Module WIRE-CHIP h4.1**E** with ETHERNET port

User manual (firmware s4.21E)

Description

Only 2 wires are needed to operate 64 temperature sensors (GND, wire - "1-wire" bus). This makes the use of the module with DS18B20 sensors much cheaper than traditional thermocouple or resistance sensors. All sensors are connected in parallel, each sensor has its own unique number, which allows the sensors to be distinguished.

The version with an ETHERNET port enables communication via the MODBUS TCP protocol (except MODBUS RTU) and operation as a MODBUS TCP / RTU protocol converter.

Free, easy-to-use software is used to configure the device. The module also works with other elements using the 1-wire bus (analog inputs - DS2438, digital inputs / outputs - DS2413 and DS2408).

Usage

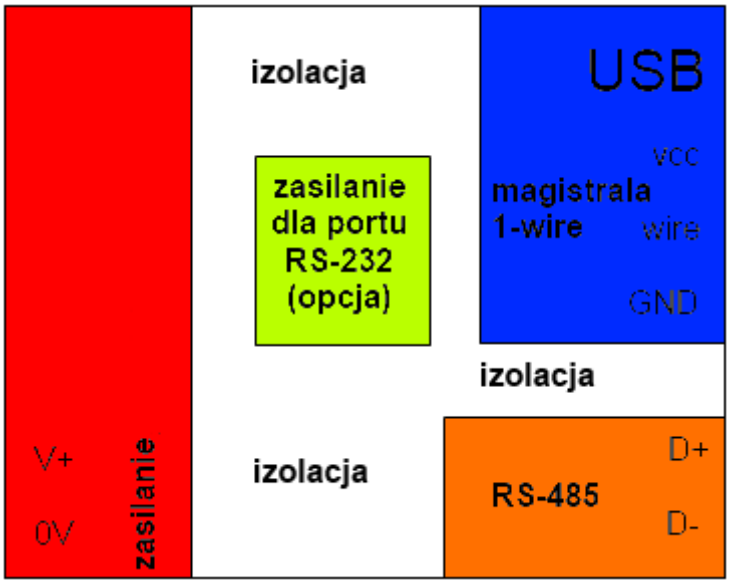
- intelligent building;
- multi-point temperature monitoring (grain silos, mushroom farms, warehouses);
- temperature monitoring and control applications (greenhouses, dryers, heating);
- monitoring for HACCP purposes;
- distributed digital input/output systems – using 1-wire DS2408 or DS2413 components

Technical data

- Temperature measurement:
  - **temperature measurement range: -55 .. +125 °C** (sensor **DS18B20**);
  - Max measurement error: ±0,5°C in the range -10 +80°C (sensor DS18B20);
  - Measurement resolution: 1/16°C = 0,0625°C (sensor DS18B20);
  - transducer **DS2438**:
    - Measurement **: -40 .. +85 °C**;
    - Maximum temperature measurement error: ±2°C
- Measurement of relative humidity (transducer CHIP-2438-MICRO-RH):
  - Measurement range: 0..100 % (bez kondensacji)
  - Max error: +/- 3,5%
- Digital inputs / outputs:
  - Open-Drain inputs (0 – contain from GND lub 1 – don't contain)
  - **DS2413** – 2x input / output;
  - **DS2408** – 8x input / output;
    - Entries with activity recording (Activity Latches)
- Other data exchanged with elements 1-wire:
  - Voltage measurement from the range 0..10V dc (transducer CHIP-2438-MICRO-0..10V);
  - Current measurement 0..20 mA (-20 .. +20 mA) (transducer CHIP-2438-MICRO-0..20mA);
  - EEPROM memory support contained in 1-wire systems (read and record; currently for DS18B20 and DS18S20);
  - Other measurements depending on the firmware version;
- Max bus length 1-wire: 300m;
- Serial ports**: RS-485, USB** (communication protocol **MODBUS RTU)**;
- ETHERNET port: MODBUS TCP protocol, RJ45 connector;
  - Auto MDI/MDIX;
  - Transmission speed 10/100 Mbps;
  - number of simultaneous TCP connections: 8;
  - KeepAlive function with adjustable response time;
  - MODBUS TCP / MODBUS RTU converter;
  - mini web server with simple visualization – through macro instructions
- Power **10..30 V dc, max 2W**;
- **Galvinic isolation 2,5 kV** between the main circuits.
  Isolated circuits:
  - power,
  - RS-485,
  - 1-wire and USB,
  - optional equipment;

- Adaptation housing for **DIN rail mounting** (dimensiojs: width 34mm, depth 65mm, height without plugins 89 mm, with plugs 120 mm);
- Degree of protection of the housing: IP30;
- working temperature: -30 ... +50 °C;
- relative humidity: <95%, without condensation;
- opportunity to develop (equipment and software) according to the customer's need.

Galvinic isolation (izolacja) diagram of the WIRE-CHIP h4.1 module



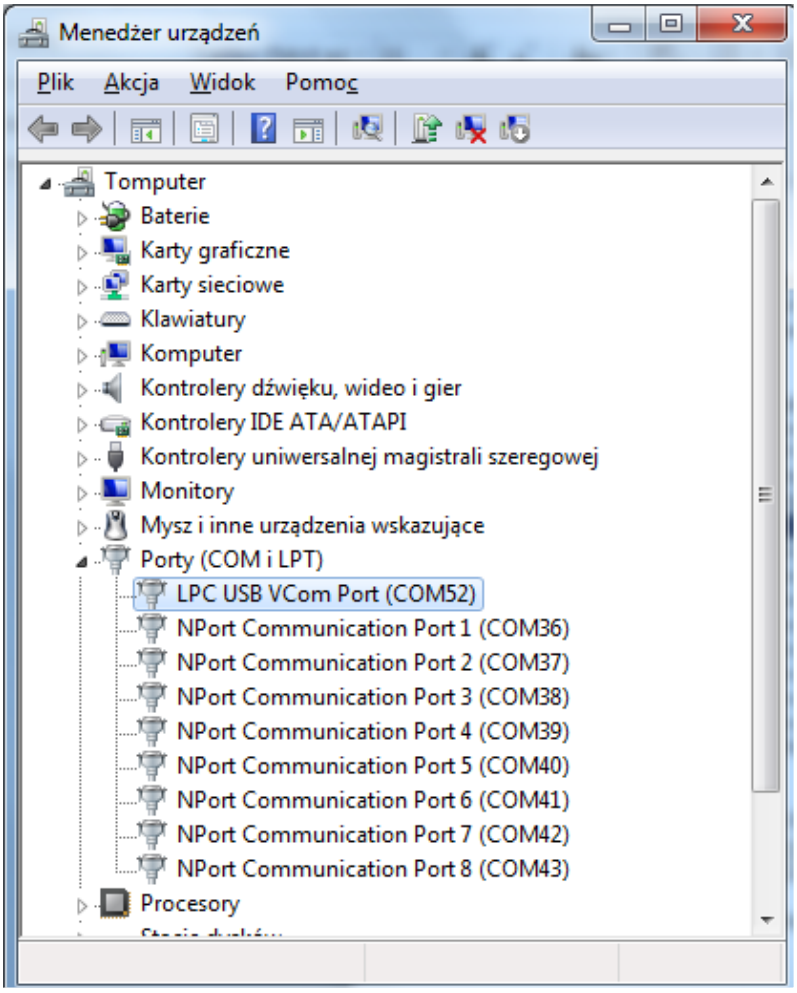
Module installation

Connection to computer via USB

Connect the WIRE-CHIP h4.1 module to your PC via the USB port (WIRE-CHIP has a 5-pin USB mini B socket). Windows will ask for the driver file, point to "lpc-vcom.inf" (which comes with the configuration software (wire-configurator) on CD). Windows will create a new COM port that can be used with a wire-configurator or other software that needs to share measurement data.

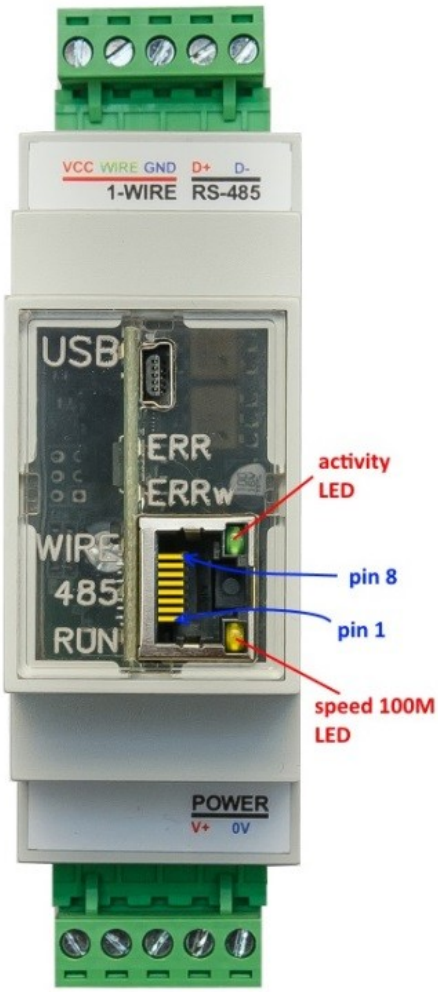
You can check the correct installation of the module in the Windows "Device Manager" window. In the "Ports (COM and LPT)" branch there should be a new device "LPC USB VCom Port" along with the COM port number.

The figure below shows a screenshot of the "Device Manager" with a correctly installed module that received the COM52 port number in the system.



Connecting the ETHERNET port

The module is connected to the LAN network via an RJ-45 connector.
Individual pins in the RJ-45 connector mean:
1 – TX+
2 – TX-
3 – RX+
4,5 – Vin1 – power connection;
6 – RX-
7,8 – Vin2 – power connection;



Thanks to the "Auto MDI/MDIX" function, you can exchange TX and RX signals, i.e. you can use a straight or crossed network cable. For proper communication with other network devices, 2 pairs of wires are needed in the cable (pins 1,2,3,6).
The remaining 2 pairs (pins 4.5 and 7.8) can be used to power the module and other devices located nearby with a maximum average current consumption of 0.3A. The supply voltage from the ETHERNET port connector is connected to the module's main power supply terminals V+ and 0V via a rectifier bridge.

The ETHERNET connector is equipped with 2 LEDs:

- activity LED – blinking indicates that the ETHERNET port is communicating with the network environment;
- speed LED – lit means that the port is operating at a speed of 100 Mbps, the LED off means that the speed is 10 Mbps.

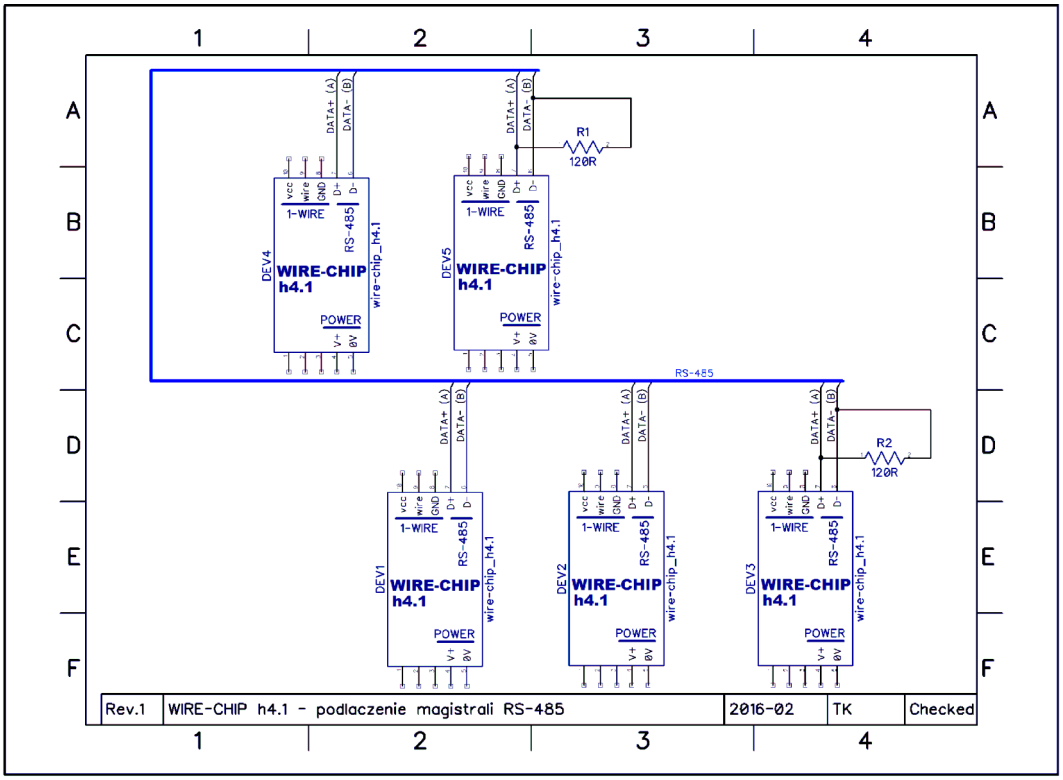
MODBUS TCP protocol support

After initializing the ETHERNET port, the WIRE-CHIP module is ready to work as a server answering queries compliant with the MODBUS TCP protocol. The MODBUS TCP frame contains a field marked "UNIT IDENTIFIER", which in the MODBUS RTU protocol corresponds to the ID of the SLAVE module. In the WIRE-CHIP module, the UNIT IDENTIFIER field specifies what the query of the client device (remote device connecting to WIRE-CHIP) concerns. Values 0 or 255 mean that the question concerns the resources of the WIRE-CHIP module in question (registers 0..2047 or bits in these registers). The values 1..254 mean for the module that the question should be sent via the COM1 (RS-485) serial port to an external device connected to WIRE-CHIP communicating via RS-485. Then the module works as Modbus Data Gateway (MODBUS TCP / RTU protocol converter). In order for the module to forward the MODBUS TCP query to the RS-485 port, this port must be initialized as MASTER. For more information, see the description of the "PORT_INIT" macroinstruction.

