W5100 Porting Guide Ver. 1.0

WIZnet default driver source is based on Atmega128. Therefore, in case of porting with Atmega128, there is no need to modify existing driver source. However, if you port W5100 in a system using other MCU, you should consider following points.

1. MCU dependant part

- **Big-endian or Little-endian**
  The default byte ordering is 'Little-endian' in sample source code. If you want to change it to 'Big-endian', you should change the 'SYSTEM_ENDIAN' value into '_ENDIAN_BIG_' as below.

<Refer to /mcu/type.h>

```
#define     _ENDIAN_LITTLE_         0
#define     _ENDIAN_BIG_            1
#define     SYSTEM_ENDIAN           _ENDIAN_BIG_
```

- **Base address of W5100**
  Define W5100's base address as '__DEF_IINCHIP_MAP_BASE__'.

```
#define __DEF_IINCHIP_MAP_BASE__        0x8000
```

If its base address is more than 0xFFFF, following parts should be changed.

<Refer to /iinchip/w5100.c>

```
static u32 SBUFBASEADDRESS[MAX_SOCK_NUM];
static u32 RBUFBASEADDRESS[MAX_SOCK_NUM];
u32 getIINCHIP_RxBASE(u8 s)
u32 getIINCHIP_TxBASE(u8 s)

u8 IINCHIP_WRITE(u32 addr, u8 data)
u8 IINCHIP_READ(u32 addr)
```

- **Interrupt service routine**
  When you develop an application using W5100, you can use the interrupt-driven mode or polled mode. If you build F/W using the interrupt-driven mode, you should define interrupt enable/disable.

<Refer to /mcu/type.h>

```
#define IINCHIP_ISR_DISABLE()           (EIMSK &= ~(0x10))
#define IINCHIP_ISR_ENABLE()            (EIMSK |= 0x10)
```

2. W5100 interface

The value of '__DEF_IINCHIP_BUS__' should be changed based on the way to connect W5100 to MCU. You should choose a mode among direct mode, indirect mode or SPI mode based on implemented H/W interface.

<Refer to /mcu/type.h>

```
#define __DEF_IINCHIP_DIRECT_MODE__     1
#define __DEF_IINCHIP_INDIRECT_MODE__   2
#define __DEF_IINCHIP_SPI_MODE__        3
#define __DEF_IINCHIP_BUS__             __DEF_IINCHIP_DIRECT_MODE__
```


3. W5100 initialization

1) Initialize W5100 with S/W reset.
   MR [0x0000] ← 0x80;

2) Set up the network configuration.

   Basic network configurations to be configured for networking are as below.
   - Gateway Address Register
   - Source Hardware Address Register
   - Subnet Mask Register
   - Source IP Address Register

   ex) Gateway address set up 192.168.0.1
   GWR [0x0001 ~ 0x0004] ← [0xC0, 0xA8, 0x00, 0x01];

   Likewise, set up Subnet Mask Register, Source IP Address Register, and Source Hardware Address Register.

3) In case of using Interrupt-driven mode, set up Interrupt Mask Register
   IMR ← 0xEF;

4) Set up RX Memory Size Register and TX Memory Size Register.
   You can allocate the amount of memory to be used for each channel.
   If you allocate 2KBytes on each channel, you should write '0x55' in RMSR and TMSR as below.
   RMSR ← 0x55;
   TMSR ← 0x55;

5) If you finish 4) step, you can get ICMP reply message from W5100. It means, if you execute "ping" in your PC with the destination IP address of W5100, you can get reply.

If you want to know details, please refer to W5100 datasheet.

4. Check Points after Driver porting

- Link LED
  When LAN cable is connected, is Link LED on?
  W5100's Link LED should be ON in connected condition.
  During transmitting packets, W5100's Link LED is flickering.
  If you have something wrong in Link LED, you should check H/W circuit again.

- Register check
  Check whether W5100's registers are able to be written and read correctly?
  For example, after you write '0xAA' in the register for Gateway Address, GAR0, '0xAA' should be able to read from there at reloading.
  You can confirm the interface design between MCU and W5100 through this test.

- Ping test
  Do ping test in PC using IP address of W5100.
  Through this test, you can check basic operation and H/W condition of W5100.
  If you have something wrong in ping operation, you should check network setup information and connecting condition.

- Data transfer test using Simple TCP loop-back program
  You can confirm TCP communication function of your system with making a TCP connection and simple test related to data transmission.
  You can download a program for a loop-back test from www.wiznet.co.kr.

