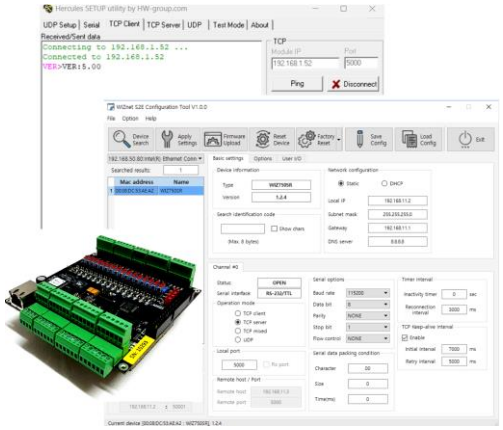
3el 2x16 IO Card Ethernet V4.1

2x16 IO Eth V4.1Programming Manual

Features:

- Communication over Ethernet
- TCP/UDP server
- Configurable IP address
- Human readable commands and messages
- Extension board support
- Grouped output control
- I2C, RS485 protocol translator



Preparing the software tools:

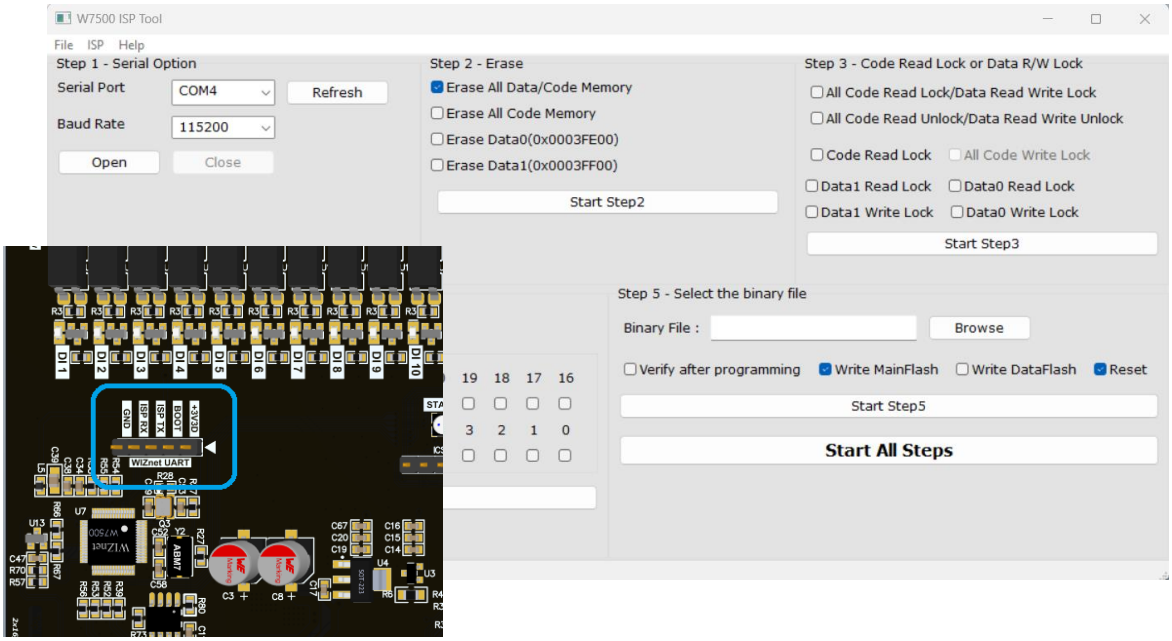
The Ethernet connection is implemented with a WIZnet W7500P-S2E IC. The related overview and datasheet are available on the following links:

W7500P overview
W7500P-S2E Datasheet [En]

Updating/flashing the application firmware on the W7500P can be done either through the ISP header, either with the help of a preloaded bootloader (if present in the chip's memory).

