AUTOMATYKA ELEKTRONIKA INFORMATYKA PRZEMYSŁOWA

Series devices

iMod / TRM

User's Manual

Document version	**Creation Date**	**Changes**
1.0.0	27.04.2010	Base version
1.01.2000	1.06.2011 r.	Version 2 (1wire, Mbus, WWW, chapter, Basic commands, XML Tutorial, Added chapter 8., Appendix A, Appendix B, Appendix C,)

For further information on work with the device, consult the following documents:
NPE Instructions
NXDynamics Instructions

Table of contents

Chapter 1 Introduction

1.1. About document

This documentation relates to a series of NPE devices based on iMod platform.

Device version

Functionality of the iMod entirely depends on the configuration of the device. Some very simple configuration that can be customized by the user are described in Quick Start.

TRM Series represents access to a specific configuration and additional protocols added as extended the base of iMod functionality. The new functionality is also defined in a consistent manner using the iMod platform configuration file. The specific configuration for the series converters Chapter 6 describes the TRM Tutorial *.

In the rest of the document when it adheres to device that means NPE device running iMod platform.

Some of the hardware resources embedded devices, such as GSM / GPRS modem only occur in some versions of the device - which is why these places were used in the option star *.

Newest version of documentation

The iMod is a product that is being dynamically developed. Thanks to this, new device functions can be added by updating the software. A new version of the documentation is released with each software update.

The newest version of the documentation can be found on the website www.a2s.pl, or on the ftp server: ftp.a2s.pl. The login and password to the server can be found on the iMod DVD.

1.2. About the device

The iMod is a telemetric module with an innovative solution based on the concept of three channels:

- source-channel
- access-channel
- message-channel

The logic of the device's operation is described in one file – MainConfig.xml.

A TRM device is an appropriately developed iMod platform being an example of a final telemetric solution. The TRM converter for managing heat centers has become a popular device not only within the Polish heat engineering sector.

The system structure of the device is illustrated by the scheme below:

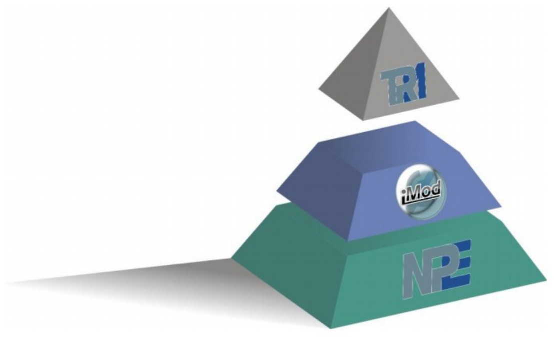


Fig. 1 iMod device application structure

1.3. Introduction to the iMod platform

Channel based

iMod platform is defined as a system of independent channels. There are three types of channels:

- source channel
- access channel
- message channel

The data flow in the channels illustrated with the following diagram:

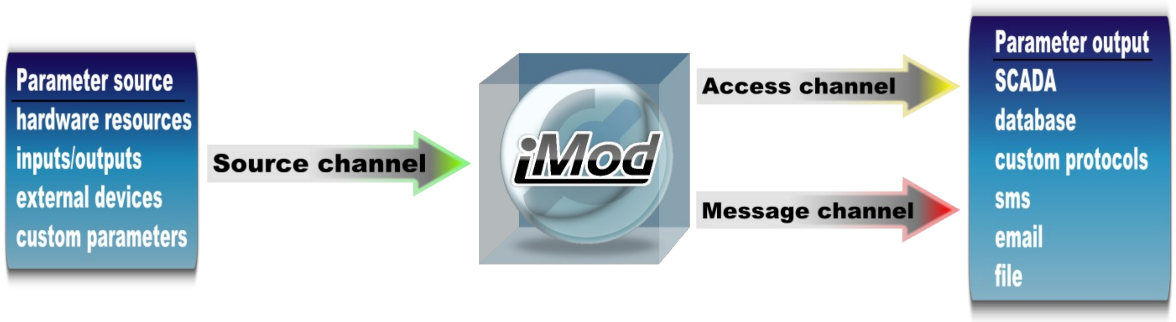


Figure 2: Structure of channels in the iMod platform

iMod platform defines a consistent way to configure. Thanks to XML format file, you can completely change the way the unit operates. This file defines the entire application logic of iMod platform. Detailed information about the configuration are described in subsequent chapters.

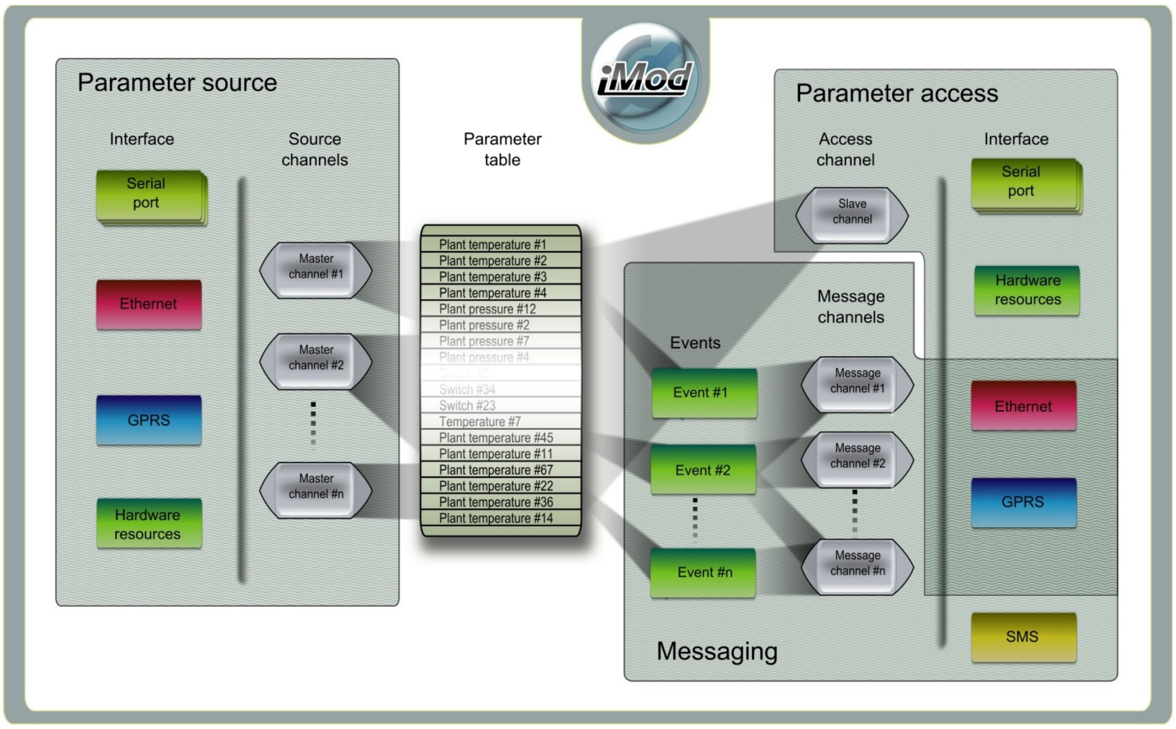
iMod platform has been designed in such a way as to be able to add additional channels (protocols or application logic) as plug-ins. Example of this particular feature are TRM series devices.

Such an implementation of a dedicated protocol ensures that the application works in an orderly manner based on a coherent system. The base functions carried out by iMod are:

- retrieval of data from the NPE interfaces or external devices using the Modbus protocol, MBus or other
- caching data the data retrieved with data logger feature optionally
- sharing of data for monitoring systems such as SCADA, or other devices like iMod or PLC, in a continuous or event mode

Internal structure

Below is an internal schema of iMod platform elements. In addition to the above three groups of channels are shown in the internal table of parameters. This is a cached array of instantaneous values representing the processed data (eg readings inputs), which can then be passed on through output channels.

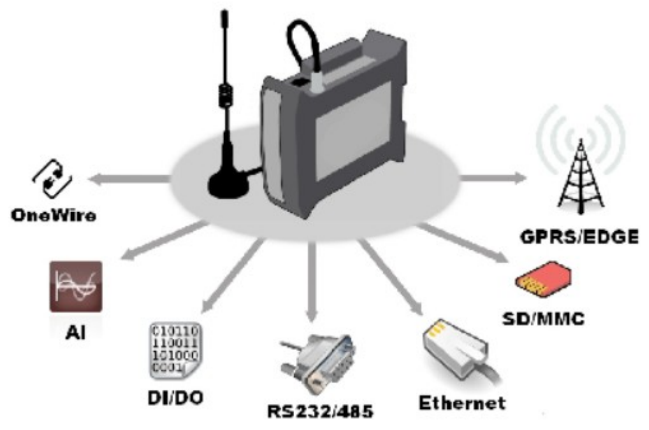


Functional Diagram iMod platform

1.4. About system NPE

As mentioned earlier the device operates on a NPE system, which is consistent with Linux. This means that using the terminal it is possible to access the administration facility in a manner consistent with other devices or computers, based on the Linux system. iMod device also provides set of standard Linux services such as FTP, SSH, SQL, IPTABLES, WWW and so on.