MODBUS TCP functions in WIRE-CHIP:

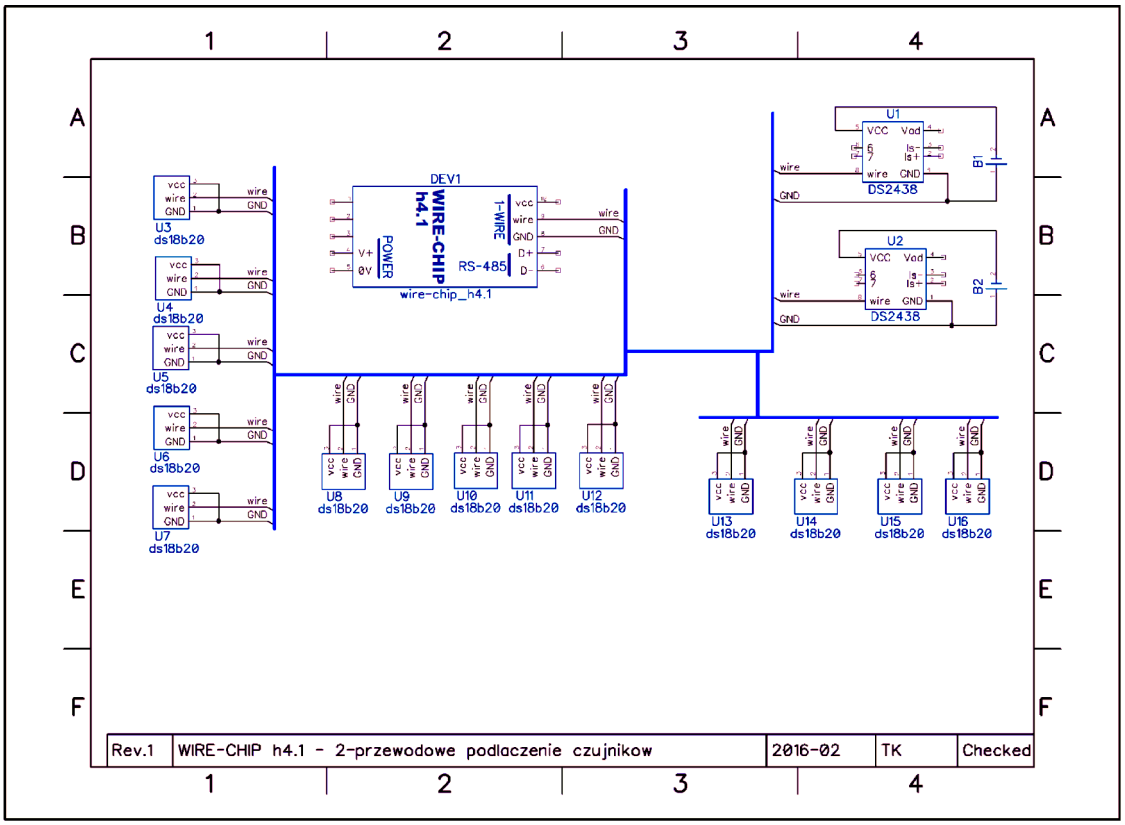
- x03 – Read Holding Registers – reading from module registers (range 0..2047);
- x10 – Preset Holding Registers – writing to the module registers
- x01 – Read Coils – reading bits located inside the above registers
    - specifying the bit address "bit_no" means that the first bit will be read bit = (bit_no % 16) {remainder from dividing bit_no by 16} from the register = (bit_no / 16) {integer division of bit_no by 16};
        - Example: if bit_no = 1245, bit = 13 is indicated in register 77
- x0F – Force Coils – entering bit values (as above)
- these and other functions of the MODBUS protocol can be transferred via the RS-485 port to other devices;

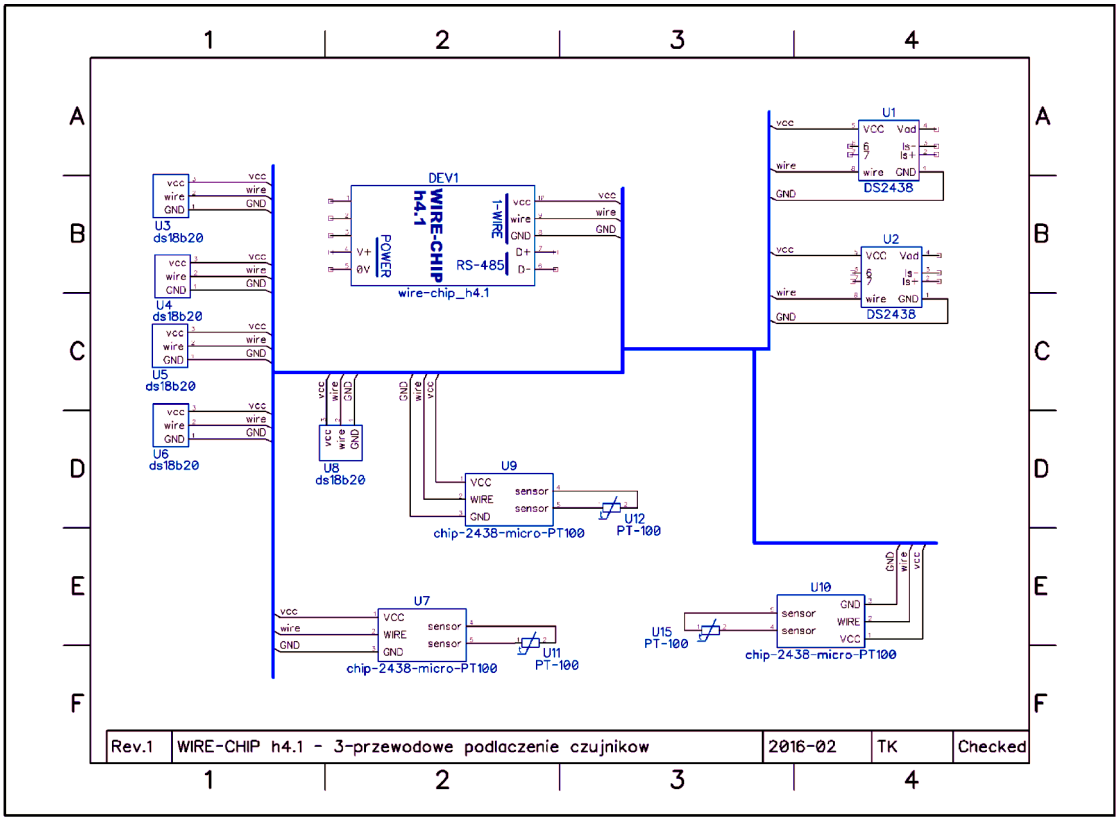
Connecting bus RS-485



Note the resistors R1 and R2 about resistance 120Ω, which should be installed for the bus RS-485 with a length exceeding 30..50 m. With shorter bus RS-485 lengths, the „wave reflection" phenomenon is negligible.

Connection bus 1-wire 2-wire i 3-wire (sensors DS18B20, DS18S20, DS2438, DS2413, DS2408)





Module power supply

The module can be powered in three ways:

1. through the 0V and V+ (POWER) power supply inputs, connecting a voltage in the range of 10...30 Vdc to them;
    a. this allows all circuits to work:
        i. 1-WIRE andprocessor;
        ii. RS-485;
        iii. optional equipment,
        iv. port ETHERNET;
2. via USB port
    a. this allows operation only for the 1-WIRE bus and the processor and the ETHERNET port
3. via ETHERNET port
    a. through the rectifier bridge, the voltage is supplied to the 0V and V+ (POWER) power supply input, and then as in 1.
    b. from the POWER terminals you can power an external device whose current consumption does not exceed 0.3A (e.g. another WIRE-CHIP module with which SCADA communicates using WIRE-CHIP with Ethernet as a MODBUS TCP/RTU converter);

Software

To configure the module, use the „wire-konfigurator” software. It allows you to make any changes to the module

The „wire-konfigurator” software should be copied to the computer’s disk so that it can save log files in its own folder, which can be helpful in analyzing the operation of the software and the module.

Basic elements of the configuration software:

The next picture contains:

- Communication status informs about the curremtly performed operations or status (succes / failure) of the last operation. Kclicking on „communication status” opens the log file.
- Communication parameters with the configure module
    - COM port number (module connected via USB or converter USB/RS-485, opptional port RS-232),
    - Baud rate, parity (they don’t matter when directly connected via USB),
    - ID – when directly connected via USB always zero („0”). when connected via RS-485 or RS-232 in accordance with the module settings.;
- Reading and writing module configuration. The „wire-konfigurator” program zremembers a configuration that can:
    - read from a file or module (when reading from a module, macro instructions aren’t read),
    - modify
    - save to a file or module
        - Some settings require a module reset, są a checkbox has been provided that allows you to perform a reset after entering the configuration.

reading / writing configuration from / to file

To read the configuration from file, press the „read from file” button. An open file selection window with a file type filter will appear. By default, files with the extension 'chip' are displayed, intended for use with the WIRE-CHIP module in version h4.1.

All available extensions:

- 'chip' - XML configuration files for the WIRE-CHIP module h4 and h4.1;
- 'rej4' - binary configuration files for WIRE-CHIP h4 (currently not developed);
- 'rej' - binary configuration files for WIRE-CHIP in versions h3.1 and h2.

To save the configuration to a file, press the 'Save to FILE' button, and then select a file name. The available format for saving is XML with the 'chip' extension.

