Driver

The ioLibrary means **“Internet Offload Library”** for WIZnet chip. It includes **drivers** and **application protocols**. There are three kinds of libraries explained on this page The first two drivers (ioLibrary_BSD, ioLibrary) can be used for W5500 application designs. These will be updated continuously. The former BSD-Type driver will not be updated, as it is only meant to be a migration help from W5200 to W5500.

---

1. ioLibrary_BSD

Overview

This driver provides the Berkeley Socket type APIs. The function names of this ioLibrary_BSD are the same as the function names of the ioLibrary.

- Directory Structure
  - Ethernet : SOCKET APIs like BSD & WIZCHIP(W5500,W5200 and etc) Driver
  - Internet :
    - DHCP client
    - DNS client
    - Others will be added.

Download

< ioLibrary_BSD : latest version >

	Type	Version	Note	Download Link
Source code	Ethernet (Berkeley Socket type APIs)	1.0.3	-	Click
	Internet (Application protocols)	1.1.1	-	Click
Documents	Socket APIs Help (chm, html)	1.0.3	-	Click

< ioLibrary_BSD : old version >

	Type	Version	Note	Download Link

Source code	Ethernet (Berkeley Socket type APIs)	1.0.2	-	Click
	Ethernet (Berkeley Socket type APIs)	1.0.1	-	Click
		1.0.0	-	Click
	Internet (Application protocols)	1.1.0	-	Click
		1.0.0	-	Click
Documents	Socket APIs Help (chm, html)	1.0.2	-	Click
	Socket APIs Help (chm, html)	1.0.1	-	Click
		1.0.0	-	Click

< Update History >

- ioLibrary_BSD
  - Ethernet : Berkeley Socket type APIs
    - Document (chm, html): Socket APIs Help
    - Revision History
      - V1.03 (Refer to 20140501)
        - wizchip_conf.c
          1. ***Explicit type casting in wizchip_bus_readbyte() & wizchip_bus_writebyte()***
          2. uint32_t type converts into ptrdiff_t first. And then reconverting it into uint8_t*. For remove the warning when pointer type size is not 32bit. ***If ptrdiff_t doesn't support in your complier, You should must replace ptrdiff_t into your suitable pointer type.***
        - w5500.c : ***Implicit type casting → Explicit type casting***
          1. wizchip_read_data() & wizchip_write_data() : Fixed the problem on porting into under 32bit MCU
        - socket.h
          1. Modify the comment : SO_REMAINED → PACK_REMAINED
          2. Add the comment as zero byte udp data reception in getsockopt().
        - socket.c
          1. ***Implicit type casting → Explicit type casting.***
          2. replace 0x01 with PACK_REMAINED in recvfrom()
          3. Validation a destination ip in connect() & sendto(): It occurs a fatal error on converting unint32 address if uint8* addr parameter is not aligned by 4byte address. Copy 4 byte addr value into temporary uint32 variable and then compares it.

- V1.02
- socket.c (Refer to 20131220)
  1. setsockopt() : Remove warning message (delete tmp variable)
- w5500.c (Refer to 20131220)
  1. WIZCHIP_READ_BUF() & WIZCHIP_WRITE_BUF() in _WIZCHIP_IO_MODE_SPI_FDM_ case
     1. Remove warning message
     2. Remove unnecessary 'for' loop
- V1.01
- socket.c (Refer to 20131104)
  1. sendto() : Add to clear the timeout interrupt status of socket(Sn_IR_TIMEOUT).
- V1.00
- First released.
- Internet : Application protocols
- Revision History

- V1.11 (2013-12-26)
- DHCP Client
  1. Modify variable declaration(dhcp_tick_1s) for code optimization in dhcp.c
- V1.10
- DHCP Client
  1. Optimize code
  2. Add reg_dhcp_cbfunc()
  3. Add DHCP_stop()
  4. Integrate check_DHCP_state() & DHCP_run() into DHCP_run()
  5. Don't care system endian
  6. Move unreferenced DEFINE to dns.c
  7. Remove the unused DEFINE
  8. Add comments
- DNS Client
  1. Remove secondary DNS server in DNS_run
     1. If 1st DNS_run failed, call DNS_run with 2nd DNS again
  2. DNS_timerHandler → DNS_time_handler
  3. Move unreferenced DEFINE to dns.c
  4. Remove the unused define
  5. Integrated dns.h dns.c & dns_parse.h dns_parse.c into dns.h & dns.c
- V1.00
- First released.
  - DHCP Client (Dynamic Host Configuration Protocol Client)
  - DNS Client (Domain Name System Client)