NPE system and iMod software can access the resources of the NPE hardware as shown below.



This figure presents the hardware interfaces, it should be noted that some of them occur only in certain versions of the NPE series devices.

A detailed description of the NPE hardware and Linux beyond the scope of the current document.
Additional Information:

- ➔ NPE - User's Manual
- ➔ Linux Documentation

Chapter 2 Starting device

2.1. Description of the connectors and how to connect

Pin Description

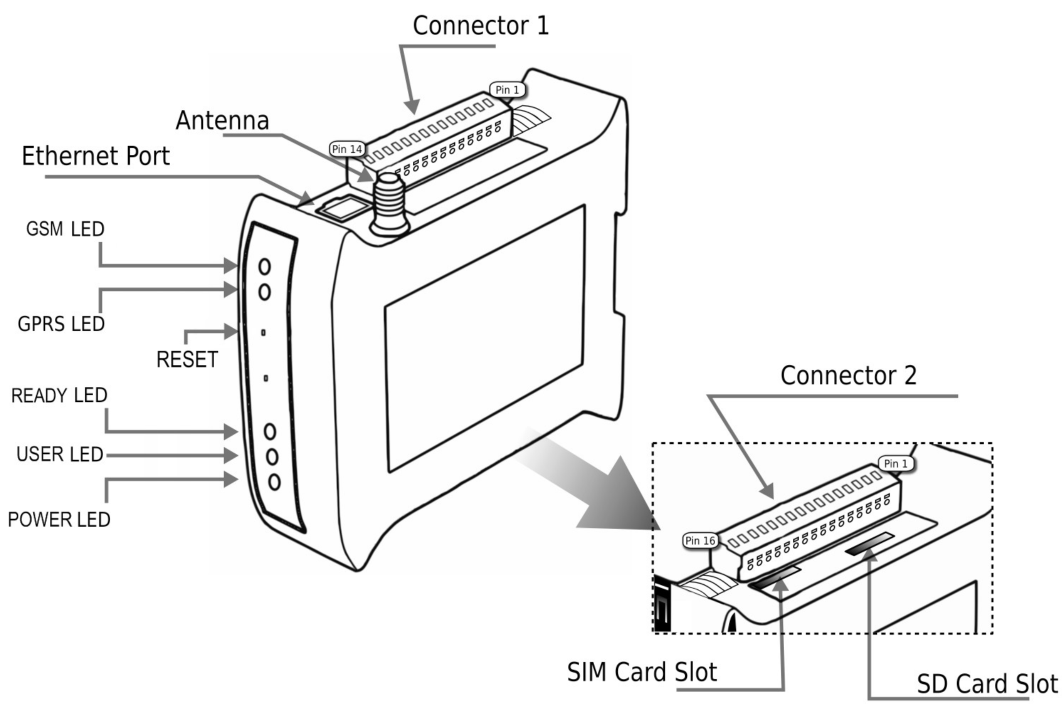
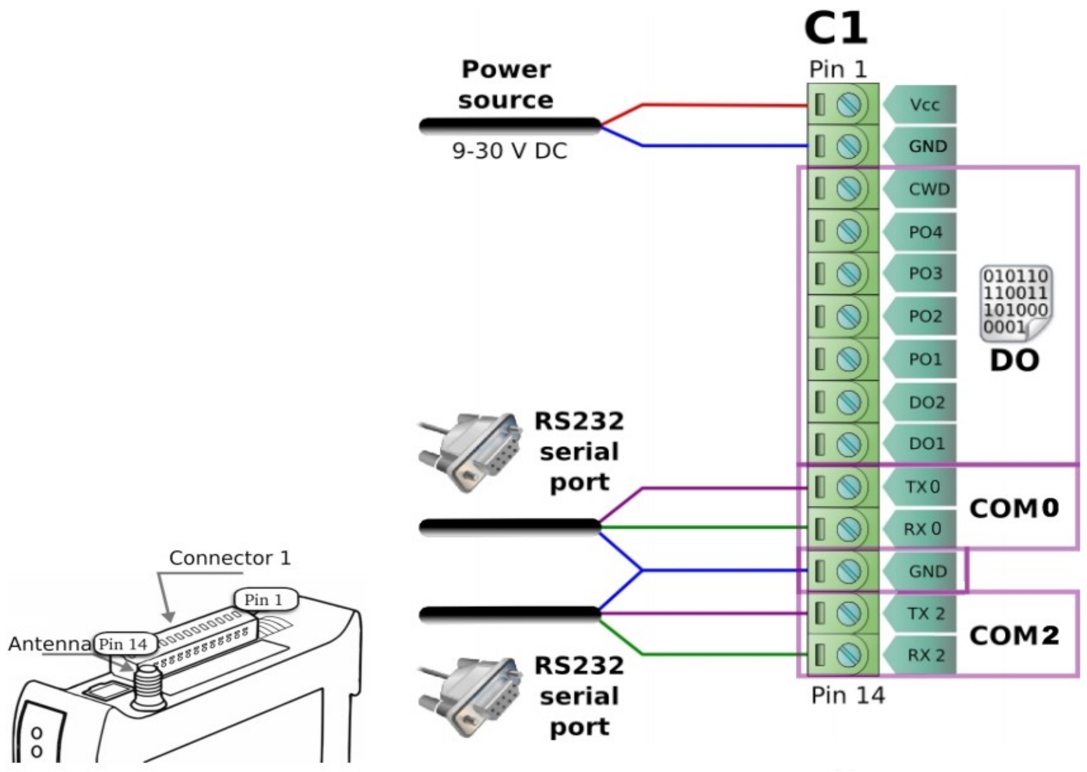


Figure 3 Appliance design

Element name	Label	Opis
Connector 1	C1	Power connector and serial ports. See more »
Connector 2	C2	Connector, analog inputs and RS485 port. See more »
SIM Card Slot*	SIM	SIM card slot
SD Card Slot	SD	SD/MMC card slot
Ethernet Port	ETH	Ethernet RJ45 socket
Antenna*	ANT	Aerial Input Modem
Power LED	PWR	Led indicating power on
User LED	USER	Led for user application purpose
Ready LED	RDY	iMod platform status LED. See more »
GPRS LED	GPRS	GPRS connection status LED. See more »
Modem LED	GSM	Led indicating modem power on