1. The ISP update is done through a WIZnet UART header on the 2x16IO Card PCB (highlighted with blue in the next picture) and the W7500 ISP Tool for flashing the W7500P-S2E. The binary file, the software tool and documentations can be downloaded from the following links:

Firmware Binary (WIZISP – boot and application firmware)V 1.3.3
W7500 ISP Tool
App. hints - How to use ISP Tool



2. A regular firmware update (when the bootloader is present in the chip's memory) is done by a special software tool provided by the WIZnet manufacturer. The binary file, the software tool and documentations can be downloaded from the following links:

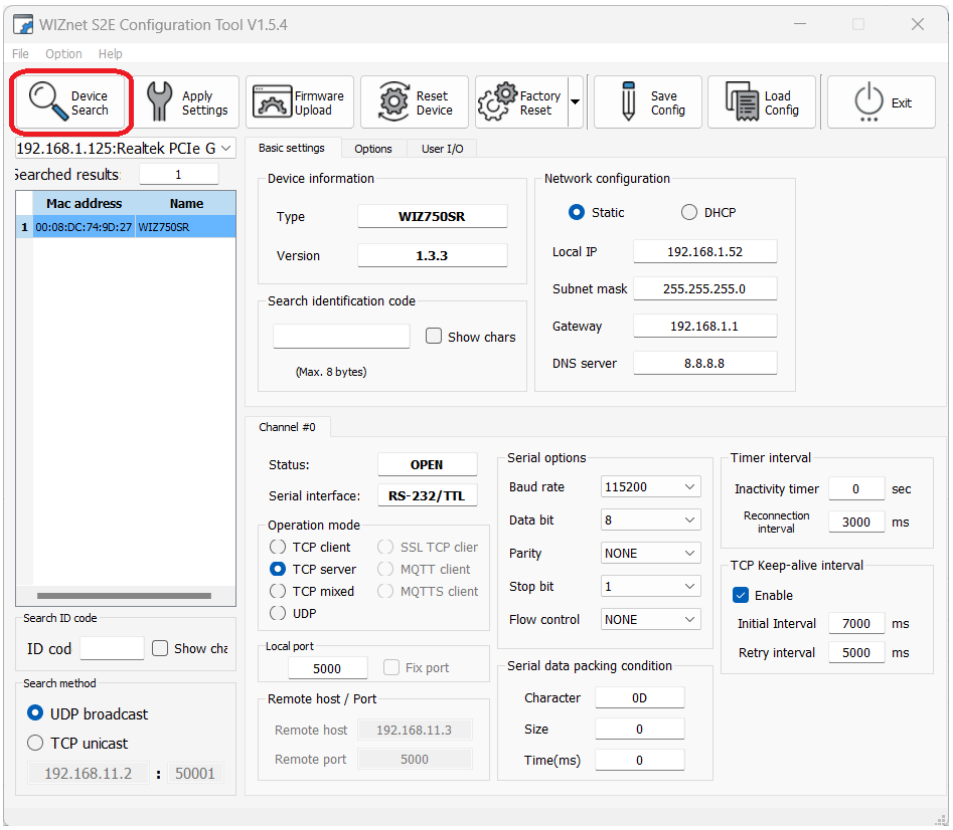
Firmware Binary (application firmware) V 1.3.3
WIZnet-S2E-Tool-GUI V 1.5.4
App. hints - How to use WIZnet S2E Tool
Commands for configuring the WIZnet circuit [En]

Configuring the 2x16IO Card:

The configuration tool software is used to configure the 2x16IO Card's TCPIP/UDP settings in four mandatory steps, followed by pressing the [Apply Settings] button on the GUI:

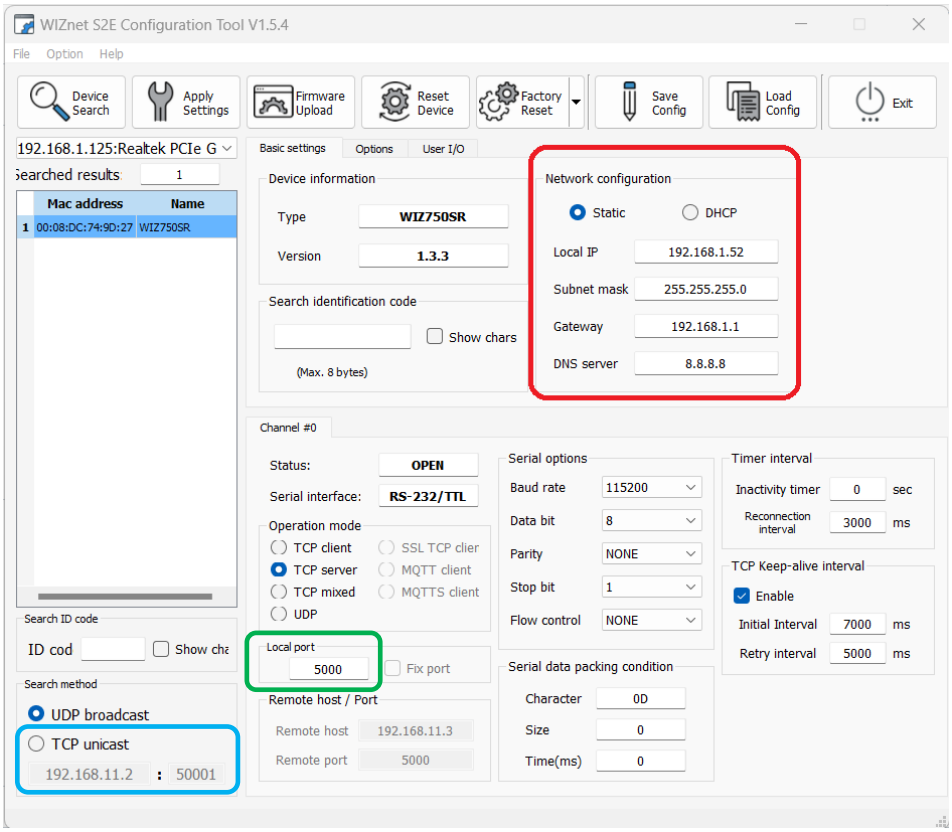
1. Once powered and plugged in to an Ethernet network a search is initiated for the 2x16IO Card on the network. This can be performed with the [Device Search] button on the config tool's GUI. Once the search is done the 2x16IO

Card's MAC address should appear in the [Search results] list of the tool.

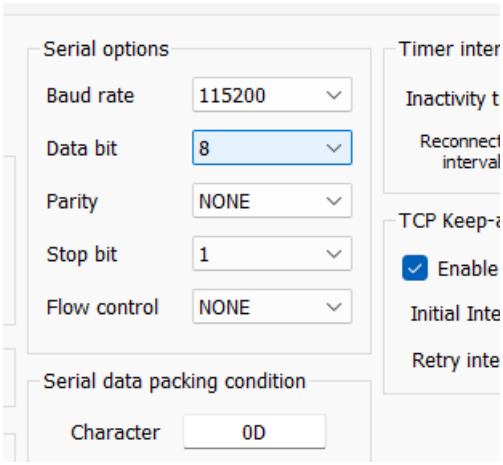


Note: for computers connected to more networks than one, this search might not work on all networks but rather the primary one only. In that case recommendation is to configure the unit on the primary network and then switch it to the network where it will be used.

2. TheWIZ7500P communication distinguishes two different TCP/IP port referenced to the same IP address (for example 192.168.1.52 marked with red in the following picture). The 50001 port is for configuration over TCP unicast frames (marked with blue in left lower corner of the following picture), while the 5000 port is for data communication (marked with green in the following picture).