< Application code examples : latest version >

	Application	Update	Note	Download Link
STM32F103X CooCox CoIDE Project	Loopback Test	2013-11-04	-	Click
	DHCP Client	2013-12-26	-	Click
	DNS Client	2013-12-26	-	Click
EnergyMicro Tiny GECKO(EFM32TG840F32) IAR Project	Loopback Test DHCP Client DNS Client	2013-12-20	-	Click

< Application code examples : old version >

	Application	Update	Note	Download Link
STM32F103X CooCox CoIDE Project	Loopback Test	2013-10-23	-	Click
	DHCP Client	2013-11-08	-	Click
	DNS Client	2013-11-08	-	Click

These projects do not contain [Ethernet] and [Internet] codes. (Empty directory)

Please download ioLibrary_BSD APIs and Application protocols,
and then insert to each of same named directory in provided project.

< History >

- Application code example
  - Example project was made by CooCox CoIDE with the STM32F103X Cortex-M3 platform.
    - Loopback Test
      - 2013-11-04 Changes
        1. main.c : refine and rearrange source code.
        2. Separated Project code / APIs and Applications
    - DHCP Client
      - 2013-12-26 Changes
        1. main.c : refine and rearrange source code for improved DHCP code.
    - DNS Client
      - 2013-12-26 Changes
        1. main.c : refine and rearrange source code for improved DNS code

Description

This driver provides BSD-type Socket APIs for W5500. Because the function names of this driver are more user-friendly than those of the older drivers, …, current WIZnet chip users can easily migrate from their WIZnet chip application to the W5500 application. All drivers for W5100, W5200 and W5300 will be merged into the ioLibrary in the near future. All application protocols will also be merged into ioLibrary based on this BSD-type Socket APIs.

This table shows the differences between other BSD drivers and new W5500 driver.

Driver	Other BSD Drivers	W5500 Driver
Variables Type	type.h (made by wiznet) ex) uint16	stdint.h (standard) ex) uint16_t
Register Naming	REGName + Index ex) SIPR0 , SIPR1, SIPR2, SIPR3	REGName & WIZCHIP_OFFSET_INC ex> SIP, WIZCHIP_OFFSET_INC(SIP,1), WIZCHIP_OFFSET_INC(SIP,2), WIZCHIP_OFFSET_INC(SIP,3)
Basic IO function	IINCHIP_READ IINCHIP_WRITE IINCHIP_READ_BUF IINCHIP_WRITE_BUF 16bit Address Space User should implement Functions MCU Dependent	WIZCHIP_READ WIZCHIP_WRITE WIZCHIP_READ_BUF WIZCHIP_WRITE_BUF 32bit Address Space Since users implement MCU-dependent parts and then register them as Callback function, users don't need to implement the Function itself. Supports IINCHIP_XXX function for backward compatiblity.
Register Function	IINCHIP_XXX can be used. Supports some getREG() & setREG() functions.	It is not recommended to use WIZCHIP_XXXX. Supports getREG() & setREG() functions or macros for all registers.

Extra Functions	None	Optional and omissible Supports User-friendly named functions. All extra functions can be implemented by using setREG() & getREG().

Socket APIs	**Other BSD Drivers**	**W5500 Driver**
Return Value	void Success or Fail Transmit/Receive Size	Success or Fail Transmit/Receive Size All functions return.
Error Code	None	SOCK_BUSY : 0 SOCKERR_SOCKNUM SOCKERR_SOCKOPT SOCKERR_SOCKINIT SOCKERR_SOCKCLOSED SOCKERR_SOCKMODE SOCKERR_SOCKFLAG SOCKERR_SOCKSTATUS SOCKERR_ARG SOCKERR_PORTZERO SOCKERR_IPINVALID SOCKERR_TIMEOUT SOCKERR_DATALEN SOCKERR_BUFFER SOCKFATAL_PACKLEN
IO Mode	Block & Non-Block Fixed	Block or Non-Block configurable
Block Function	send recv sendto recvfrom	send recv sendto recvfrom connect
Non-Block Function	connect	
recvfrom	Should read data in received packet unit.	Can read data received packet separately.