* - depending on the version

Interface description

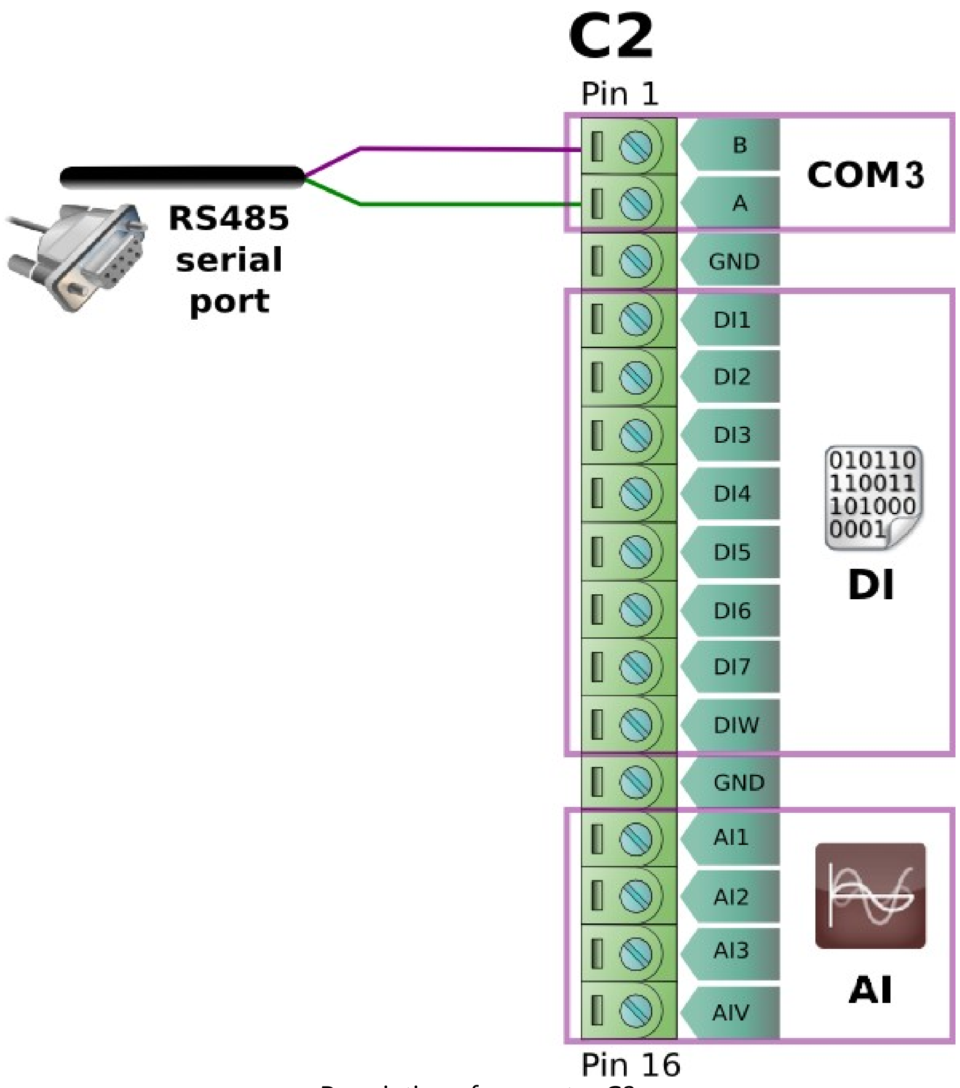


Description of connector C1

Numer pinu	Oznaczenie	Opis
1	Vcc	Power Hi Voltage
2	GND	Power Lo Voltage (ground)
3	CWD	Protection diodes clamp transistor outputs
4	PO4	Open Collector digital output or relay output
5	PO3	Open Collector digital output or relay output
6	PO2	Open Collector digital output or relay output
7	PO1	Open Collector digital output or relay output
8	DO2	Digital Output Open Collector
9	DO1	Digital Output Open Collector
10	TX0	RS232 serial port - COM0 Transmit Data
11	RX0	RS232 serial port - COM0 - ReceiveData
12	GND	Minus power supply (ground)
13	TX2	RS232 serial port - COM2 Transmit Data
14	RX2	RS232 serial port - COM2 - ReceiveData

Tab. 1 Description of the connector pins C1

COM1 is an internal serial port for communication between the device and the modem GSM/GPRS/EDGE.



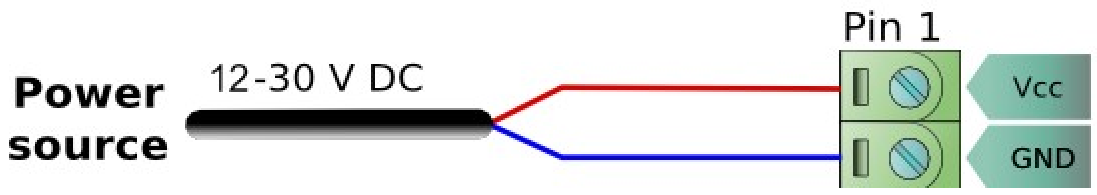
Description of connector C2

Numer pinu	Oznaczenie	Opis
1	B	Serial Port RS485 – COM3 - DATA+
2	A	Serial port RS485 – COM3 - DATA-
3	GND	Ground
4-7	DI1 - DI4	Digital input
8-10	DI5 - DI7	Digital input with break function
11	DIW	OneWire interface or Digital input with break function
12	GND	Ground
13-15	AI1 - AI3	DC analog input
16	AIV	DC or AC analog input

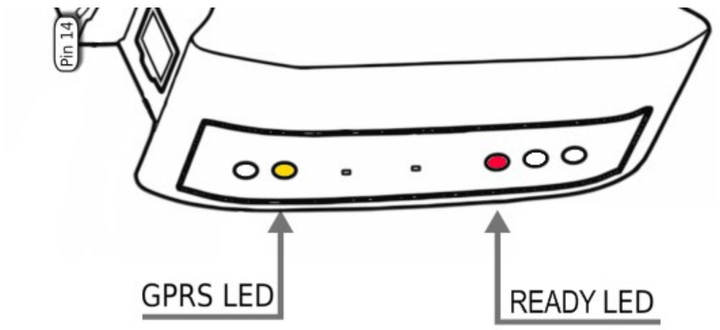
Tab. 2 Description of connector C2

Power Connection

The device is ready for operation immediately after power is supplied. The device needs a power source of 12V DC to 30V with 12W power. The connection of power supply is shown in Figure Error: Source not found the appeal setting out connector C1.



2.2. Signaling LEDs



There are two LEDs on the front panel indicating current device status:

1. iMod platform status LED - READY LED
2. GPRS connection status LED - GPRS LED *

* Available only in versions with built-in modem, GPRS / EDGE

Below is described in detail about signaling modes.

Status of operation

The current status of device operation - iMod application status, is indicated by the Ready LED [RDY].

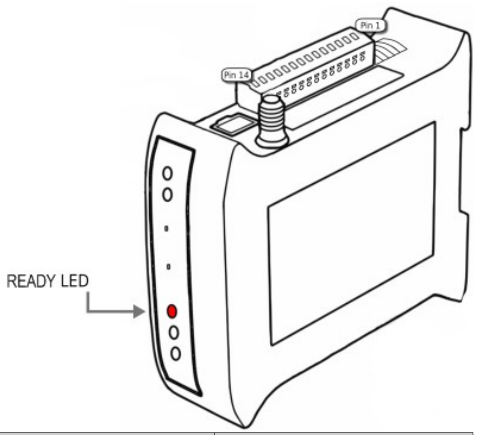
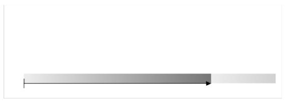
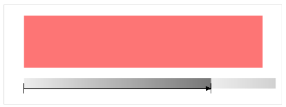
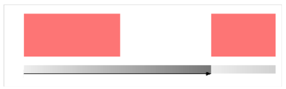
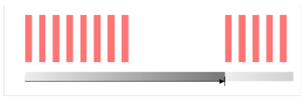
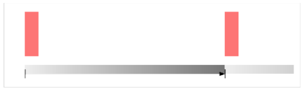
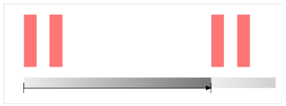


Table 3 shows the behavior of Ready LED depending on the state of the job.

No.	State	Type	Presentation	Description
1	The device inactive	info		LED switched off
2	Initialization	info		Continuous illumination
3	Normal operation	info		Slow blinking (About 1 Hz)
4	Turn off	info		Slow flashing 0.5 Hz modulated fast flashing
5	Initialization error of one of the channels	warning		Repeated single pulse
6	Configuration file error	error		Repeated two short pulses

Tab. 3 Conservation Status LED depending on the status of the device

Warning and error flags prove incorrect job of iMod application and should be eliminated for example by setting the appropriate configuration file. More information the device configuration, see the documentation later.

GPRS connection status[*]

The current status of the GPRS connection is indicated by the LED marked as LED.

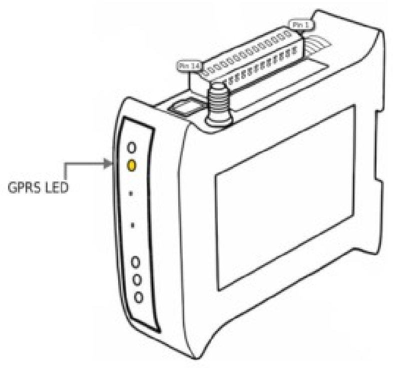
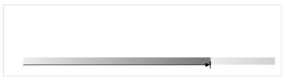
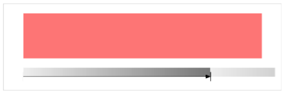
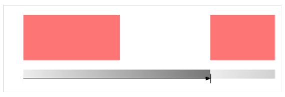
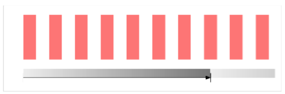
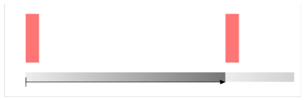
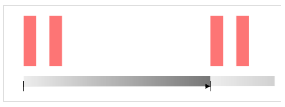
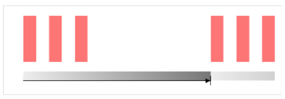


Table 4 shows the importance of each of a sequence of flashing GPRS LED

No.	State	Type	Presentation	Description
1	No connection	info		LED switched off
2	Initialization	info		Continuous illumination
3	Active connection	info		Slow blinking (1Hz)
4	Waiting for registration	info		Fast flashing
5	Error response to a ping	warning		Repeated single pulse
6	Invalid PIN	error		Repeated two short pulses
7	PUK required	error		Repeated three short pulses

Tab. 4 The meaning of GPRS LED

*Applies only to the version with built-in modem, GPRS / EDGE

GPRS connection auto start is enabled by default. Nerveless appropriate parameters in configuration should be applied in most cases in order to work properly.