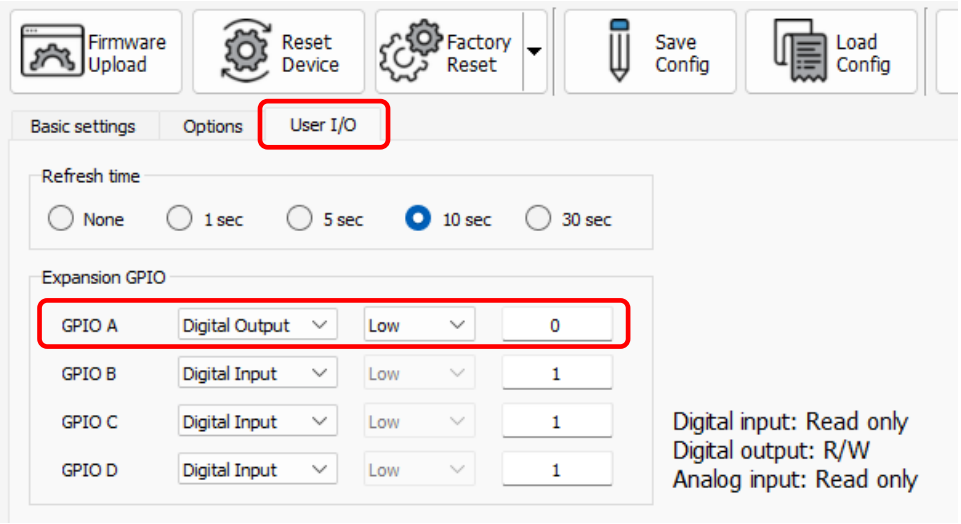


3. For the WIZ7500P and MCU communication the serial port parameters should be set to 115200baud with 8 data bits, without parity, using 1 STOP bit and without Flow Control.



4. The last mandatory setting is to configure the [Expansion GPIO] ports of the WIZ7500P to ensure that the PIC MCU is released from reset. For this [GPIO A] should be selected as [Digital Output] and set to [Low] value. As a new feature

the system MCU can be remotely reset through the Ethernet interface either using direct WIZnet commands sent to the configuration port (default at 50001) or by using the configuration tool GUI (WIZnet S2E Configuration Tool V 1.5.4). Resetting the MCU is done by setting GPIO A first to High than to Low level.



Note: As the last final step don't forget to press the [Apply Settings] button on the GUI.

Once the connection and hardware setup is done for the 2x16IO Card, the device will be available to respond over Ethernet network with using any TCP/IP (or UDP) tool able to transmit and receive data packages in the required format. For example, the Hercules software tool or Putty console are both handy tools for this purpose:

Hercules setup utility V 3.2.8
Putty console V 0.79

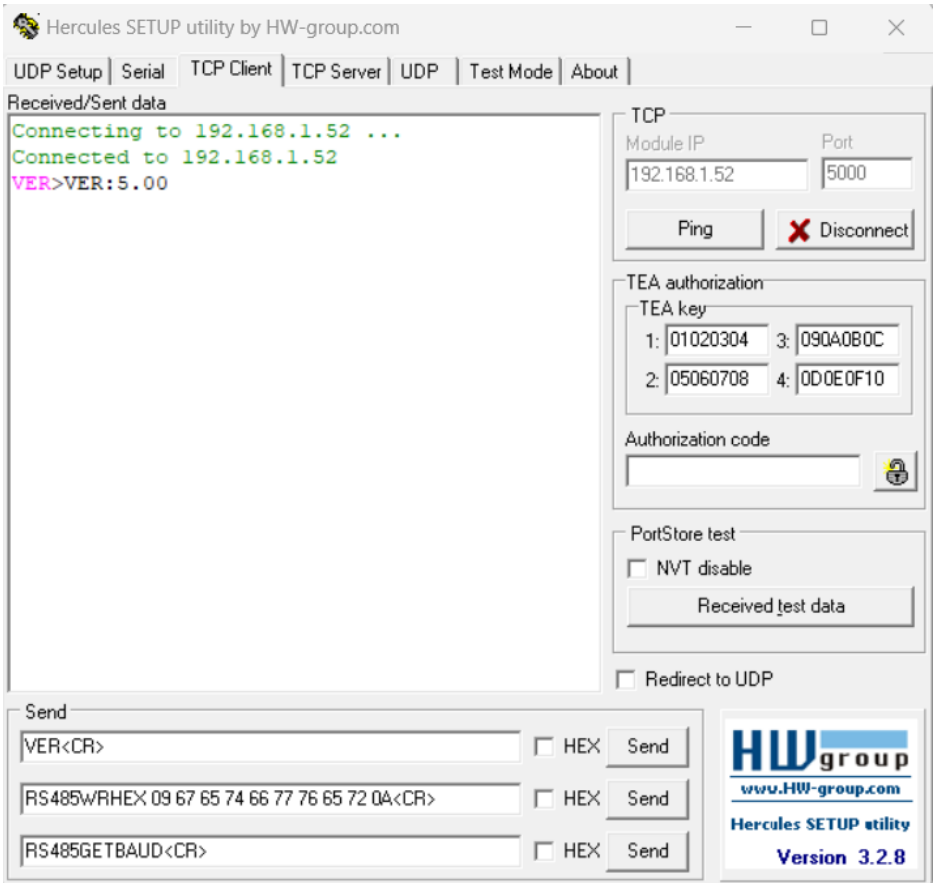
IO Card Protocol

The protocol is human readable, uses ASCII coded characters and each message ends with a Carriage Return (CR) character.