Reading / Writing Configuration to / from the Module

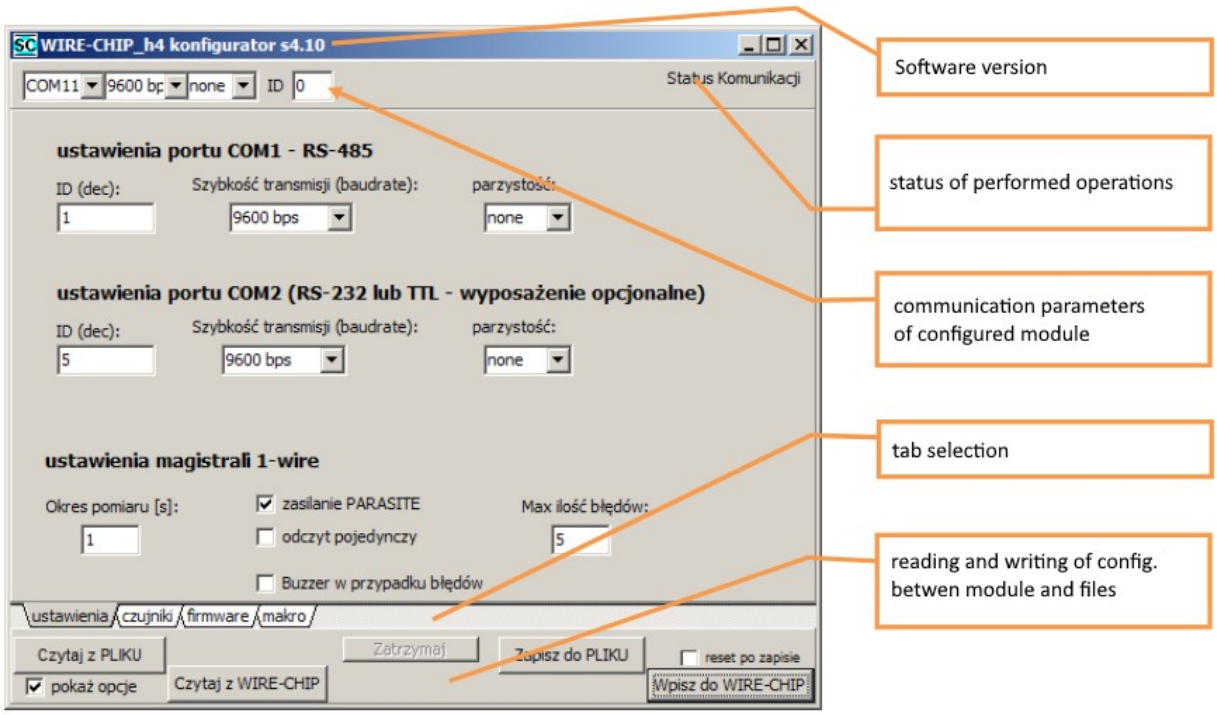
To read the configuration from the module, press the "Read from WIRE-CHIP" button.
To read or write the configuration, communication parameters should be set (COM port number, baudrate, parity and module ID). If the module is connected to the computer via USB, the module ID is "0", the baud rate and parity do not matter.

Interrupting ongoing operations

Operations performed by the configuration software on the WIRE-CHIP module can be interrupted by pressing the "Stop" button.

Basic Module Configuration

The following figure illustrates the appearance of the 'Settings' tab, which is used to make changes to basic settings:



The "Settings" (ustawienia) tab provides access to the following parameters:

- COM1 Serial Port - RS-485:
  - ID (dec) –Module ID for MODBUS RTU protocol (slave) {0..255};
  - Transmission speed (Szybkość transmisji baudrate) {9600, 19200, 38400, 57600, 115200 bps};
  - Parity (parzystość) {none, ODD, EVEN}
- COM2 Serial Port (RS-232 lub TTL) – optional equipment
  - Same parameters as for COM1
- 1-wire bus:
  - **Measurement interval** (Okres pomiaru) – time interval beetwen individual measurements {1..255s},
  - **PARASITE powering** (zasilanie PARASITE) – enables 2-wire temperature sensor support. In the 2-wire mode, after sending the "measure temperature" command to the temperature sensor, no communication can take place on the 1-Wire bus as the "wire" provides power to the sensor. After 1 second, power is disconnected from the "wire," and communication with the sensor occurs.
  - **Single reading** (odczyt pojedynczy) – sending the "measure" command to each sensor separately, after another sensor has completed its measurement. In the case of using current barriers (Zener barrier for hazardous explosive [EX] areas), there may be a situation where, after starting the temperature measurement in all sensors

simultaneously, the barrier will block the transmission of the appropriate current to the sensors, and measurements will not take place. This option protects against measurements being blocked by current barriers.

- **Buzzer for errors** (Buzzer w przypadku błędów) – the module signals sensor errors with sound.
  - This option is useful when testing sensors.
- **Max error count** (Max ilość błędów) – the number of errors (not separated by successes) in communication with the sensor after which the module stops providing the last successfully read temperature (e.g., 2245 = 22.45°C) and provides a value indicating an error (-20000 = -200°C - a value outside the range);

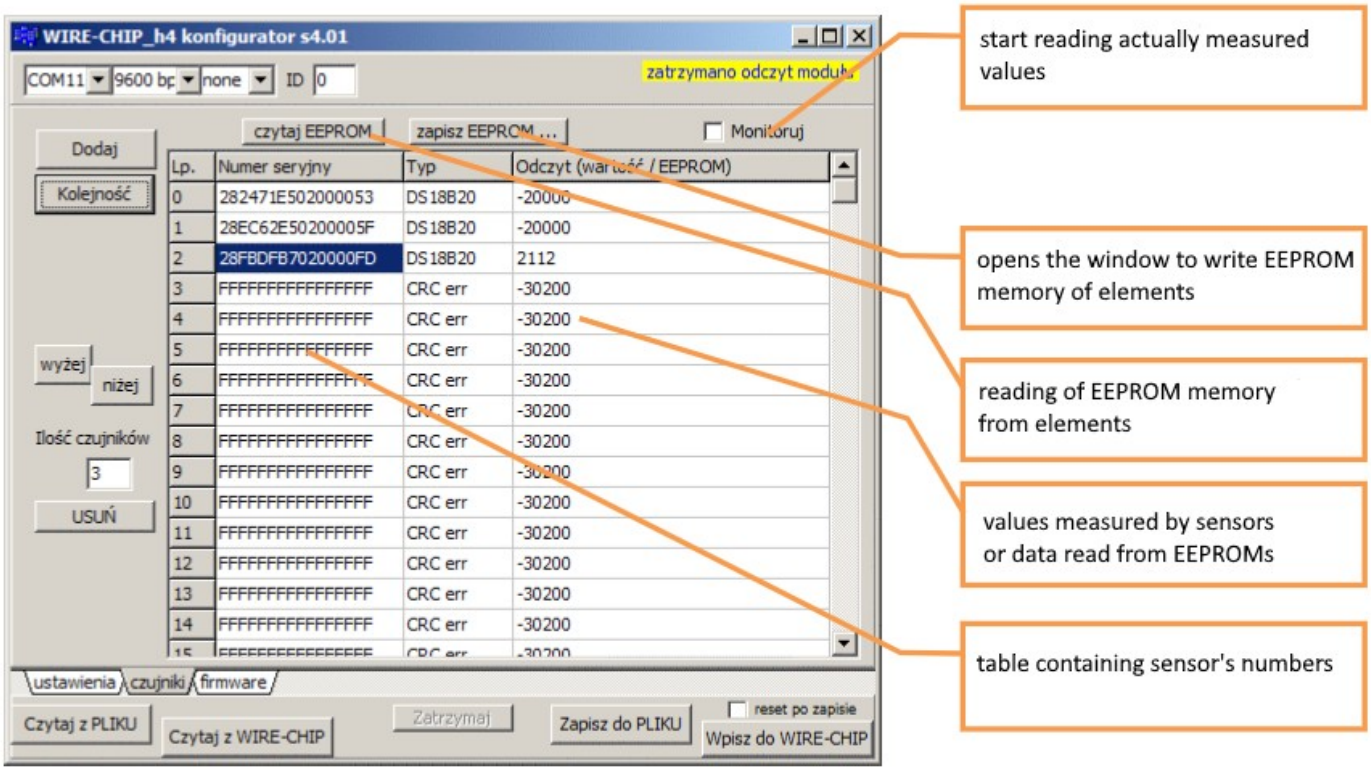
Frame data parameters for WIRE-CHIP module for serial port transmission:

- Number of data bits: 8;
- Number of stop bits: 1;

Baudrate and parity are configurable.

Configuration of temperature sensors and other "1-wire" elements

Below is a screenshot of the "sensors" tab, where the configuration of elements connected to the 1-wire bus is made:



As part of the configuration of elements (sensors) connected to the 1-wire bus, the following operations can be performed:

- add elements (sensors);
- determine the order;
- sort;
- to remove;
- read the contents of the EEPROM memory;
- save the contents of the EEPROM memory;

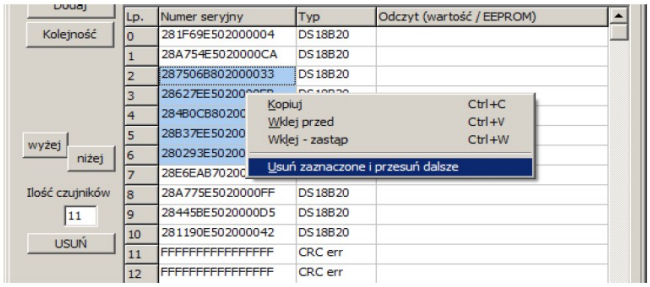
Sensor table pop-up menu

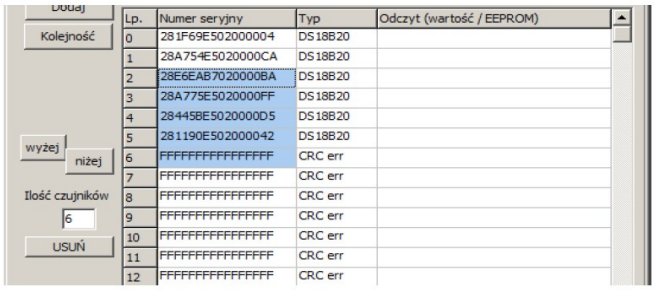
It enables:

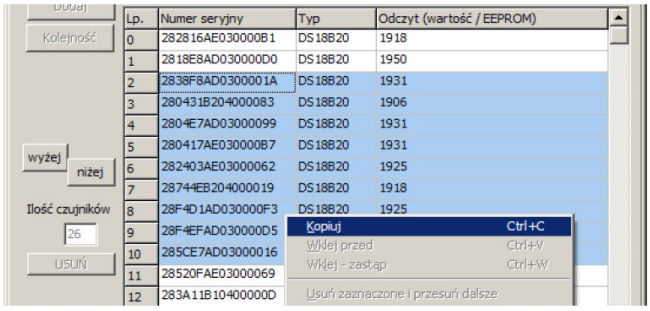
- **copying** (kopiowanie) the contents of selected fields;
    - select cells in the sensor table;
    - select "Copy" in the context menu or select the keyboard shortcut "Ctrl+C";
    - the data is saved in the MS WINDOWS clipboard and can be used in another program (e.g. notebook, MS EXCEL, ...);
- **pasting** sensors **before** (wklej przed) the selected row;
    - select 1 or more cells from the column with sensor serial numbers
        - the number of inserted sensors will be:
            - if 1 table cell is selected, then the number of sensors will be inserted the same as the number of serial numbers saved in the MS WINDOWS clipboard;
            - if more than 1 table cell is selected, then the number of serial numbers selected will be inserted, unless a smaller number of serial numbers are stored in the clipboard, then the same number of sensors will be inserted as in the clipboard.
    - if the clipboard contains serial numbers of sensors (e.g. copied from the WIRE-CHIP configuration program or from MS EXCEL), select "Paste before" from the context menu (or the keyboard shortcut "Ctrl+V");
    - the sensor selected first and all subsequent sensors will be moved by the inserted number of sensors,
    - the number of sensors will increase:
        - if sensors are pasted before or at the end of previously installed sensors, the "number of sensors" parameter will be increased by the inserted number of sensors;
        - if sensors are inserted far from the installed sensors, then the number of new sensors will be added to the current number of sensors, as well as the distance between the installed and new sensors
            - Example: 5 sensors are installed; 5 more sensors were pasted, starting from No. 10. After the operation, the "number of sensors" field contains the number 15, Sensors No. 5 to 9 contain empty data.
- **pasting-replacing** selected sensors (wklej zastąp) ;
    - select 1 or more cells from the column with sensor serial numbers
        - the number of inserted sensors will be:
            - see – paste before (above);
    - if the clipboard contains serial numbers of sensors (e.g. copied from the WIRE-CHIP configuration program or from MS EXCEL), select "Paste - replace" from the context menu (or the keyboard shortcut "Ctrl+W");
    - the sensor selected first and all subsequent sensors will be moved by the inserted number of sensors,
- **deleting** selected sensors (usuwanie);
    - select 1 or more cells from the column with sensor serial numbers;
    - select "Delete selected and move further" from the context menu
    - sensor serial numbers will be replaced with "F" values
    - The number of sensors will be reduced by the number of selected sensors, if installed sensors were selected;
    - sensor serial numbers in later positions will be moved to earlier positions

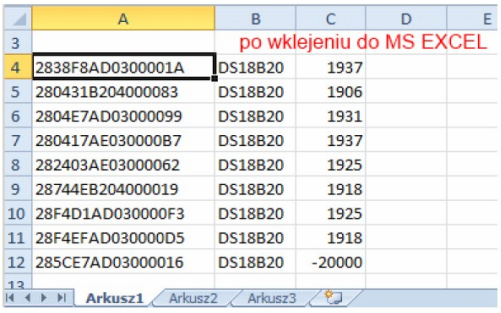
To call up the context menu of the sensor table, press (click) the right mouse button over the sensor table, or press the context menu key on the keyboard (when the sensor table is the active visual element of the program).