Error signaled as invalid PIN is described in Table 4 indicates that the PIN code as defined in the file **/mnt/mtd/syscfg** is incorrect. Please verify it and restart the device.

WARNING

After a three-run the converter with the wrong PIN code (setting pin code is disabled by default) typically simcard will be blocked and will be a need to unlock it giving the PUK code. The PUK code can be obtained by using the *gprs puk* command.

2.3. Connecting to a PC

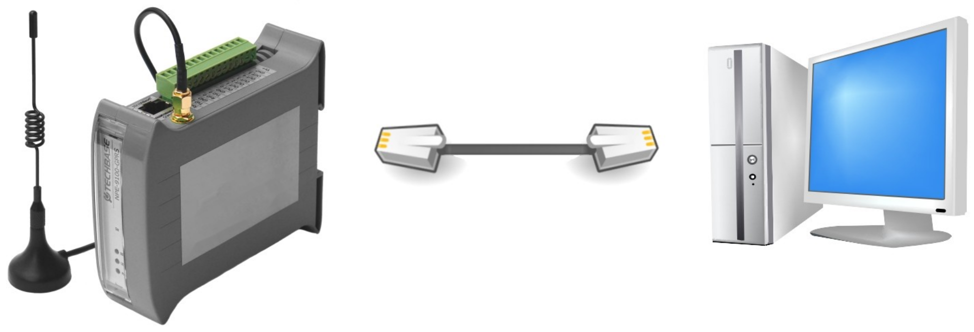
Device can be connected to a computer using the following comm channels:

1. Ethernet
2. Serial port – COM1

Below is step by step tutorial how to establish connection using those channels.

Connecting via the Ethernet

This tutorial describes how to connect to the device time using Ethernet for the first - that is using initial factory settings.



Direct connection to the PC

Default network configuration

Following configuration is set on device by default:

IP address: **192.168.0.101**

Subnet Mask: **255.255.255.0**

This information will be used in the next step.

IP Configuration

The computer (PC) must be the same network subnet as the device. Taking into account the factory setting s of the device (showed above) a computer must have following IP configuration:

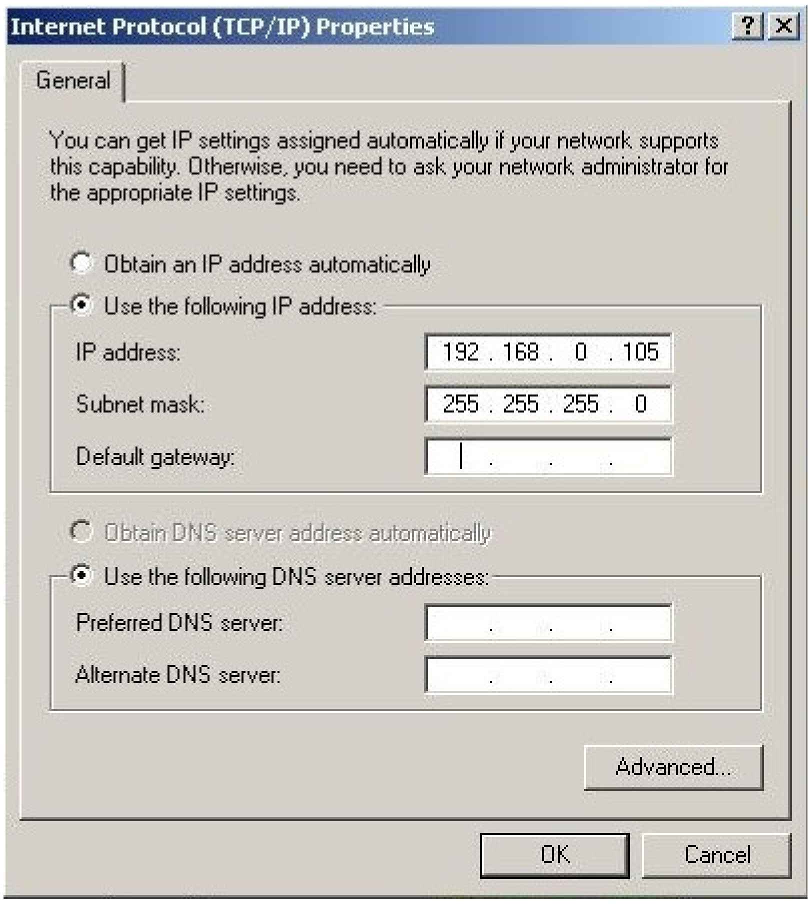
IP address: **192.168.0.X**
Subnet Mask: **255.255.255.0**

where X is any value between 1-244 excluding 101
Examples of valid IP addresses are: 192.168.0.1 and 192.168.0.102

The current tutorial assuming you have factory settings on the device. If you do not know what current network settings are you can:

- restore device to the factory settings
- try to connect using serial port (described in the next chapter)

Below is shown how to set IP configuration in Windows.



Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(●) Use the following IP address:

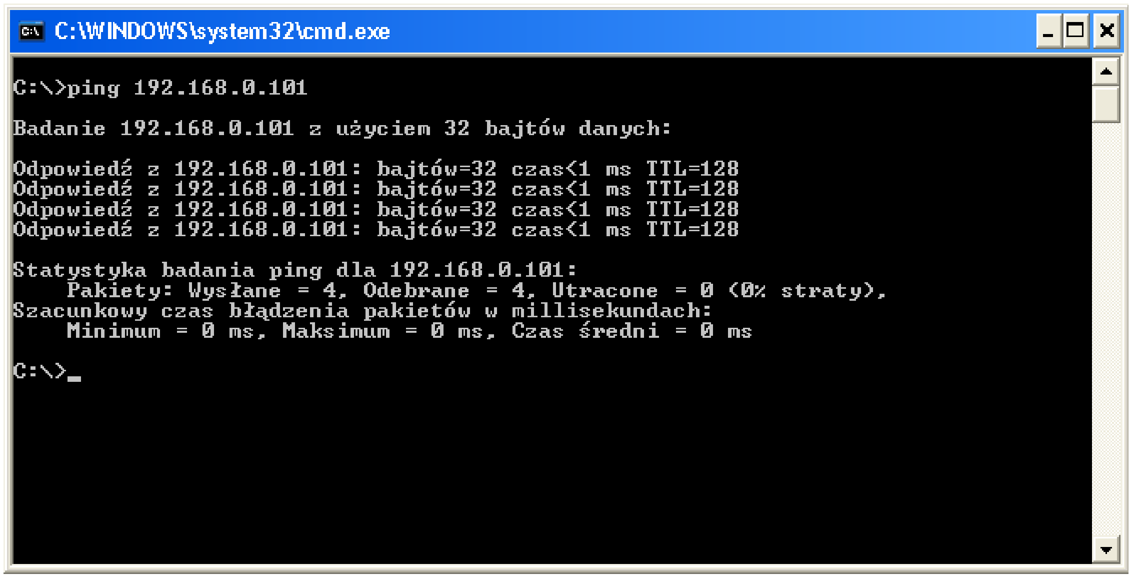
IP address:	192 . 168 . 0 . 105
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

( ) Obtain DNS server address automatically
(●) Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

Test the accuracy of the network connection

Properly connected device should respond to the ping command



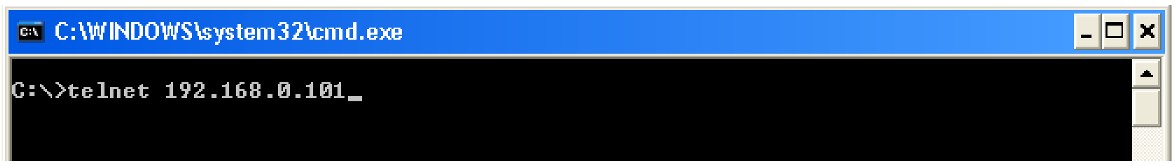
C:\WINDOWS\system32\cmd.exe

```
C:\>ping 192.168.0.101

Badanie 192.168.0.101 z użyciem 32 bajtów danych:

Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128
Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128
Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128
Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128

Statystyka badania ping dla 192.168.0.101:
    Pakiety: Wysłane = 4, Odebrane = 4, Utracone = 0 (0% straty),
Szacunkowy czas błądzenia pakietów w millisekundach:
    Minimum = 0 ms, Maksimum = 0 ms, Czas średni = 0 ms

C:\>_
```