+ Explanation for Function

WIZnet provides the FW for products. The socket interface provided by F/W is as below.
If you want to know more details, please refer to W5100 source code and datasheet.

uint8 **socket**(SOCKET s, uint8 protocol, uint16 port, uint8 flag)
- This function is used to initialize socket channels.
- Parameter
  . SOCKET s: Socket index (0-3) to be connected
  . uint8 protocol: Socket Protocol
  Ex) Sn_MR_TCP, Sn_MR_UDP, Sn_MR_IPRAW, Sn_MR_MACRAW, Sn_MR_PPPOE
  . uint16 port: Socket Port number to be opened.
  If you write '0(zero)', the temporary port defined in source will be allocated.
  . uint8 flag: Set up the Socket option.
  Ex) Sn_MR_ND, Sn_MR_MULTI
- Return: If you succeed, the return value is '1'. But if not, '0(zero)' is returned.

* If you open socket without No Delay ACK ("Sn_MR_ND" setup for flag), an ACK message does not be sent as soon as getting receiving data. A little time delay can be happened and the ACK message will be sent since getting next data

void close(SOCKET s)

- This function is used to disconnect a socket compulsorily.
- Parameter
  . SOCKET s: Socket index
- Return: None.

* Attention: If you disconnect a socket using 'close()', a FIN message won't be sent to the opposite system.

uint8 listen(SOCKET s)

- This function is used to make a socket operate as a TCP server. The socket will wait for the peer's request.
- Parameter
  . SOCKET s: Socket index
- Return: If you succeed, the return value is '1'. But if not, '0(zero)' is returned.

uint8 connect(SOCKET s, uint8 * addr, uint16 port)

- This function is used to make a socket operate as a TCP client. The socket will try to connect with server.
- Parameter
  . SOCKET s: Socket index
  . uint8 * addr: Server IP address. (i.e. "192.168.11.8" => addr = {C0,A8,0B,08})
  . uint16 port: Server port. (i.e. 5000 => 0x1388)
- Return: If you succeed, the return value is '1'. But if not, '0(zero)' is returned.

void disconnect(SOCKET s)

- This function is used to make a connected socket disconnected in TCP mode. Disconnecting process will start with sending a FIN message.
- Parameter
  . SOCKET s: Socket index
- Return: None.

* If a connection does not be disconnected with 'disconnect()', you can disconnect compulsorily with 'close()' function.

uint16 send(SOCKET s, const uint8 * buf, uint16 len)

- This function is used to transmit data to destination system in TCP mode (Sn_MR_TCP).
- Parameter
  . SOCKET s: Socket index
  . const uint8 * buf: Data pointer to send.
  . uint16 len: Data length to send.
- Return: If you succeed, the return value is '1'. But if not, '0(zero)' is returned.

uint16 **recv**(SOCKET s, uint8 * buf, uint16 len)

- This function is used to receive data from the opposite system in TCP mode (Sn_MR_TCP).
- Parameters
  - . SOCKET s: Socket index
  - . uint8 * buf: Buffer pointer to receive.
  - . uint16 len: Data length to receive
- Return: Message length to be received.

* This function does not wait until data receipt is completed unlike general BSD socket. After checking the existence and the length of received data, recv() is called to get data.

uint16 **sendto**(SOCKET s, const uint8 * buf, uint16 len, uint8 * addr, uint16 port)

- Object: This function is used to transmit data to destination in UDP mode (Sn_MR_UDP).
- Parameters
  - . SOCKET s: Socket index
  - . const uint8 * buf: Data pointer to send.
  - . uint16 len: Data length to be sent
  - . uint8 * addr: Destination IP Address.
  - . uint16 port: Destination port number
- Return: None.

uint16 **recvfrom**(SOCKET s, const uint8 * buf, uint16 len, uint8 * addr, uint16 *port)

- This function is used to receive data from the opposite system in UDP mode (Sn_MR_UDP).
- Parameters
  - . SOCKET s: Socket index
  - . const uint8 * buf: Buffer pointer to receive.
  - . uint16 len: Data length to be received
  - . uint8 * addr: Sender's IP Address.
  - . uint16 * port: Sender's Port number
- Return: Message length to be received

* This function is also used for IPRAW and MACRAW mode as well as UDP mode. Sn_MR_MACRAW mode is only enabled in socket '0(zero)'. If you want more details, refer to W5100 datasheet.