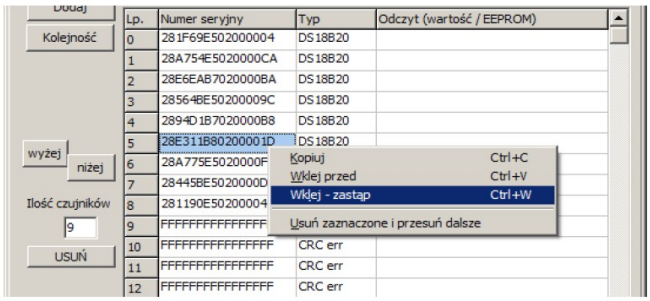
Examples of using the shortcut menu function

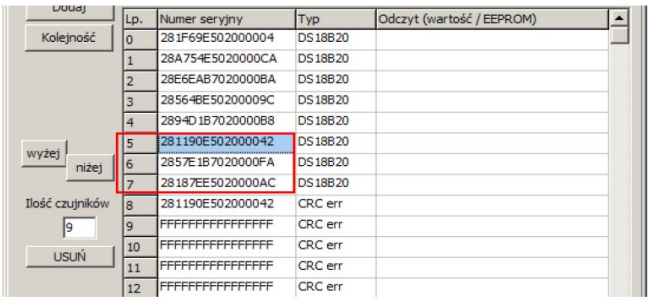


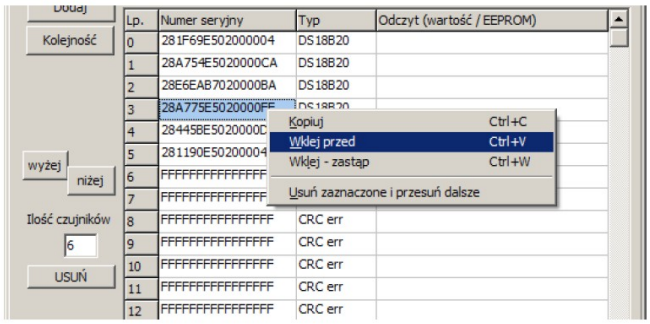


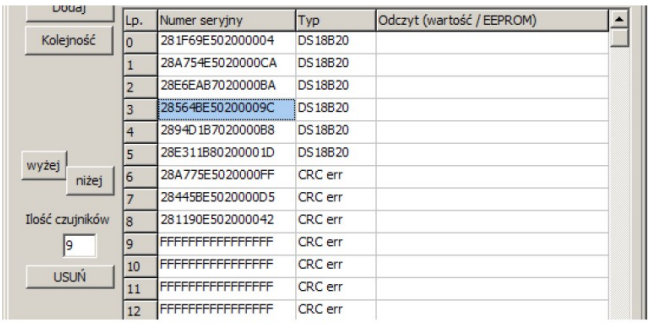


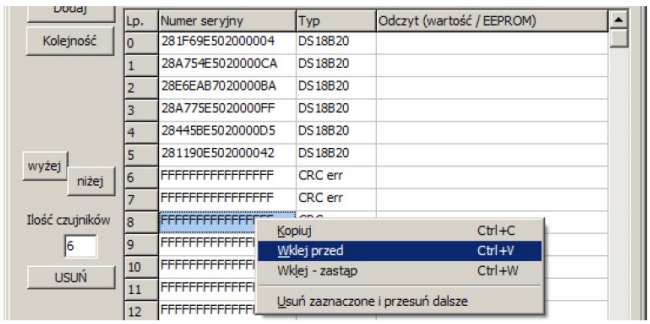


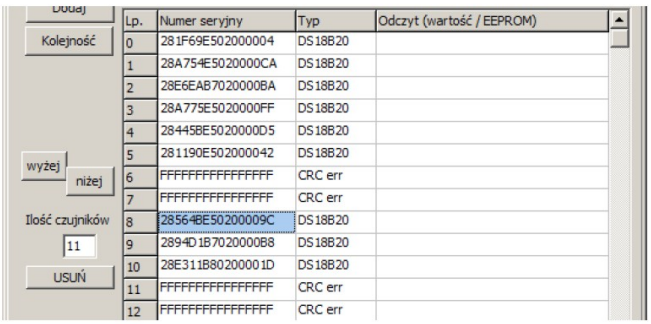












To add sensors...

Press the "Dodaj" button

The module will search for all connected sensors. It will compare the serial numbers of the found sensors with the numbers saved in flash memory (own configuration memory). The configuration program (wire-configurator) returns a list of serial numbers already saved in flash and a list of elements that the module has not yet saved (newly added).
To add new elements (sensors), previously installed elements do not have to be connected.
The maximum number of installed 1-wire elements is 64 pcs. The module will support the number of elements stored in the "Number of sensors" edit field.

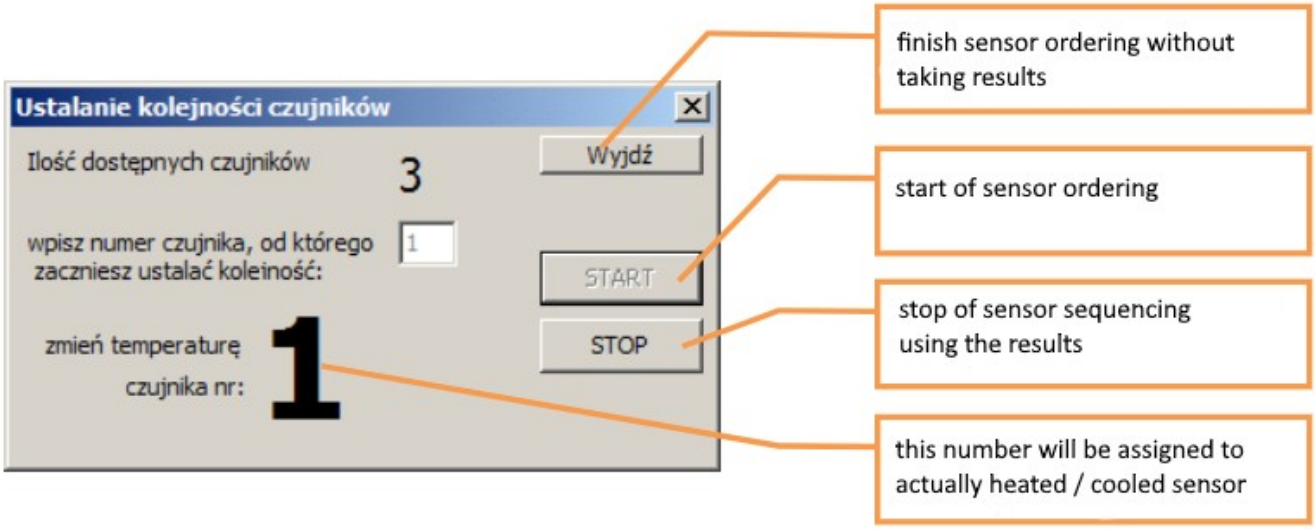
To arrange the order of 1-wire elements...

Press the "Kolejność" button. This will open a dialog box (the window appears in the figure below). The window allows you to start and stop sequencing. The module determines the order by detecting a rapid (1°C / measurement period) temperature increase (decrease) on individual sensors (heating with a burner, cooling with a freezer, etc.). If the sensor sequencing function asks you to change the temperature of a sensor with a specific number (in the figure below, no.=1), a quick change of temperature will result in assigning no. 1 to the currently heated/cooled sensor.

Within one of the launch functions, tracking, the change occurs only 1 time.

After detecting a temperature change on the last sensor number (e.g. no. 2 for 3 available sensors), the program will automatically copy the results to its own memory.

The appearance of the ordering dialog box



Order of digital inputs/outputs (DS2413 and DS2408)

Instead of temperature changes (DS2413 and DS2408 systems do not measure temperature), digital inputs are used to set the sequence. For the purposes of determining the sequence, a change in the state of the digital input is treated as a change of 1°C in the case of temperature sensors.

The program allows you to **manually arrange the order of sensors.** Select the sensor serial number (click on the number), then press the "wyżej" (up) and "niżej" (down) buttons to change the order of the sensors.

To sort sensors...

"by serial numbers", after reading the module configuration, click on the column header with serial numbers of the sensor table. One click will set the ascending order, another click will set the descending order.

To sort **by the contents of the EEPROM memory** from the sensors, read the data from the sensors by pressing the "read EEPROM" button. Then click on the column header with the read memory content of the sensor table.

To remove all sensors

Press the "USUŃ" (delete) button. Information about the sensors is then deleted from the memory of the configuration program. This does not mean that sensor information is removed from the WIRE-CHIP module.

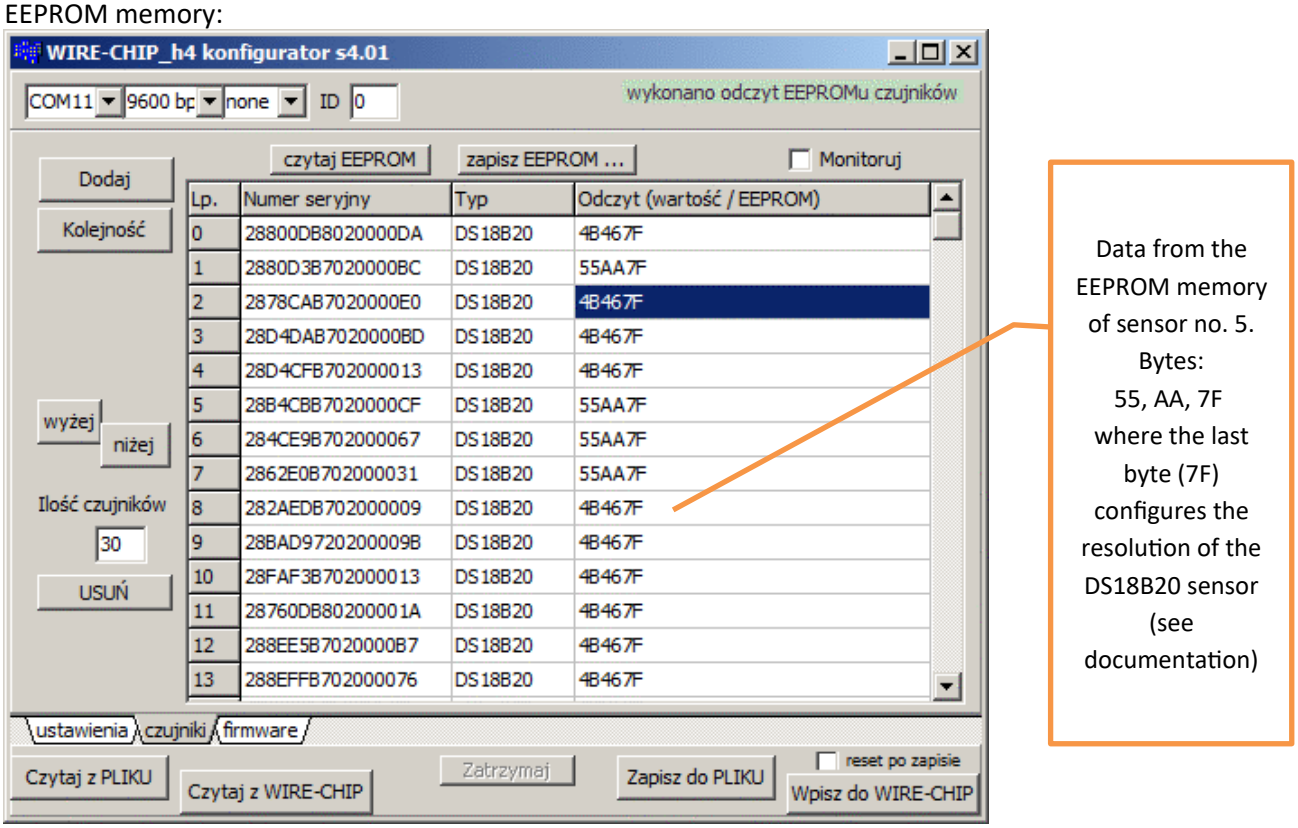
Attention

Items are installed in the module if they are saved in flash memory. After each configuration change in the configuration program, enter the new configuration into the flash memory by pressing the "Write to WIRE-CHIP" button.