An example ping after proper connection

A connection terminal

To start terminal connection start the following command line command:

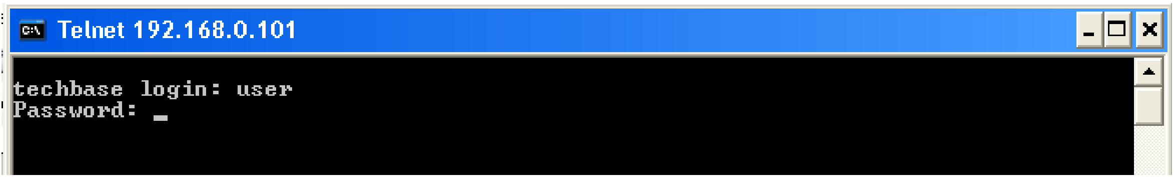


C:\WINDOWS\system32\cmd.exe

```
C:\>telnet 192.168.0.101_
```

Example of connection via telnet

As a result the window should shown:



Telnet 192.168.0.101

```
techbase login: user
Password: _
```

Login Screen telnet

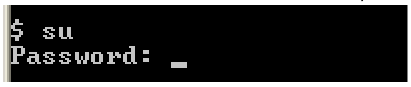
To log in provide user name and password. The default values are:

Login: *user*

Password: *user*

Entered password doesn't appear on the screen.

For device administrator access, use the command su.



```
$ su
Password: _
```

The default password is: *techbase*

The telnet service is deactivated by default in the Windows 7 OS. It must be turned on manually.

FTP connection

In order to connect to the device via FTP client - device must be connected using direct Ethernet cable or LAN throw router.

How to connect the device via an Ethernet network is described in chapter *Connecting via the Ethernet* .

The settings of the FTP client must provide the following information:

Server Address : **device IP address**

Port: **21**

User: **root**

Password: **techbase**

Optionally, you can set the remote directory to "/" to always start from the root

When accessing configuration files: Some FTP client programs such as TotalCommander, doesn't support copying syscfg configuration file from the location /mnt/mtd/ iMod/config. You must download it from the location /mnt/mtd. This restriction does not apply to the program FreeCommander.

You can download ftp client from location http://www.freecommander.com/pl/index.htm

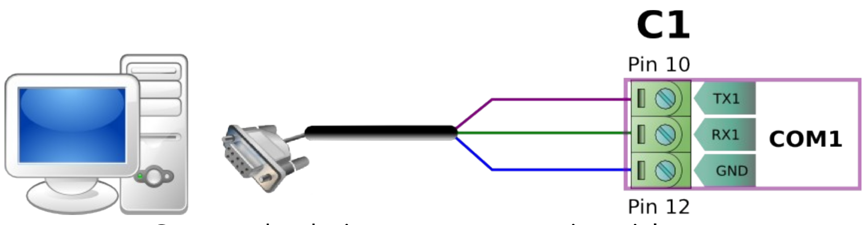
Connecting via the serial port

This chapter describes the step by step how to connect device using serial port.

To connect via serial port, serial terminal service access must be enabled. This service is turned off by default. To switch it on the configuration file SYSCFG must be changed by the line: START_CONSOLE = Y

Connection cable for serial communication RS-232

Figure below shows how to connect the connecting cable, connecting the PC serial interface with a serial interface COM1 furnishing. From arranging the cable is a three veins connected to the respective terminal connectors C1.

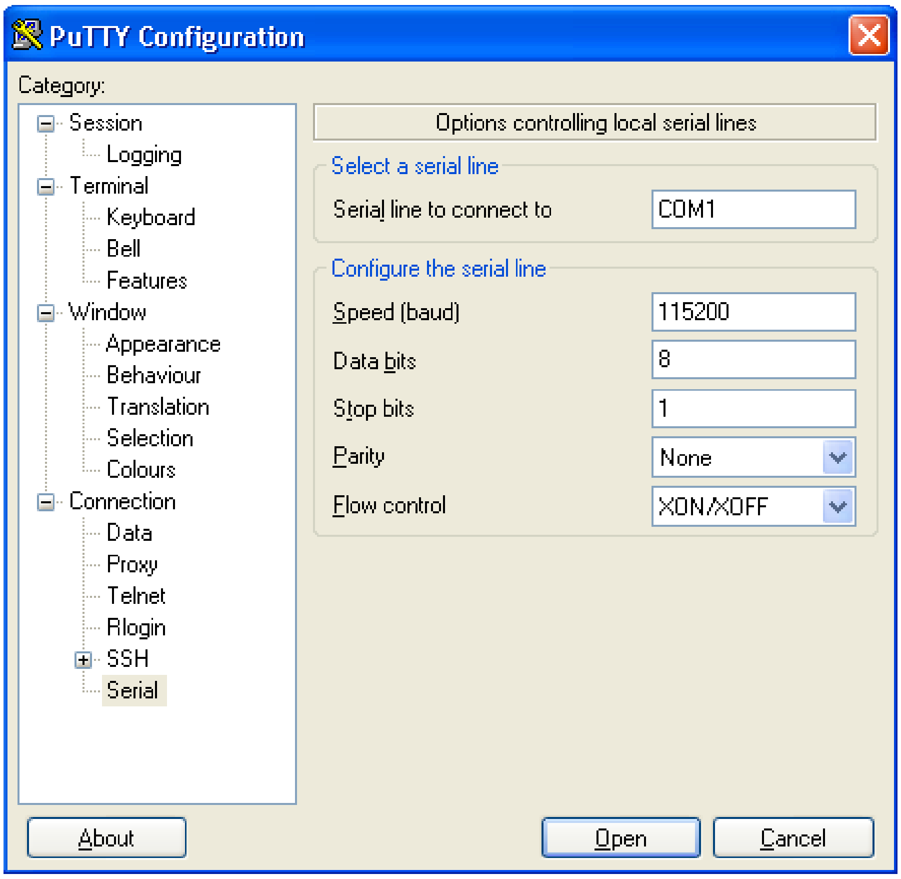


Connect the device to a computer via serial port

Starting terminal application

When serial connection cable is connected to the device, next step is to start terminal application on a PC. This can be any terminal program supports serial ports. Here we describe the connection configuration on the example of the popular PuTTY software.

Putty application can be downloaded from the following location:
http://putty.very.rulez.org/download.html


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Options controlling local serial lines

Select a serial line

Serial line to connect to	COM1

Configure the serial line

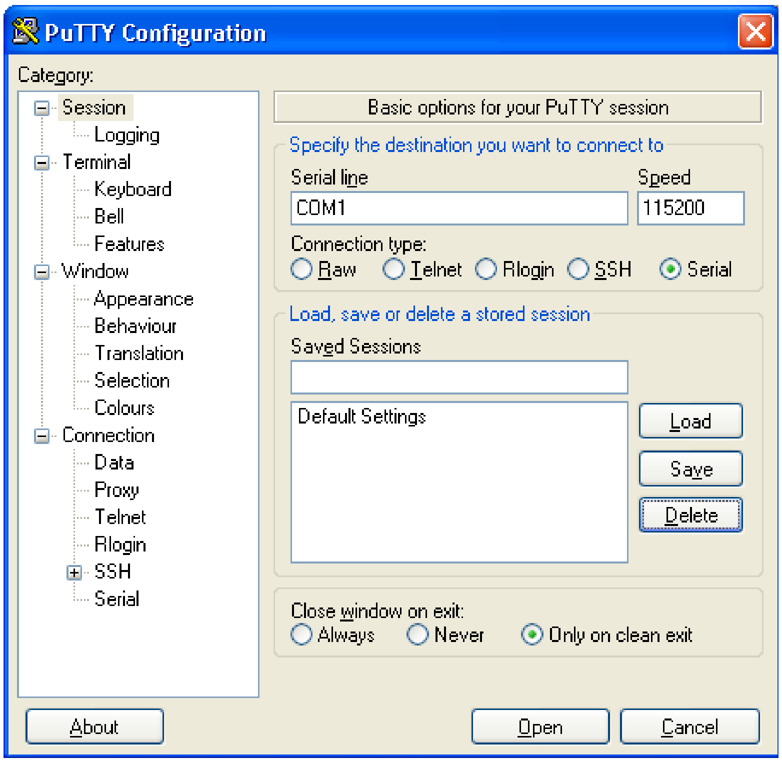
Speed (baud)	115200
Data bits	8
Stop bits	1
Parity	None
Flow control	XON/XOFF

Serial port settings in the application PuTTY

The port number on the computer must be properly set (eg COM1), depending on which computer port connected the device.

