it is ‘0x00’.

Tabla

El contenido generado por IA puede ser incorrecto.

**IMR (Interrupt Mask Register) [R/W][0x0016][0x00]**

IMR is used to mask interrupts. Each bit of IMR corresponds to each bit of IR. When bit of IMR is ‘1’ and the corresponding bit of IR is ‘1’, an interrupt will be issued. In other words, if a bit of IMR is ‘0’, an interrupt will not be issued even if the corresponding bit of IR is ‘1’.

Tabla

El contenido generado por IA puede ser incorrecto.

**SIR (Socket Interrupt Register) [R/W] [0x0017] [0x00]**

SIR indicates the interrupt status of Socket. Each bit of SIR be still ‘1’ until Sn\_IR is cleared by the host. If Sn\_IR is not equal to ‘0x00’, the n-th bit of SIR is ‘1’ and INTn PIN is asserted until SIR is ‘0x00’.

Tabla

El contenido generado por IA puede ser incorrecto.

**SIMR (Socket Interrupt Mask Register) [R/W] [0x0018] [0x00]**

Each bit of SIMR corresponds to each bit of SIR. When a bit of SIMR is ‘1’ and the

corresponding bit of SIR is ‘1’, Interrupt will be issued. In other words, if a bit of SIMR

is ‘0’, an interrupt will be not issued even if the corresponding bit of SIR is ‘1’.

Tabla

El contenido generado por IA puede ser incorrecto.

**Sn\_IR (Socket n Interrupt Register) [RCW1] [0x0002] [0x00]**

Sn\_IR indicates the status of Socket Interrupt such as establishment, termination, receiving data, timeout). When an interrupt occurs and the corresponding bit of Sn\_IMR is ‘1’, the corresponding bit of Sn\_IR becomes ‘1’.   
n order to clear the Sn\_IR bit, the host should write the bit to ‘1’.  
  
Tabla

El contenido generado por IA puede ser incorrecto.