- Socket APIs
  - Function Name
    - Same as the function name of previous drivers
  - Function Return value
    - Previous Drivers: Void or Success/Fail and Transmit/Receive Size
    - W5500 Driver: All functions return Success and Fail. In Fail case, operations are subdivided.
      - Success: SOCK_OK, Socket Number, Transmit and Receive Size
      - Fail: SOCK_BUSY, SOCKERR_XXX, SOCKFATAL_XXX (0 or Negative value)
        1. **SOCK_BUSY** : 0
        2. SOCKERR_SOCKNUM
        3. SOCKERR_SOCKOPT
        4. SOCKERR_SOCKINIT
        5. SOCKERR_SOCKCLOSED
        6. SOCKERR_SOCKMODE
        7. SOCKERR_SOCKFLAG
        8. SOCKERR_SOCKSTATUS
        9. SOCKERR_ARG
        10. SOCKERR_PORTZERO
        11. SOCKERR_IPINVALID
        12. SOCKERR_TIMEOUT
        13. SOCKERR_DATALEN

14. SOCKERR_BUFFER
15. **SOCKFATAL_PACKLEN**

- Block / Non-Block IO mode
  - Previous Drivers : Block function and Non-Block function were mixed.
    - Block Function : send(), recv(), sento(), recvfrom()
    - Non-block Function : connect()
    - Blocking can be avoided by using getSn_SR(), getSn_TX_FSR(), and getSn_RX_RSR() properly.
  - W5500 Driver
    - Block / Non-Block IO mode can be selected by user. (Default: Block mode)
    - socket() with new flag SF_IO_NONBLOCK or setsockopt() with SO_SET_IOMODE Can be configured.
    - Block and Non-block Configurable Function
    - connect(), send(), recv(), sendto(), recvfrom()
    - **getSn_SR(), getSn_TX_FSR() and getSn_RX_RSR() functions can be used like … like previous drivers. They are not related to IO mode**

---

2. ioLibrary

Download

< ioLibrary with example project : latest version >

	Application	Version	Note	Download Link
Cookie board	Loopback test	1.0.2	-	Click

<Revision History>

- v102
  - socket.c(Refer to 2014-03-18)
    1. TCPReSend() : Remove this function and related codes because TCP send mechanism was changed.
    2. TCPReSendNB() : Remove this function and related codes because TCP send mechanism was changed.
    3. TCPSendCHK() : Modify return value.
    4. TCPSend() : Change return value to len.
  - loopback.c(Refer to 2014-03-18)
    1. Existing mechanism resend packet if don't send all received packet, but change not to resend.
- v100
  - First release

< ioLibrary : latest version >

	Description	Version	Note	Download Link
Driver Source code	ioLibrary source code	1.0.2	-	Click

< ioLibrary : old version >

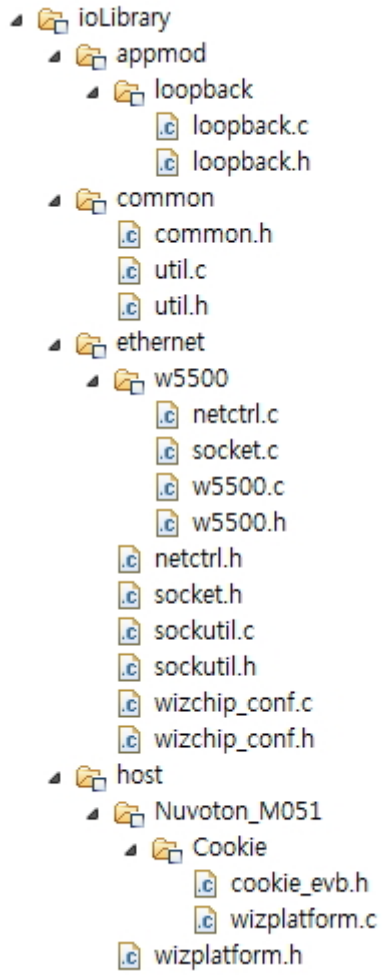
	Description	Version	Note	Download Link
Driver Source code	ioLibrary source code	1.0.0	-	Click
Driver documents	Socket APIs Help(chm, html) (To use html, open the index.html)	1.0.0	-	Click

This ioLibrary has basic I/O functions, socket register access functions, common register access functions, utilities and functions for setting up a platform and network This code has been evaluated on the CooCox Cookie Board with ARM Cortex-M0 MCU.

Please refer to this link for more details.

- How to use on cookie board.

The figure below shows the folder structure of this ioLibrary.



---

3. BSD Type driver for W5200 User

- Driver Source code : w5500_cortexm3_firmware_for_legacy.zip

This driver has the same BSD as the API for W5200 users. We have been evaluating this code on the **ARM-CortexM3(STM32F103 series)** chipset.

This type of driver is the final version. We will not update it later. Please use the new (well coded) driver code for new projects.

From:
http://wizwiki.net/wiki/ -

Permanent link:
http://wizwiki.net/wiki/doku.php?id=products:w5500:driver

Last update: **2014/08/06 14:07**