Warning
COM number in the terminal program, refers to the computer's serial port and has nothing in common with the identifier (COM1) on the device.

Open button opens another window - a terminal session.



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM1

Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

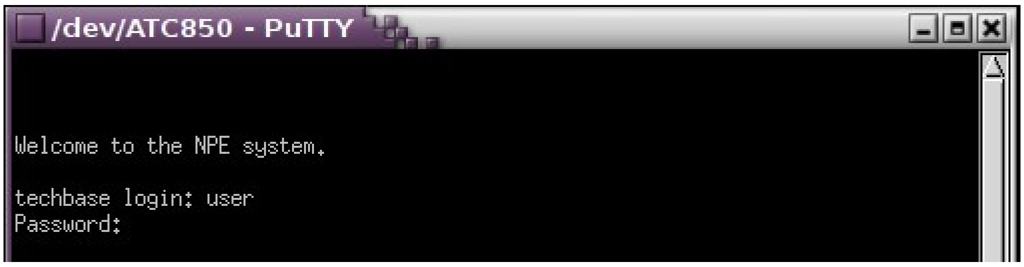
Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Setting up session in PuTTY

After pressing the Open button should see the following window:



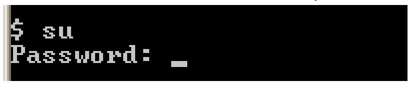
```
Welcome to the NPE system.

techbase login: user
Password:
```

To log in please provide user name and password. The default values are:

Login: *user*

Password: *user*

For an administrator access, use the command *su.*



```
$ su
Password: _
```

Default password: *techbase*

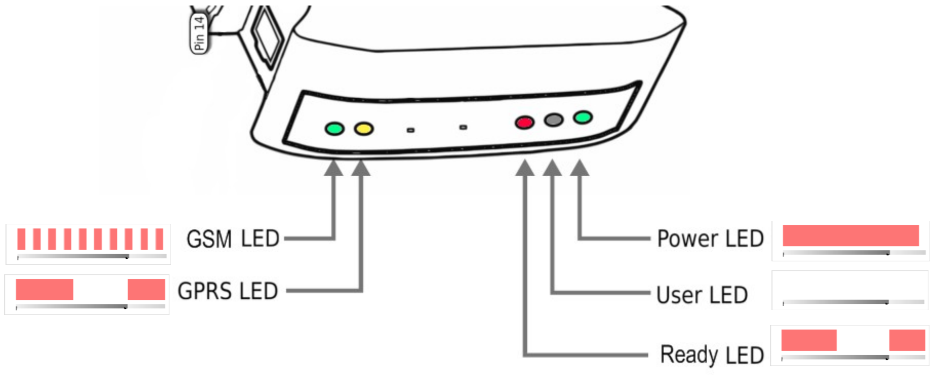
2.4. Checking the operation of the device.

To make sure that the device is working properly, follow the three steps:

1. Verification of LEDs. (Refer to: The status of the unit)
2. Checking the device configuration.
3. Verification of iMod platform.

1. Verification of LEDs

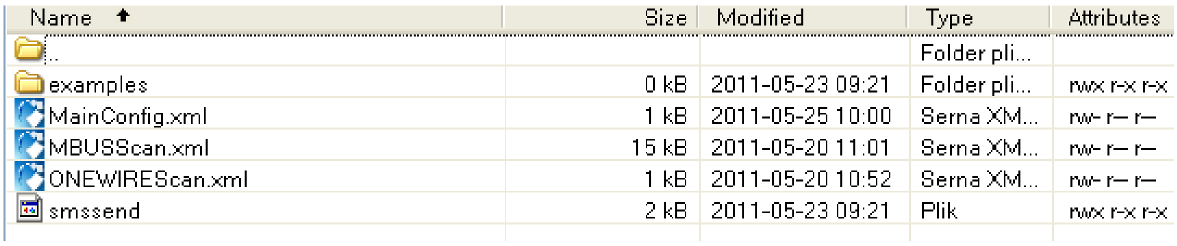
Normal operation of the device is signaling by LEDs as follows:



LED	CORRECT OPERATION
POWER	Continuously lit indicating power on of the device
USER	Off by default. During device start up the LED will shortly blink indicates a valid boot process NPE.
READY	Blinks at a frequency of 1Hz. For more information.:Status LED
GPRS*	Blinks at a frequency of 1Hz. For more information: GPRS Status
GSM*	Quick flashing when modem is power on.

2. Checking configuration files

Please verify the configuration file: *MainConfig.xml*



Name	Size	Modified	Type	Attributes
..			Folder pli...	
examples	0 kB	2011-05-23 09:21	Folder pli...	rwx r-x r-x
MainConfig.xml	1 kB	2011-05-25 10:00	Serna XM...	rw- r-- r--
MBUSScan.xml	15 kB	2011-05-20 11:01	Serna XM...	rw- r-- r--
ONEWIREScan.xml	1 kB	2011-05-20 10:52	Serna XM...	rw- r-- r--
smssend	2 kB	2011-05-23 09:21	Plik	rwx r-x r-x

Semantic correctness of the file should be reviewed using the information contained in the The structure of the application configuration file

The default configuration file is presented in Chapter 3 iMod Tutorial

Chapter 3 Configuration

This chapter describes the basics of NPE configuration.

3.1. How to configure the device

The device based on the iMod platform is configured using two configuration files.

1. syscfg file responsible for the boot options - referred as Start-up Configuration.
2. MainConfig.xml file responsible for iMod platform configuration iMod - referred as Application Configuration

To access configuration files you can use an FTP connection to the device. Detailed information how to setup ftp connection are described in chapter: FTP connection

In order to get new Application Configuration working you need to restart iMod platform. This can be done using the command *imod* command from the console. You can also simple switch off/on the power of the device – that will restart the device and iMod platform.

Configuration files are located in the location: /mnt/mtd/iMod/config

That is the only place where files should be modified.

Changing other files on the system may cause erroneous operation.

3.2. Startup Configuration

File Format

Startup configuration file is a text file format type:

Parameter=value

Forbidden to place a space between the sign '=' and the text. Also, do not use quotation marks eg "Value".

File Location

The configuration file is located at:

Path: /mnt/mtd/

Name: syscfg

For iMod users safter is to change file using syscfg from directory /mnt/mtd/iMod/config

Sample file

```
#
# Custom System configuration file
#

HOST_NAME=techbase

RTC_RESTORE=Y
OUT_RESTORE=Y
DEFAULT_ROUTE=GPRS

GW_IP=192.168.0.1
NET_MASK=255.255.255.0
HOST_IP=192.168.0.101
HOST_MAC=18:83:13:00:30:03

START_CONSOLE=N
START_BLINK=N
START_DHCP=Y
START_FTP=Y
START_TELNET=Y
START_SNMP=Y
START_NPESRV=Y
START_APACHE=Y

GSM_BAUD=230400
GPRS_PIN=
GPRS_APN_NAME=
GPRS_MUX=Y
GPRS_AUTOSTART=Y
GPRS_DNS=AUTO
GPRS_RECONNECT=N
GPRS_PING_IP_1=208.67.222.222
GPRS_PING_IP_2=208.67.220.220

GPRS_AUTO_DNS=Y
GPRS_DNS_1=211.138.151.161
GPRS_DNS_2=202.101.103.55
GPRS_DNS_3=8.8.8.8
GPRS_LOGIN=ppp
GPRS_PASSWORD=ppp
GPRS_DYNDNS=N
```

Description of parameters

Parameters defining the start of

Parameter Name	Description	Value format	Default value
START_CONSOLE	Launch console on serial port COM1 (/dev/ttyS0)	Y or N	N
START_DHCP	Run DHCP client (dynamic IP)	Y or N	Y
START_FTP	Run FTP server	Y or N	Y
START_TELNET	Run telnet	Y or N	Y
START_SNMP	Starting the SNMP service This option is only available for versions of SNMP	Y or N	N
START_NPESRV	Service for finding the device in a network.	Y or N	Y

Configuring the Ethernet interface

Parameter Name	Description	value format	Default value
HOST_NAME	Host name	String	techbase
GW_IP	Gateway IP address	XXX.XXX.XXX.XXX	192.168.0.1
NET_MASK	Subnet Mask	XXX.XXX.XXX.XXX	255.255.0.0
HOST_IP	IP address	XXX.XXX.XXX.XXX	192.168.0.101
HOST_MAC	Device MAC address	XX.XX.XX.XX.XX.XX	Visible on the housing
START_APACHE	Autostart Apache2 server	Y/N	Y