Reading the contents of the EEPROM memory of elements

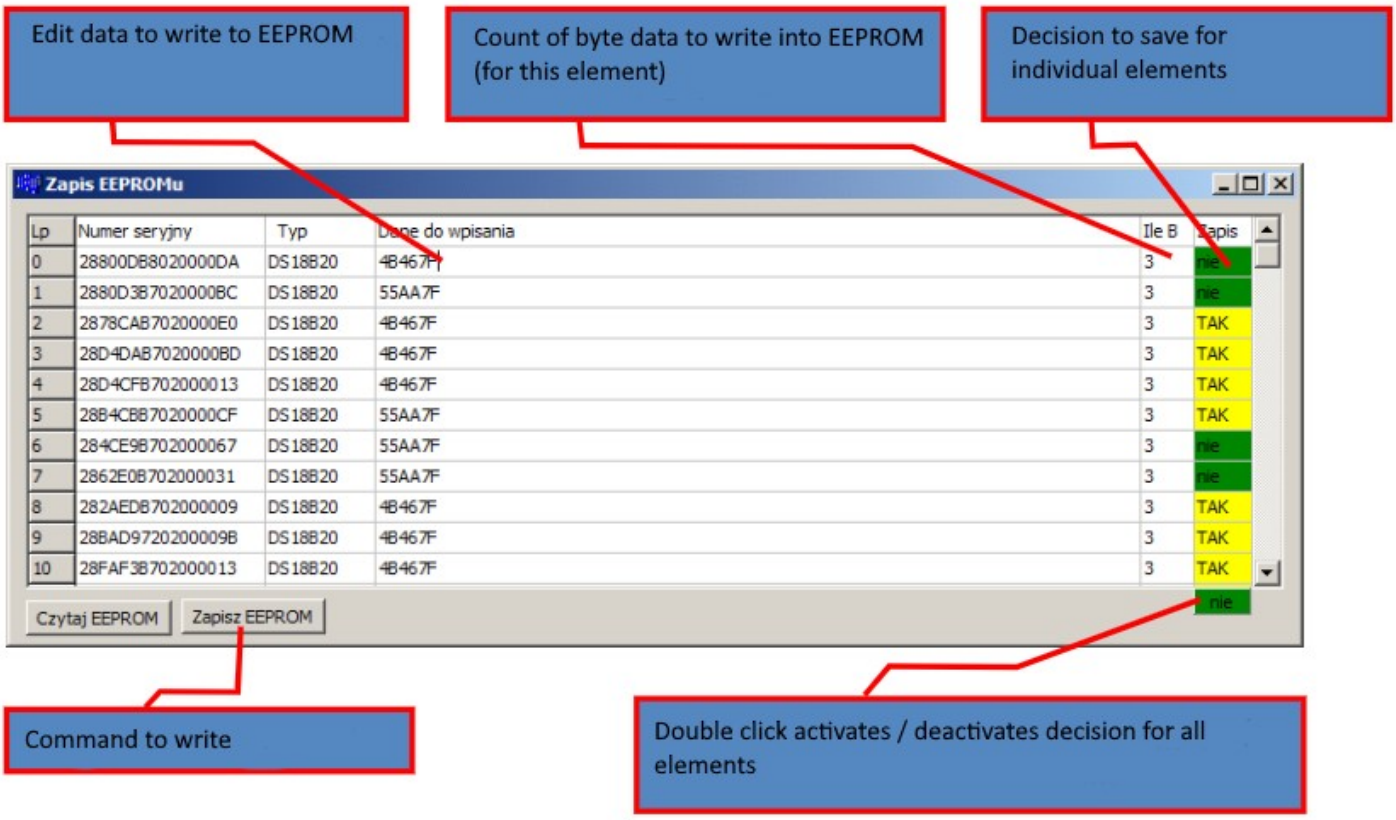
To read the contents of the EEPROM memory of 1-wire components installed in the module, press the "read EEPROM" button. Then the module reads data from the elements and passes it to the configuration program. The configuration program displays data in the sensor table, in the "Read" column. Data is displayed in hexadecimal (HEX). If the sensor is installed in the module but not connected, the program displays bytes with the value FF (All bits '1').

The figure below shows the appearance of the configuration program after reading the EEPROM memory:



Writing the contents to the EEPROM of 1-wire elements

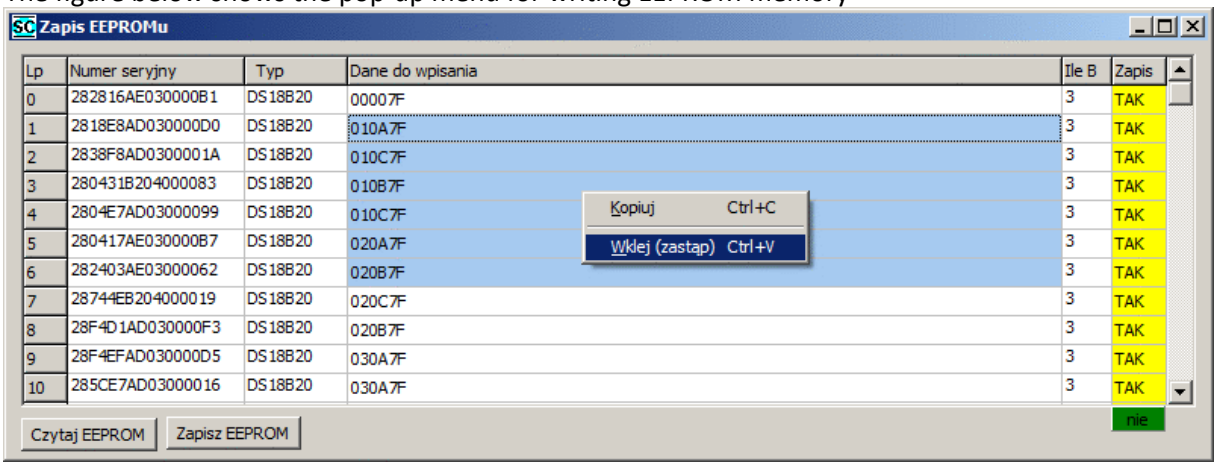
After reading the data from the EEPROM memory, you can edit it and then write it to the elements. To open the editing and saving window, press the "Zapisz EEPROM..." button. The appearance of the window in the next drawing.



After editing the data to be entered for each sensor, press the "Zapisz EEPROM" (save eeprom) button. Pressing the button will transfer data to the WIRE-CHIP module and save data to individual elements. If you do not want to enter data for a given item, you must select "no" in the save decision field. After reading the contents of the sensors' EEPROM memory, the configuration program initially decides whether a given sensor can be saved. If the EEPROM content could not be read from the sensor (communication error), the configuration program then suggests not to write data to this sensor (element).

EEPROM memory saving table pop-up menu

The figure below shows the pop-up menu for writing EEPROM memory



The menu allows:

- **copying** the contents of selected fields;
    - select cells in the table;
    - select "Copy" in the context menu or select the keyboard shortcut "Ctrl+C";
    - the data is saved in the MS WINDOWS clipboard and can be used in another program (e.g. notebook, MS EXCEL, ...);
- **pasting – replacing** data to be entered
    - select 1 or more cells from the column with the data you want to enter
        - the amount of data replaced will be:
            - if 1 table cell is selected, then the amount of data stored in the MS WINDOWS clipboard will be replaced;
            - if more than 1 table cell is selected, then the amount of data selected will be replaced, unless a smaller amount of data is stored in the clipboard, then the same number of sensors will be inserted as in the clipboard.
    - if the clipboard contains data to be written to sensors (e.g. copied from MS EXCEL), select "Paste (replace)" from the context menu (or keyboard shortcut "Ctrl+V");

Monitoring the current operation of the module

To check the currently measured temperatures, select the "Monitor" checkbox. Then the "Read" column will display the current values read from the elements (sensors). Monitoring continues until further notice (interruption of operations or unchecking the checkbox).

Reading measurement values from the module

MODBUS RTU register map

The WIRE-CHIP module provides registers in the following ranges:

- **0..2047** – of which registers 0..63 contain information about measured temperatures. The remaining registers in this range are free to use.
- **≥ 65000** – registers in this range are reserved for module configuration. Writing to registers in this range may cause incorrect operation of the module - do not write data to these registers yourself.

Interpretation of data read from registers

Temperature measurement values are read from registers 0..63 by the MODBUS RTU protocol through any of the serial ports.
The value from the register divided by 100 corresponds to the temperature in degrees Celsius (for temperature sensors).

Examples of values read from registers:

- For temperature measurements (e.g. DS18B20) and for analog inputs (DS2438)
    - 2245 = 22,45°C;
    - 12500 = 125°C;
    - -20000 = -200°C – error value (sensor is installed, but communication failed);

- For digital inputs/outputs (DS2413 and DS2408) – below (in the subsection "Supporting various elements with a 1-wire bus").

Support for various components with 1-wire bus

- Temperature sensors – **DS18B20**, DS18S20, DS1820
  - The temperature is read cyclically and placed in the register with the number corresponding to the element number (0..63);
  - Read/write EEPROM memory contents;
- **DS2438 voltage converter**
  - The temperature is read cyclically - as for temperature sensors (above);
  - Using the cyclically called DS2438_CONVIV macro instruction (DS_ALL event) - measurements of the voltages connected to the inputs of the system are performed, the values are saved in the registers indicated in the macro:
    - Voltage connected to Vad (pin.4) – in the range 0..10V
    - Supply voltage Vcc (pin.5) (range 0..10V);
    - Voltage on the current measurement differential inputs (pin.2 V+ and pin.3 V-) in the range -250mV .. +250mV
  - Converting voltage values to other quantities:
    - Relative humidity – using the RH_HIH5031 macroinstruction (the CHIP-2438-MICRO-RH module uses the HIH5031 humidity sensor)
    - PT100, PT1000, sunlight, and others - using the "linear" macro (MATH group);
- Digital input/output systems
  - **DS2413** – 2 inputs/outputs;
    - The status of the inputs and the status of the outputs are updated cyclically (together with the temperature reading);
    - The register number corresponding to the number among the 1-wire elements is intended to operate DS2413;
    - Meaning of register bits:
      - b15 – communication error (1=error, 0=communication ok);
      - b8, b9 – bits written to digital outputs (o0 and o1);
      - b1, b3 – reading the status of digital outputs (o0 and o1);
      - b0, b2 – reading the status of digital inputs (i0 i i1);
      - remaining bits - irrelevant;
  - **DS2408** – 8 inputs/outputs
    - The status of the inputs and special bits is read cyclically (together with the temperature reading). (AL, PORL, VCC_stat);
    - The register number corresponding to the number among the 1-wire elements is intended to operate DS2413;
    - Meaning of register bits:
      - b15 – communication error (1=error, 0=communication ok);
      - b10 – VCC_stat - DS2408 power status (1=VCC, 0=no_VCC);
      - b9 – PORL – system shutdown detected (Power On Reset Latch);
      - b8 – AL – any Activity Latch bit is 1;
      - b0..b7 – any Activity Latch bit is 1; (i0..i7);
      - remaining bits - irrelevant;
    - the output state is saved cyclically by the DS2408_WR macroinstruction
      - bits b0…b7 of the register with data are transferred to the digital outputs;
      - remaining bits - irrelevant;

- reading the Activity Latch register is performed by the DS2408_AL macroinstruction;

MAKROINSTRUCTIONS

The WIRE-CHIP module can perform other activities in addition to temperature measurement. Other module activities are grouped according to function:

- **DIG_IO** – digital input/output support (optional equipment);
- **MATH** – mathematical functions;
- **LOGIC** - logical functions (binary);
- **MEMORY** - memory operations;
- **JUMPS** – program flow control;
- **SPECIAL** - special – conversion, complex analysis of measurement results;
- **SERIAL** - communication via COM ports (serial);
- **1-WIRE** - communication via 1-wire;
- **HTTP** – generating visual elements for the website

Resources

WIRE-CHIP h4.1 has the following resources at its disposal, which it can use via macro instructions:

- 2048 16-bit registers – treated as numbers or bits
- 5 inputs/outputs – optional equipment
- Port 1-wire
- Port RS-485
- Port RS-232 – optional equipment

Events

The module executes macro instructions when a specific event occurs. The following events are currently defined for which macros are executed when:

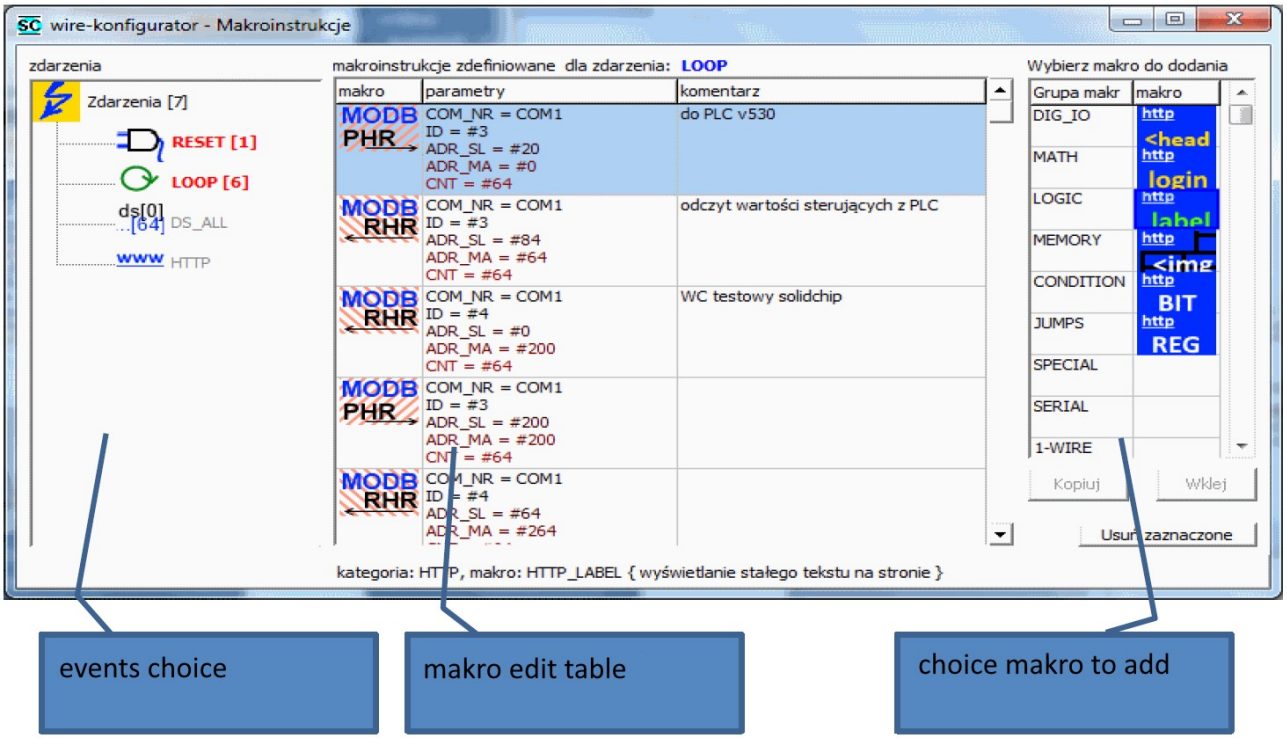
- **RESET** – macros defined in this group are executed after each module reset;
  - Useful when you need to initialize a resource (register value, bit value, serial port operating mode, etc.)
- **LOOP** – macros within the event are executed cyclically (when no other activities are performed);
- **DS_ALL** – generated after each reading of all temperatures from all temperature sensors connected to the 1-wire bus;
  - Can be used to take readings from elements other than temperature sensors - e.g. reading values measured by an analog-to-digital converter (ADC) type DS2438;
- **HTTP** – generated after receiving a page transfer request from a web browser.



Macro instruction configuration

To configure macro instructions, select the "macro" tab in the wire-configurator program. The main program window will show defined variables and constants that can be used in macros. The program will also open a separate macro instruction configuration window. In this window, you can select the event (on the left side of the window) for which macros will be configured (table in the middle of the window). The table on the right side of the window allows you to select a macro to add to the currently edited event.

Below is a view of the macroinstruction configuration window



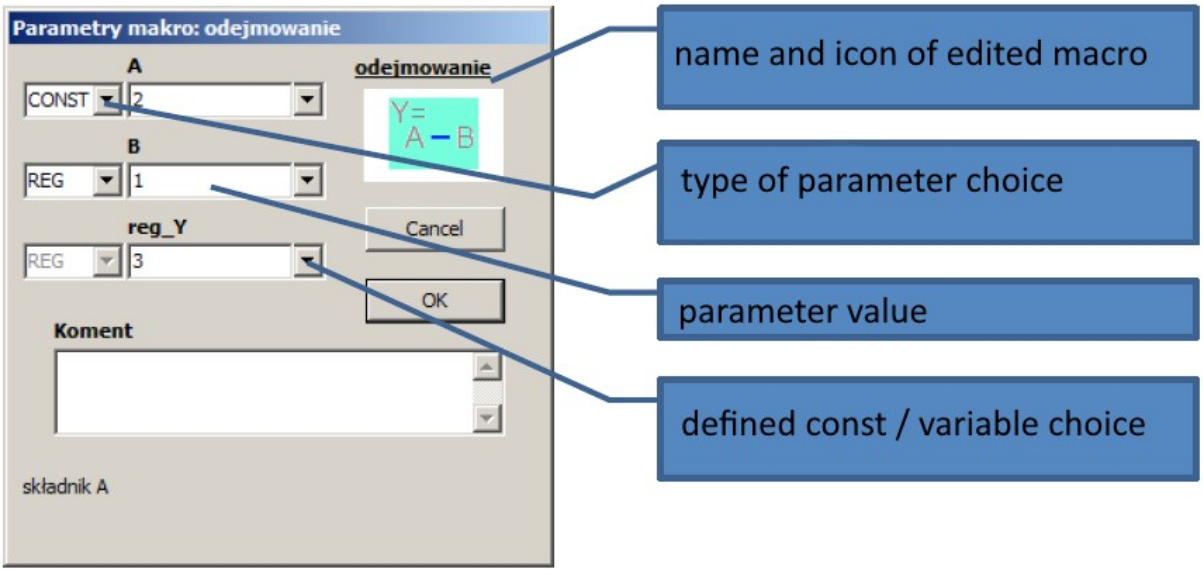
The programmer can enter parameters when adding a macro - a window with parameters will be opened automatically. The parameters can be changed later - after double-clicking on the selected macro, a window with parameters will open.

Adding macro instructions to the selected event

- click on a macro group (table on the right) - available macros from a given group will be shown;
- click the macro to add (select the macro to be added for the event being edited);
- click in the macro editing table to indicate the row before which the macro should be added to the event;

Editing macroinstruction parameters

Macros are configured using parameters. After adding a macro to the event, or after double-clicking a macro defined in the event, a window with parameters will open.
Below is the appearance of the macroinstruction parameters window:



Types of macroinstruction parameters

Depending on the type of macroinstructions, they may need different types of configuration parameters. For some parameters you can select one of the available types.
Available parameter types:

- **REG** – 16-bit register with sign – indication of the register number from which data is to be downloaded (to which data is to be sent);
- value range of register number 0..2047;
- **CONST** – constant – integer; if a CONST type parameter is selected instead of REG, the value of the parameter to be processed is not taken from the indicated register, but entered permanently; CONST appears only as an input parameter to a macroinstruction;
- value range: -32768 … + 32767;
- **BIT** – means the value of one of the bits of the indicated 16-bit register; to define BIT, enter the register number, dot, and bit number (REG.bit) in the parameter edit field; e.g. "8.0" means bit no. 0 from register 8;
  value range: 0.0 .. 2047.15;
- **LABEL** – jump destination - this is a text with a maximum length of 10 characters, containing letters and numbers, or a dot cannot begin with a digit; each jump destination must have a unique name;
- **IN** – digital input number; used by functions from the DIG_IO group;
- value range: depending on the module, e.g. 0..4
- **OUT** – digital output number; used by functions from the DIG_IO group;
- value range: depending on the module, e.g. 0..4
- **NR_DS** – number of the element installed on the 1-wire bus; a parameter of this type can be used by macros from the 1-WIRE group;
  value range: 0..63;
- **TEXT** – text field with a maximum length of 50 characters;
- **IP** – IP address;

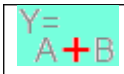
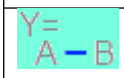
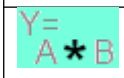
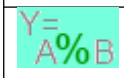
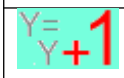
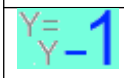
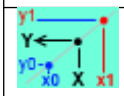
Description of individual macroinstructions

Please use the macro instructions described in the documentation. If your configuration software provides other macro instructions that are not described in this documentation, please contact us for support.

Macros from the DIG_IO group – digital input s/ outputs (optional equipment)

	NAME - DESCRIPTION: „**zapis 1wyjścia**” - writing one digital output; ACTION: the value of the indicated bit is transferred to the latch (remembered until a new value is provided or reset) of the indicated digital output; PARAMETERS: • **bit** – bit number from which the value is to be taken; • **out.nr** – number of the digital output to which the value of the indicated bit should be transferred;
	NAME - DESCRIPTION: „**odczyt wejść**” - reading of all digital inputs; ACTION: enter the values of digital inputs into the bits, starting from the indicated bit ; PARAMETERS: • **bit** – bit number from which values are to be taken for subsequent outputs;
	NAME - DESCRIPTION: „**odczyt 1wejścia**” - reading one digital input; ACTION: enter the value read from the indicated digital input into the indicated bit; PARAMETERS: • **bit** – bit number to which the value is to be written; • **in.nr** – number of the digital input from which to copy the value to the indicated bit;

Macros from the MATH group – mathematical functions

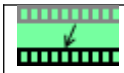
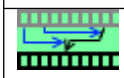
	NAME - DESCRIPTION: „**dodawanie**" – adds two integers; ACTION: the result of adding A + B is entered into the Y register; PARAMETERS: • **A** – (parameter type CONST or REG); • **B** – (parameter type CONST or REG); • **reg_Y** – address of the register into which to enter the result of the operation (REG parameter type);
	NAME - DESCRIPTION: „**odejmowanie**" – odejmowanie dwóch liczb całkowitych; ACTION: wynik odejmowania A - B wpisuje do rejestru Y; PARAMETERS: As for addition;
	NAME - DESCRIPTION: „**multiply**" – multiplying two integers; ACTION: the result of multiplying A * B is written into the Y register; PARAMETERS: As for addition;
	NAME - DESCRIPTION: „**divide**" – dividing two integers; ACTION: the result of the A/B division is entered into the Y register; PARAMETERS: As for addition;
	NAME - DESCRIPTION: „**modulo**" – the remainder from dividing two integers; ACTION: puts the remainder of the A/B division into the Y register; PARAMETERS: As for addition;
	NAME - DESCRIPTION: „**inkrementacja**" – increasing the indicated register by 1; ACTION: takes the value of a register, increments it by 1 and writes it to the same register; PARAMETERS: • **reg** – registry no;
	NAME - DESCRIPTION: „**dekrementacja**" – decreasing the indicated register by 1; ACTION: takes the value of a register, decreases it by 1 and writes it to the same register; PARAMETERS: • **reg** – registry no;
	NAME - DESCRIPTION: „**linear**" – calculating proportional value; ACTION: calculates the output value Y in such proportions to Y0 and Y1 as the input value X is to X0 and X1; The result is calculated according to the relationship **reg_Y = (X – x0) * (y1- y0)/(x1 – x0) + y0;** PARAMETERS: • **X** – input value; (REG or CONST); • **x0**, **x1**, **y0**, **y1** – linearization parameters; (REG or CONST); • **reg_Y** - address of the register into which to enter the result of the operation (REG only);