Every valid response starts with "**>**" character. If an error occurs or invalid command was entered the protocol returns a "**!**" character.

NOTE: some commands are firmware version dependent (for this you will see something like v5.0+ meaning that the command works from firmware 5.0 and up.

For connection, the previously configured IP address and Port number should be used. After connecting to the IO Card over TCP/IP and sending the basic **VER** command followed by a Carriage Return character (VER<CR>) a valid response should be got from the IO Card containing the firmware version.



Command	Description	
VER<CR>	Get firmware version	
Example:	Send	VER<CR>
	Receive	>VER:5.00
IN0<CR>	Get Input states Main Board	
Example:	Send	IN0<CR>
	Receive	>IN0:0000000000001000
IN1<CR>	Get Input states Extension Board 1	
Example:	Send	IN1<CR>
	Receive	>IN1:0000000000001000
IN2<CR>	Get Input states Extension Board 2	
Example:	Send	IN2<CR>
	Receive	>IN2:0000000000001000
OUTnn s<CR> nn - channel number(1..16) s - state (0,1)	Set ON/OFF digital output	
Example:	Send	OUT01 1<CR>
	Receive	>OUT01 1
PWRn s<CR> n - channel number(1..2) s - state (0,1)	Set ON/OFF High-Power output	
Example:	Send	PWR1 1<CR>
	Receive	>PWR1 1
INA<CR>	Get analog Data	
Example:	Send	INA<CR>
	Receive	>INA:1952 1955 1981 2007
IND<CR>	Get all digital input values at once from mainboard and from the two extension boards	
Example:	Send	IND <CR>
	Receive	>IND:0 32 32 32 0 01
Note:	Response: contains 6 decimal numbers (8 bits each) for the 16+16+16 inputs	

Command	Description	
	(the values should be the same reported by IN0, IN1, IN2)	
CLEAR<CR>	Clear all outputs	
Example:	Send	CLEAR<CR>
	Receive	> CLEAR
SETBYMASK 1234 5678 9ABC [FFFF FFFF FFFF] <CR>	Set all outputs at once (v4.2+)	
Example:	Send	SETBYMASK 10 10 10 10 10 10<CR>
	Receive	>SETBYMASK 0010 0010 0010
Note:	First 3 mandatory parameters: hexadecimal output values for a base card and two extensions (1st bit is OUT1). Second 3 optional parameters: masks considered 0xFFFF by default (permits setting only the selected bits in the masks instead of all bits). The response will contain the three output registers for card and two extensions after the input values were applied. All parameters and the response are in hexadecimal.	
GETOUT<CR>	Get all outputs at once (v4.2+)	
Example:	Send	GETOUT<CR>
	Receive	>GETOUT 0010 0010 0010
Note:	Response contains all outputs in one response (hexadecimal). The values are the last set ones, there is no HW support to measure back the real digital outputs.	
SETLATCH dd..d<CR>	Setting latches	

Command	Description	
dd..d - hexadecimal 64 bit number (with swapped LSB order comparing to IN0 - least significant 1st), therefore 1st byte is IN1-IN8, IN9-IN16, IN17-24, IN25-32 etc. - the number is 64byte to support 2nd extension too.		
Example for latching IN1:	Send	SETLATCH 0100000000000000<CR>
	Receive	>SETLATCH 0100000000000000
Note:	This command also resets current latches (used to reset the latched pins at the start of every sequence).	
I2CEXT [Fffff] [Waadddd...] [Raann]...<CR>	Generic write and Read I2C Port	
ffff – I2C clock frequency in kHz W - write, R - read, F – I2C clock frequency in kHz aa - slave I2C address dddd - data to write nn - number of bytes to read		
Example for reading 2 bytes from all I2C expanders present on the I2C input extension:	Send	I2CEXT W2000 R2002 W2100 R2102 W2200 R2202 W2300 R2302<CR>
	Receive	>I2CEXT W2000 R200000 W2100 R210000 W2200 R220000 W2300 R230000
Example for writing the I2C expanders present on the I2C output extension:	Send	I2CEXT W20020000 W21020000 W22020000 W23020000<CR>
	Receive	>I2CEXT W2000 R200000 W2100 R210000 W2200 R220000 W2300