GSM Modem Configuration*

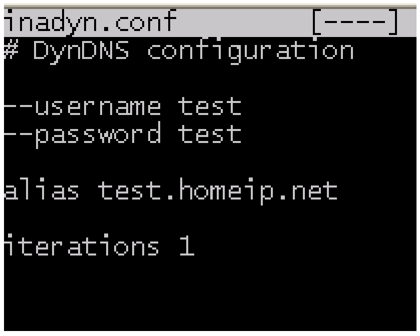
Parameter Name	Description	value format	Default value
GPRS_AUTOSTART	Auto Connect on start-up GPRS modem	Y lub N	Y
GPRS_RECONNECT	Triggering refresher GPRS connection after its discontinuation	Y lub N	N
GPRS_APN	APN for GPRS	String	blank
GPRS_APN_LOGIN	Login to point GPRS APN	String	blank
GPRS_APN_PASSWORD	Password to point GPRS APN	String	blank
GPRS_ PIN	The PIN for the card. In the absence of a PIN on the card this setting will be ignored.	4 cyfry	blank
GPRS_DNS	Determination of the DNS server for GPRS	AUTO - automatically download to your DNS server XXX.XXX.XXX.XXX-explicit typing a specific IP address	AUTO
The parameters under normal operating conditions do not require modifications			
DEFAULT_ROUTE	Setting the default route - Gateway - for outgoing calls outside Their own subnet APN and Ethernet - a typical Internet connections.	GPRS - External calls handled by a built-in GSM / GPRS modem ETHERNET-external calls supported by the Ethernet interface	GPRS
GPRS_PING_IP_1	IP address used for checking the GPRS related GPRS_RECONNECT.	XXX.XXX.XXX.XXX	208.67.222.222 *(adres serwera OpenDNS)*
GPRS_PING_IP_2	IP address used for checking the GPRS related GPRS_RECONNECT.	XXX.XXX.XXX.XXX	208.67.220.220 *(zapasowy adres serwera OpenDNS)*
MUX	GSM modem multiplexing mode Allows simultaneous use of GPRS session and sending SMS	Y lub N	Y
GSM_BAUD	The speed of communication with the GSM modem.	Permitted speed in bits: 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400	230400

If device has a built-in GPRS modem GRPS_AUTOSTART is enabled by default. It is also recommended to enable GPRS_RECONNECT option but it must corresponding with correct GPRS_PING_IP_2 GPRS_PING_IP_1 parameters setup.

DynDNS Configuration

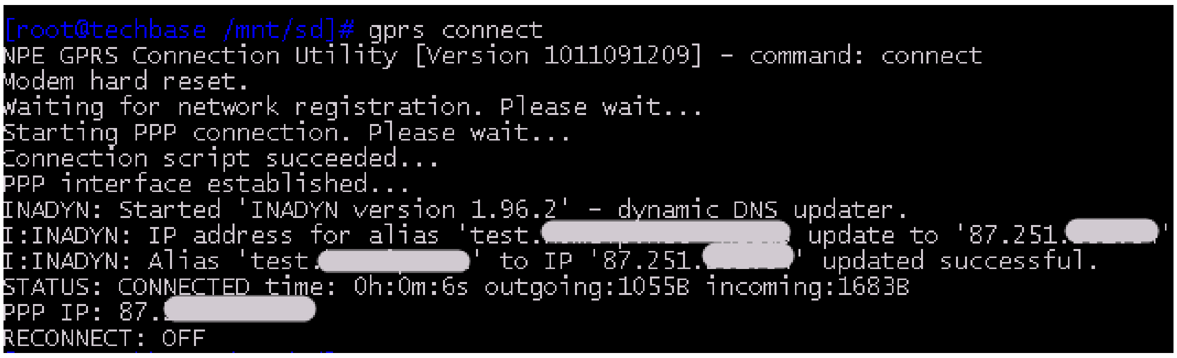
The iMod contains a mechanism for using the DynDNS service for the PPP protocol (GPRS connections). This script uses the InaDynConf library. More information about this pack can be found at: http://www.inatech.eu/inadyn/

DynDNS service configuration is limited to creating an account at the www.DynDNS.com service and editing the file /mnt/mtd/iMod/config/inadyn.conf.
An example of file configuration is shown below:



```
inadyn.conf          [----]
# DynDNS configuration

--username test
--password test

alias test.homeip.net

iterations 1
```

If GPRS_DYNDNS is configured to Y(es), the correct start-up of the PPP protocol should appear as follows:



```
[root@techbase /mnt/sd]# gprs connect
NPE GPRS Connection Utility [Version 1011091209] - command: connect
Modem hard reset.
Waiting for network registration. Please wait...
Starting PPP connection. Please wait...
Connection script succeeded...
PPP interface established...
INADYN: Started 'INADYN version 1.96.2' - dynamic DNS updater.
I:INADYN: IP address for alias 'test.[redacted] update to '87.251.[redacted]'
I:INADYN: Alias 'test.[redacted]' to IP '87.251.[redacted]' updated successful.
STATUS: CONNECTED time: 0h:0m:6s outgoing:1055B incoming:1683B
PPP IP: 87.[redacted]
RECONNECT: OFF
```

Chapter 4 iMod Tutorial

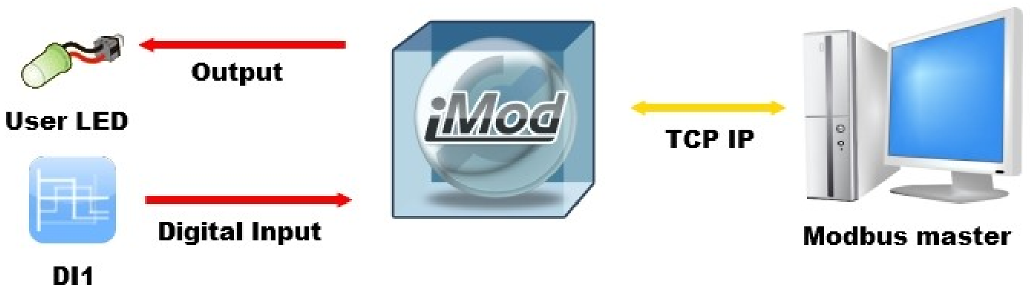
This chapter describes the example configurations for the iMod platform.

4.1. Example 1: Access to hardware resources

This example presents how to access the device hardware resources with applications such as MODBUS master (eg SCADA) systems on the network, TCP IP.

Resources that are defined to control are::

- Digital Output - for example, LEDs embedded in the device
- Digital input - for example, the entry marked DI1



Access to resources such input / output through the digital channel MODBUS TCP

The test requires a properly configured your settings iMod: MainConfig.xml.

Before configuring, make sure that the device is visible through an Ethernet interface. How to connect devices through Ethernet is described in the chapter Connecting via the Ethernet port.

Configuration of various elements is contained in file: **/mnt/mtd/iMod/**MainConfig.xml. A basic description of the various parts of this file are in chapter: Configuration.

Configuration described in this example is default configuration loaded onto the device. This configuration is also available on the CD as an example of a configuration iMod or as file: /mnt/mtd/iMod/config/exmaples/example1-hardware_access.xml.

This example shows how:

- get access to hardware resources (source channel)
- forward hardware resources to MODBUS TCP Slave (access channel)
- parameter to access the USER LED diodes
- parameter to access digital input – DI1

Definition of source channel (hardware resources)

To get access to hardware resources included in NPE system is needed the definition of this channel in the following way.

```
<source-channel name="npe_io">
      <protocol name="HARDWARE"/>
      <gap>0</gap>
      <cycle>2</cycle>
</source-channel>
```

More about gap and cycle parameters are described in section source-channel structure

Channel Definition MODBUS Slave

To enable connection from the client application MODBUS Master device to iMod, it has to be define a channel of communication a MODBUS Slave. The following section defines the configuration of such a channel on TCP port 502:

```
<access-channel name="modbus_s1">
       <protocol name="MODBUS"/>
       <port>"ET-502-TCP"</port>
       <property name="device-id" value="1" />
</access-channel>
```

LED Control

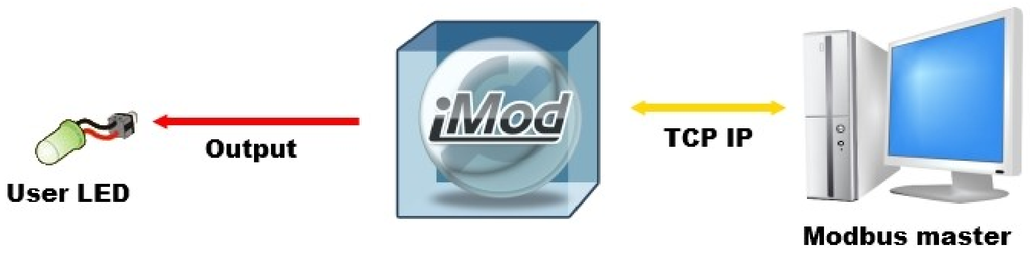


Illustration of the structure described in Example 1

To access the User LED diodes must declare a parameter that specifies:

- A unique identifier for the platform iMod parameter
- source channel for the parameter - in this case is a channel of 'hardware resources' , resource identifier 'npe_io' with value 'USER_LED'
- channel access to a parameter defined earlier as modbus_s1 channel assignment with the MODBUS address for this parameter (100). Thus make possible to control this parameter via modbus client application (eg. SCADA).