LOGIC macros – logical functions

	NAME - DESCRIPTION: „**SET_BIT**" – set bit; ACTION: sets the indicated bit to 1; PARAMETERS: • **bit** – bit number to set;
	NAME - DESCRIPTION: „**RST_BIT**" –reset bit; ACTION: resets the indicated bit (the bit value is 0 after executing the macro); PARAMETERS: • **bit** – bit number to set;

	NAME - DESCRIPTION: „**TOGGLE_BIT**” – bit negation; ACTION: changes the value of the indicated bit to the opposite; PARAMETERS: • **bit** – bit number to change the value;

Macros from the MEMORY group - memory operations

	NAME - DESCRIPTION: „**MOV**” – writing values to the register; ACTION: writes a constant value or the contents of another register to a register; PARAMETERS: • **A** – value to be entered into the register; (REG or CONST); • **reg_Y** - address of the register into which to enter the new value;
	NAME - DESCRIPTION: „**MOV_INDIRECT**” – rewriting register values to other, indirectly indicated registers; ACTION: retrieves values from source registers shifted by offset; places values in target registers shifted by offset; PARAMETERS: • **X** – number of the first register to download the value; (REG only); • **ofs_X** – moving the first register from which to download data to another register; (REG or CONST); X + ofs_X = first register to download data; • **reg_Y** – number one from the target registers; (REG only); • **ofs_Y** – target register offset; (REG or CONST); • **count** – number of registers to be rewritten; (REG or CONST);
	NAME - DESCRIPTION: „**FILL_REG_ARRAY**” – filling registers with value; ACTION: writes a constant value or the contents of another register to the indicated group of registers; PARAMETERS: • **val** – value to be entered into the registers; (REG or CONST); • **dest** - address of the first register into which the new value should be entered; (REG only); • **count** – number of registers to be saved; (REG or CONST);

Macros from the CONDITION group – checking conditions

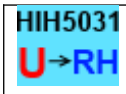
	NAME - DESCRIPTION: „**compare if A==B**” – set bit if equal; ACTION: sets a bit (value 1) if the compared registers have equal values; clears the bit if the values are different; PARAMETERS: • **A** – the first of the compared values; (REG or CONST); • **B** – the second of the compared values; (REG or CONST); • **bit** – bit number into which to enter the result of the macroinstruction;
	NAME - DESCRIPTION: „**compare if A!=B**” – set bit if different; ACTION: sets a bit if the compared registers have different values; clears the bit if the values are the same; PARAMETERS: as for "compare if A==B";
	NAME - DESCRIPTION: „**compare if A>B**” – set bit if A > B; ACTION: sets the bit if register A has a value greater than register B, otherwise clears the bit; PARAMETERS: as for "compare if A==B";
	NAME - DESCRIPTION: „**compare if A>=B**” – set bit if A >= B; ACTION: sets the bit if register A has a value greater than or equal to register B,

	otherwise clears the bit; PARAMETERS: as for "compare if A==B";
	NAME - DESCRIPTION: „**compare if A<B**” – set bit if A < B; ACTION: sets the bit if register A has a value less than register B, otherwise clears the bit; PARAMETERS: as for "compare if A==B";
	NAME - DESCRIPTION: „**compare if A<=B**” – set bit if different; ACTION: sets the bit if register A has a value less than or equal to register B, otherwise clears the bit; PARAMETERS: as for "compare if A==B";

Macros from the JUMPS group - program flow control

	NAME - DESCRIPTION: „**label**” – jump target; ACTION: taken as a jumping target by jumping macros; PARAMETERS: • **val** – unique name of the jump target;
	NAME - DESCRIPTION: „**jump**” – jump; ACTION: jumps to the target (i.e. to the macro called "label"); PARAMETERS: • **label** – indication of the jump target (values of the "val" parameter of the label macro);
	NAME - DESCRIPTION: „**jump if BIT**” – jump if bit is 1; ACTION: jumps to the target if the indicated bit is equal to 1; PARAMETERS: • **bit** - bit number to check the value; • **label** – bit number to check the value;
	NAME - DESCRIPTION: „**jump if NOT BIT**” – jump if bit is 0; ACTION: jumps to the target if the indicated bit is equal to 0; PARAMETERS: as for "jump if BIT";

Macros from the SPECIAL group - special - conversion, complex analysis of measurement results

	NAME - DESCRIPTION: „**RH_HIH5031**” – calculation of relative humidity based on signals from the HIH5030 / HIH5031 sensor; ACTION: calculates relative humidity measured by the HIH-5030 or HIH-5031 sensor; the result is a value in the range 0..100 meaning 0..100% humidity; This function should be called after reading the measurement values from the sensor (e.g. using the DS2438_CONVIV macro); PARAMETERS: • **Urh** –register number containing information about the output voltage of the HIH sensor; (REG only); • **Uzas** – register number indicating the supply voltage of the HIH sensor; (REG only); • **Temp** – register number indicating the temperature value of the HIH sensor; (REG only); • **rej_RH** – register number to which to return the calculated RH (Relative Humidity) value; (REG only);

Macros from the SERIAL group - communication via COM ports (serial)

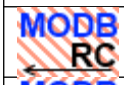
NAME - DESCRIPTION: „**PORT_INIT**" – serial port initialization (RS-485, RS-232);
ACTION: sets serial port parameters;
This function should be placed in the RESET event;
PARAMETERS:

- **COM_NR** -selection of serial port COM1 or COM2 (CONST);
- **TRYB** - selection of operating mode (CONST):
  - MODB_SL (default) – MODBUS RTU SLAVE protocol;
  - or MASTER (the protocol may be different from MODBUS RTU in the future);
- **ID** - Module ID – important if the MODB_SL port operating mode
- **FORMAT** – data frame format (CONST), you can choose from:
  - 8N1 – 8-bit data, no parity bit, 1-stop bit;
  - 8O1 – 8-bit data, ODD parity bit, 1-stop bit;
  - 8E1 – 8-bit data, EVEN parity bit, 1-stop bit;
- **BAUDRATE** – baud rate (CONST) available rates (9600, 19200, 38400, 57600, 115200 bps);
- **TIMEOUT** – maximum waiting time for a response (after which an error is indicated in the event of no response)(CONST) unit [ms], maximum value 10000ms; recommended value 300..500 ms;
- **RETRY** – number of repetitions in case of errors (CONST);
- **STATUS** – register of the status of operations performed on the selected serial port;
  bit meaning:
  - bit. 0 - STAT_BUSY - busy executing a macro (for a given port);
  - bit. 1 - STAT_DONE - macro taken;
  - bit. 2 - STAT_ERR - macro execution error;
  - bit. 3 - STAT_TMOUT - error - no response;
  - bit. 4 - STAT_ERR_CRC - CRC error of response from SLAVE device;
  - bit. 5 - STAT_ERR_SL - SLAVE responds with error information (e.g. wrong address range, wrong amount of data, etc.);
  - bity 6 – 15 – irrelevant;

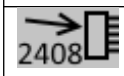
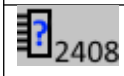
NAME - DESCRIPTION: „**MODB_RHR**" (Read Holding Registers – 0x03) – reading 16-bit registers from the SLAVE device using the MODBUS RTU protocol;
ACTION: the function reads the contents of registers from the SLAVE device via the selected and previously initiated serial port;
This function should be placed in the LOOP event;
PARAMETERS:

- **COM_NR** -selection of serial port COM1 or COM2 (CONST);
- **ID** – SLAVE module ID number; (REG or CONST); allowed values 0..255;
- **ADR_SL** – address of the register to be read from the SLAVE device;
- **ADR_MA** – address of the register into which to enter the data read from the SLAVE device in WIRE-CHIP
- CNT – ilość rejestrów danych do odczytania z urządzenia SLAVE; zakres 1..64 [rejestry];

NAME - DESCRIPTION: „**MODB_PHR**" (Preset Holding Registers – 0x10) – writing 16-bit registers in the SLAVE device using the MODBUS RTU protocol;
ACTION: the function writes the contents of the registers to the SLAVE device via the selected and previously initiated serial port;
This function should be placed in the LOOP event;

	PARAMETERS: • **COM_NR** -selection of serial port COM1 or COM2 (CONST); • **ID** – SLAVE module ID number; (REG or CONST); allowed values 0..255; • **ADR_SL** – address of the register in the SLAVE device from which to start saving data; • **ADR_MA** – address of the register in WIRE-CHIP from which to download data for writing to the SLAVE device; • **CNT** –number of data registers to be written to the SLAVE device; range 1..64 [registers];
	In preparation – MODBUS READ COILS – reading bits from another device
	In preparation – MODBUS FORCE COILS – writing bits to another device

Macros from the 1-WIRE group - communication via 1-wire

	NAME - DESCRIPTION: „**DS2438_CONVIV**" – performs measurements with the DS2438 transducer; ACTION: gives the "measure" command to the DS2438 transducer and then reads the measurement results; The measurements include the voltage at the voltage input (Vad - pin.4), the supply voltage (Vdd - pin.5) of the DS2438 system and the signal at the current measurement input (pins 2 and 3 - Vsens+ and Vsens-); Vad and Vdd measurements are performed with a resolution of 10mV, Vsens measurement is performed with a resolution of 0.2441mV; More information in the DS2438 documentation; This function should be placed in the DS_ALL event; PARAMETERS: • **nr_DS** – item number DS2438 on 1-wire bus; values 0..63; • **rej_out** – register number in which to place the first measurement value (Vad); subsequent measurement values (Vdd and Vsens) are placed in subsequent registers (rej_out+1 and rej_out+2);
	NAME - DESCRIPTION: „**DS2408_WR**" – writes value to all (8) DS2408 outputs; ACTION: sends commands to DS2408 to write data to digital outputs; The outputs are of the Open-Drain type (entering 0 causes the output to be short-circuited to GND, entering 1 causes the output transistor to open - no short-circuit); This function should be placed in the DS_ALL event; PARAMETERS: • **nr_DS** – item number DS2408 on 1-wire bus; values 0..63; • **data** – register number with data or constant to be sent to digital outputs (REG or CONST);
	NAME - DESCRIPTION: „**DS2408_AL**" – reading the Activity Latch register from DS2408; ACTION: Changing the state of the DS2408 digital input sets the bit value in the "Activity Latch" register. Thanks to this, even when reading the states of the system's inputs rarely (once a second), you can recognize that they have changed in the meantime (despite returning to the state from the last reading). After reading the AL register from DS2408, bits with the value 1 in AL are set in the resulting register (rej_out). The bits in rej_out corresponding to the bits read from AL are not reset. with the value 0. The bits should be reset by entering through an external device (e.g. RS-485 port with MODBUS RTU protocol) or through another macroinstruction. This function should be placed in the DS_ALL event; PARAMETERS:

	• **nr_DS** – item number DS2408 on 1-wire bus; values 0..63; • **rej_out** – register number in which to enter the result of reading from AL. (REG);

Macros from the ETHERNET group - communication via a computer network

	NAME - DESCRIPTION: „**ETHERNET_INIT**" – initialization of the ETHERNET port and the TCP/IP and MOSBUS TCP protocol stack;
	ACTION: sets basic port parameters;
	This function should be placed in the RESET event;
	PARAMETERS:
	• **DHCP** – selection of automatic download of network parameters (IP, MASK, GATE)
	  ○ Not available in the current firmware version – enter 0;
	• **IP** – IP address of the WIRE-CHIP module;
	• **MASK** – subnet mask;
	• **GATE** – IP address of the default gateway;
	• **KEEP.ALIVE** – checking for an inactive TCP connection (the connection is established, but no data has been exchanged for some time);
	• A negative check result (data cannot be exchanged) results in the connection being severed and allows remote clients to re-establish the connection
	  ○ the parameter value means the number of seconds from the last activity to the connection test;
	  ○ a value of 0 means that the connection will not be tested;
	• **MODB.PORT** – TCP port number for the MODBUS TCP protocol – usually = 502;
	• **HTTP.PORT** – TCP port number for http protocol – usually = 80;

Macros from the HTTP group - generating a website - mini WEB SERVER

HTTP macros should only be placed in the HTTP event.
The website reloads automatically every 120s.
Only 1 user can be logged in to the website at a time (several people can view the website).