Command	Description	
		R230000
Example for changing frequency and write two bytes to register 02:	Send	I2CEXT F100 W20020000<CR>
	Receive	>I2CEXT F03E W20020000
Note:	The F03E response for F100 contains the BRG register value (3E in this case). The value depends on MCU clock frequency (12.8MHz) ex.: I2CEXT F1 W2400 R2402 W2500 R2502 W2600 R2602 W2700 R2702<CR> >I2CEXT W20020000 W20060000 W21020000 W21060000 W22020000 W22060000 W23020000 W23060000	
PING<CR>	Ping support	
Example:	Send	PING<CR>
	Receive	>PONG
RS485SETMODE [mode]<CR>	Setting RS485 receiver communication mode	
Example:	Send	RS485SETMODE 1<CR>
	Receive	> RS485SETMODE OK<CR>
Note:	The RS485 receiving communication mode can be : text (mode=0) data received is interpreted as text, communication is terminated by <LF> character ('\n'). Data should be read by the RS485TEXT command. hex (mode=1) data received is interpreted as hex, the received buffer	

Command	Description	
	is presented as whatever is received at the moment when it is read by the ethernet interface with the RS485RDHEX command. Default value is text mode.	
RS485GETMODE<CR>	Getting RS485 receiver communication mode	
Example:	Send	RS485GETMODE<CR>
	Receive	> RS485GETMODE 1<CR>
Note:	Default value is text (mode=0).	
RS485SETBAUD [baudrate] <CR>	Set the desired RS485 baudrate.	
Example:	Send	RS485SETBAUD 115200<CR>
	Receive	> RS485SETBAUD 117647<CR>
Note:	Response contains the actual baudrate set and calculated. Default baud rate is 115200.	
RS485GETBAUD<CR>	Get the RS485 baudrate.	
Example:	Send	RS485GETBAUD<CR>
	Receive	> RS485GETBAUD 117647<CR>
Note:	Response contains the actual baudrate set and calculated. Default baud rate is 115200.	
RS485WRTEXT [message]<CR>	Write text message on RS485	
Example:	Send	RS485WRTEXT getresistance<LF><CR>
	Receive	>RS485WRTEXT OK<CR>
Note:	This example sends a request to the RDECADE24BIT unit connected on	

Command	Description	
	RS485 to the IOCard to query the actual resistance setting. Note that RDECADE24BIT unit uses <LF> ('\n')for message termination.	
RS485RDTEXT<CR>	Read text message on RS485	
Example:	Send	RS485RDTEXT<CR>
	Receive	>RS485RDTEXT OK getresistance 1666665<LF><CR>
Note:	This example gets the response sent from the RDECADE24BIT unit connected on RS485 to IOCard. The response is stored in the IOCard buffer. Note that RDECADE24BIT unit uses <LF> ('\n')for message termination.	
RS485WRHEX [length][msg]<CR>	Write hexadecimal message on RS485	
Example:	Send	RS485WRHEX 09 67 65 74 66 77 76 65 72 0A<CR>
	Receive	>RS485WRHEX OK<CR>
Note:	length – hexadecimal value, contains the length of the following message bytes message bytes – the bytes as text (2 character) separated by space character The result of this message on RS485 would be to send 9 bytes "getfwver\n"	
RS485RDHEX<CR>		

Command	Description	
	Write hexadecimal message on RS485	
Example:	Send	RS485RDHEX<CR>
	Receive	>RS485RDHEX 15 4F 4B 20 67 65 74 66 77 76 65 72 20 31 2E 30 30 30 30 30 30 0A <CR>
Note:	first byte (hexadecimal representation) in the response is the length of the following message bytes. The result of the received message on RS485 would be that it received 21 bytes "OK getfwver 1.000000\n"	