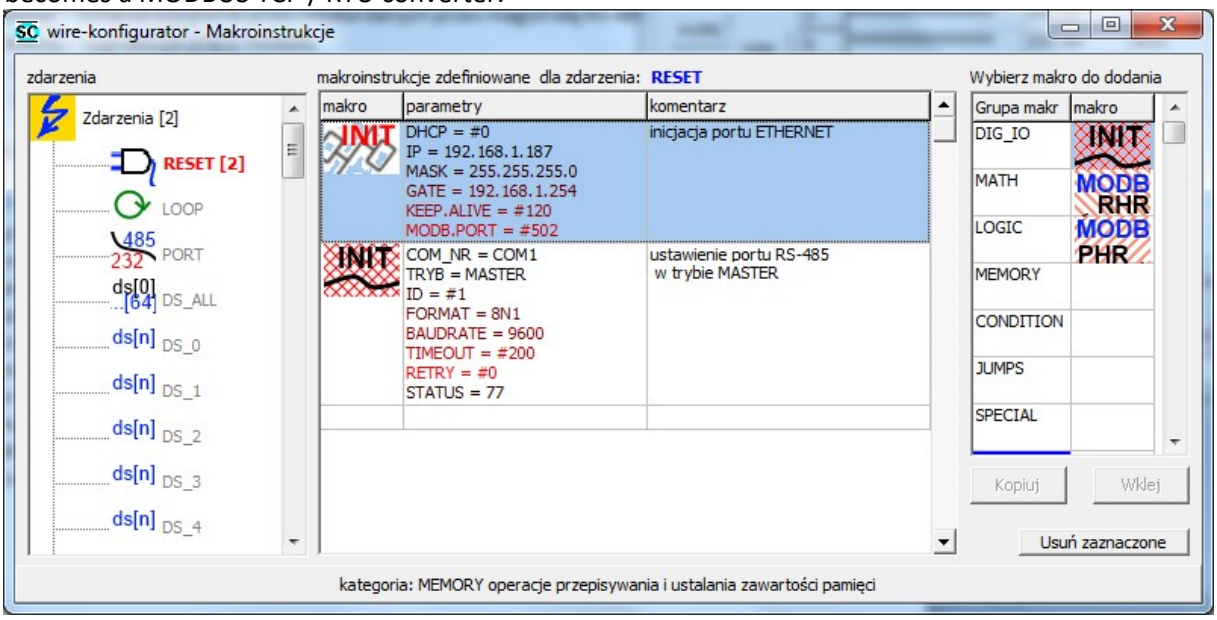
	NAME - DESCRIPTION: „**HTTP-LOGIN**" – simple protection of the website against unauthorized modification of bit/register values; ACTION: If the "HTTP-LOGIN" macro is used, a password entry field and a "LOGIN" button are shown in the upper right corner of the page. After entering the correct password and pressing the "LOGIN" button, it is possible to modify the bit values (http_BIT macro) - clicking on the bit on the website. If the "HTTP-LOGIN" macro is missing, it is not possible to modify the bit values (via a web browser); Note - the connection is not encrypted, the password is easy to intercept using programs such as WireShark; Enabling control via a website requires great caution. PARAMETERS: • **PASS** – password (max 9 characters); • **ALLOW_VIEW** – reserved;
	NAME - DESCRIPTION: „**HTTP_LABEL**" – displays short text on the page; ACTION: displays fixed text in the indicated place; PARAMETERS: • **LEFT** – distance from the left side of the web browser window - absolute coordinate of the text position on the screen; • **TOP** – distance from the top of the web browser window - absolute coordinate of

	the text position on the screen; • **WIDTH** – text field width; • **COLOR** – text color (possible colors as in HTML e.g. "black", "#123456") • **BG_COLOR** – background color of the text box; • **TXT** – display text (maximum 50 characters);
	NAME - DESCRIPTION: „**HTTP_IMG**" – displays an image on the page; ACTION: displays an image downloaded from an external server in a specified place on the page; PARAMETERS: • **LEFT** – distance from the left side of the web browser window - absolute coordinate of the text position on the screen; • **TOP** – distance from the top of the web browser window - absolute coordinate of the text position on the screen; • **URL –** image address (maximum 50 characters)
	NAME - DESCRIPTION: „**HTTP_BIT**" – displays ON/off value on the page; ACTION: in the indicated place on the page, displays the value of the sum of bits from the indicated register, possible displayed values are ON (sum of bits 1) or off (sum of bits 0); If the web browser is logged in, after clicking the result of the "HTTP_BIT" macro, when the value shown is off, the bits covered by the mask (MASK parameter) are set, and when the value shown is ON, all bits are cleared; PARAMETERS: • **LEFT** – distance from the left side of the web browser window - absolute coordinate of the text position on the screen; • **TOP** – distance from the top of the web browser window - absolute coordinate of the text position on the screen; • **WIDTH** – width of the text field; • **COLOR** – text color (possible colors as in HTML e.g. "black", "#123456") • **BG_COLOR** – background color of the text field; • **HINT** – text displayed when you hover the cursor over the text field (maximum 50 characters); • **REG** – register number whose bit values will be used • **MASK** – a constant value that is bitwise summed with the register value, the result of the summation shown on the page;
	NAME - DESCRIPTION: „**HTTP_REG**" – displays the registry value on the page; ACTION: in the indicated place on the page, displays the value of the indicated register with the REG number divided by the constant DIV value; PARAMETERS: • **LEFT** – distance from the left side of the web browser window - absolute coordinate of the text position on the screen; • **TOP** – distance from the top of the web browser window - absolute coordinate of the text position on the screen; • **WIDTH** – width of the text field; • **COLOR** – text color (possible colors as in HTML e.g. "black", "#123456") • **BG_COLOR** – background color of the text field; • **HINT** – text displayed when you hover the cursor over the text field (maximum 50 characters); • **REG** – register number whose bit values will be used • **DIV** – constant value by which the value read from the register is divided

Examples of using macro instructions

Configuring the module as a MODBUS TCP / RTU converter

The figure below shows an example configuration of the WIRE-CHIP module with an Ethernet port - the first macroinstruction, and after adding the second macroinstruction "PORT_INIT", the module becomes a MODBUS TCP / RTU converter.



Signaling

The module has the following LEDs to indicate the device status (in order - from top):

- **USB** (yellow) – lighting indicates that the module is connected to the computer;
- **ERR** (red)
  - indicates a problem with the execution of macro instructions or
  - firmware error while bootloader is active
- **ERRw** (red) – indicates a problem with the 1-wire bus;
- **WIRE** (green) – means the implementation of communication on a 1-wire bus;
- **485** (green) – means the module sends data via the RS-485 bus
- **RUN** (green) – signals module operation
  - 1 Hz – firmware program operation (normal operation)
  - 3 Hz – bootloader work

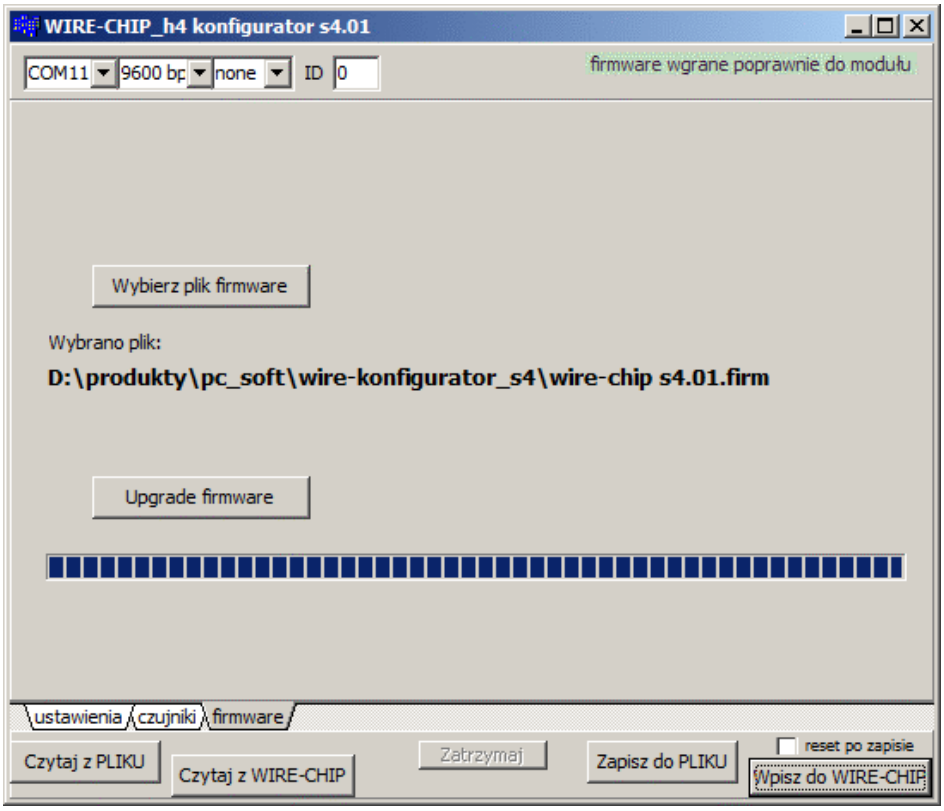
All diodes are turned on after powering up (restarting) the module for 1 second.

Module firmware upgrade

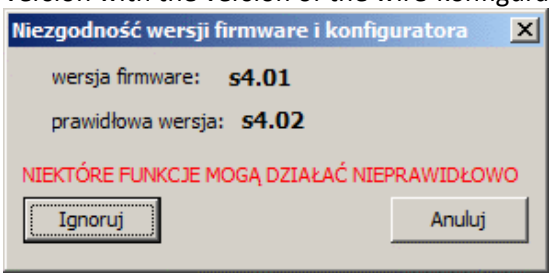
This function may be useful if a newer version of the module's built-in software has been created (e.g. a new, necessary functionality has been added). If you downloaded the latest version of the configuration software, which is newer than the firmware file contained in the module, then the software will report an incompatibility. Also in this case, it is suggested to replace the module firmware file.

To replace the software, go to the "firmware" tab. Select the appropriate firmware file by pressing the button. Make sure the module is connected to the computer and the communication settings are correct. Press the "Upgrade firmware" button. A progress bar will inform you about the current status of the software upload.

Below is the appearance of the wire-konfigurator program window after uploading the firmware file.



Below is the appearance of the window informing about the incompatibility of the module firmware version with the version of the wire-konfigurator configuration program:



Terms of warranty

1. The equipment is covered by a one-year warranty from the date of sale;
2. Warranty repairs are carried out only at the SolidChip headquarters;
3. The equipment for repair is delivered at the customer's expense, and after repair it is returned at SolidChip's expense;
4. We do not bear the costs of dismantling and reassembling equipment in the facility (these costs are the responsibility of the client);
5. When sending the equipment to the service center, please include a detailed description of the damage. Failure to provide a detailed description of the damage may result in an extension of the repair time. We reserve the right to charge a fee of 20% of the price of a new device if the equipment is sent to the service center without a detailed description of the damage;
6. Warranty repairs are usually completed within 2 weeks. If it is not possible to complete the repair within this time, the customer is informed about the estimated repair time;
7. The warranty does not cover mechanical damage or damage resulting from improper use, in particular damage resulting from exceeding the permissible ranges of input signals;
8. The customer is asked to contact us by phone before the equipment is delivered for repair (it may be possible to provide telephone assistance);
9. The cost of post-warranty repair includes the service fee (25% of the price of the new device), the cost of spare parts and shipping costs. If the total costs exceed 50% of the price of the new device, the customer is informed about this fact;
10. In the case of "out of warranty" equipment - if the customer provided a description of the damage, the device inspection is free of charge. The customer is informed about the profitability of the repair and decides whether to perform a post-warranty repair or not. Shipping of equipment at the customer's expense;
11. SolidChip makes every effort to ensure high quality of the equipment offered. We are not responsible for losses or lack of profits resulting from equipment malfunction or inappropriate use of the devices provided by us;
12. The equipment may not be used for applications that depend on human life (e.g. medical).

Producent

SolidChip
Ul. Olszowa 4, Niedziałka Druga, 05-300 Mińsk Mazowiecki, POLAND
www.solidchip.eu
biuro@solidchip.eu