List of available hardware resource identifier is placed in *source-channel-type*

```
<parameter>
        <id>100</id>
        <source-channel channel-name="npe_io" parameter-id="USER_LED" />
         <access-channel channel-name="modbus_s1" parameter-id="100"/>
</parameter>
```

Read digital input

In a similar way as for LED now define the access to the digital input.

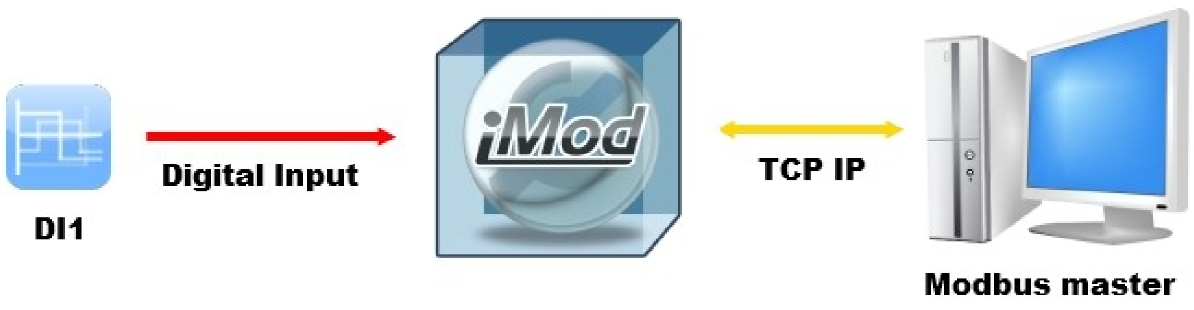
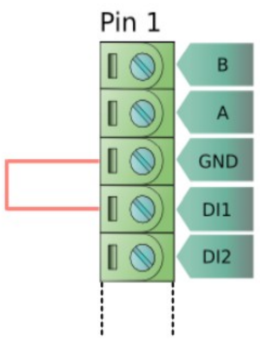


Illustration of the structure described in Example 2

In this case, the structure <parameter> defines:

- unique identifier for digital input parameter (<id></id>)
- source channel for parameter - npe_io identifier for this a parameter is DI1
- channel access with the same parameter as for LEDs, but here we assign modbus address 101

```
<parameter>
        <id>101</id>
        <source-channel channel-name="npe_io" parameter-id="DI1"/>
        <access-channel channel-name="modbus_s1" parameter-id="101"/>
</parameter>
```



Connection entry in order to read 0

Disconnected entry should indicate the status 1. After connect the digital input DI1 by cable to the connector labeled GND status will change to 0.

Verification of example

To verify the above example, will be nedded:

- modbus master application (eg. Modbus pool)
- active network connection with device (information how to get the connection can be found in the chapter *Connecting via the Ethernet*)
- MainConfig.xml file configured in accordance with Example Access to hardware resources

In this example we use a free demo tool named modbus master Modbus poll available at:

http://www.modbustools.com/download/ModbusPollSetup.exe

Check the current configuration of iMod

1. Connect using telnet.
2. Display contents of the configuration file using:

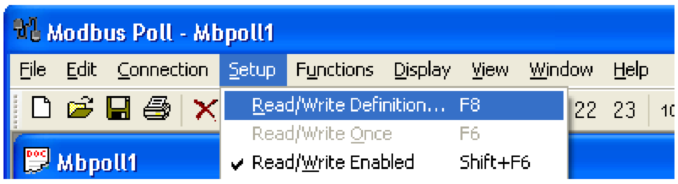
```
$ cat /mnt/mtd/iMod/config/MainConfig.xml
```

This command displays the contents of the file. If you do not include the above example, copy it from directory /mnt/mtd/iMod/config/examples

Attention!
Saved new configuration file will be implement after reboot by command `$ imod start`

Setup a connection via Modbus Pool

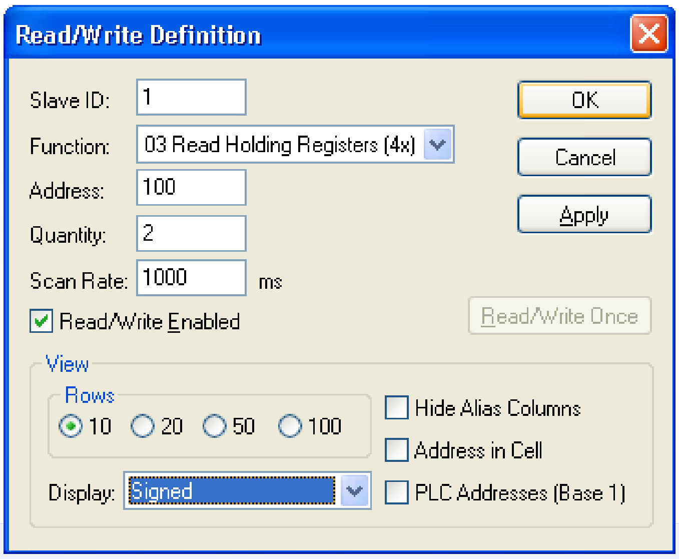
PROPERTIES SET MODBUS CONNECTIONS


Modbus Poll - Mbpoll1
File Edit Connection Setup Functions Display View Window Help
Read/Write Definition... F8
Read/Write Once F6
Read/Write Enabled Shift+F6
Mbpoll1
22 23

1.Click:

Setup -> Read / Write Definition

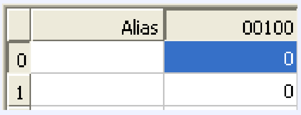
2. Set the following parameters:



FUNCTION:	03 Read[...]
ADDRESS:	100
QUANTITY:	2
SCAN RATE:	1000 ms
Read/Write Enabled	<checked>

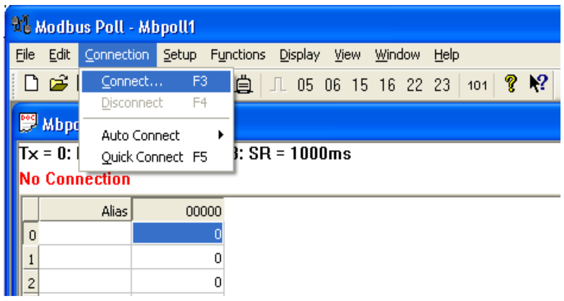
3. Then confirm: OK

Value of second column change to 00100



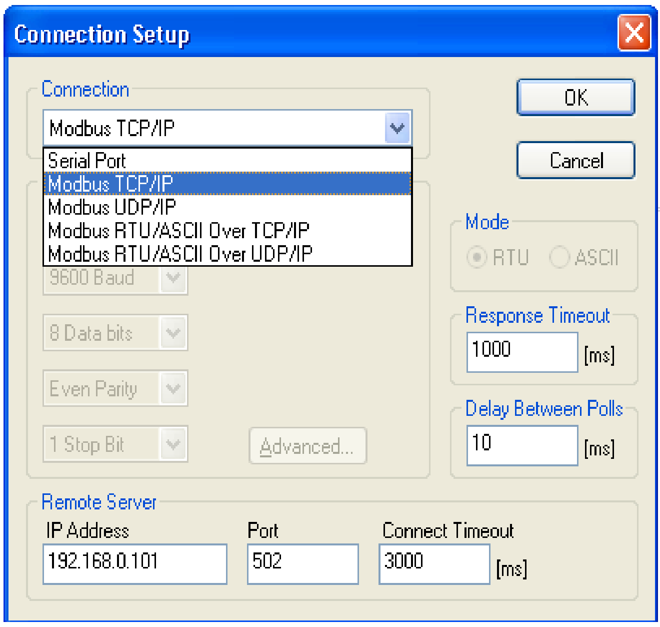
	Alias	00100
0		0
1		0

START MODBUS TCP CONNECTION



1. CONNECTION -> CONNECT (F3)

2. Skip the Registration screen by pressing:
REGISTER LATER



3. It should be set:

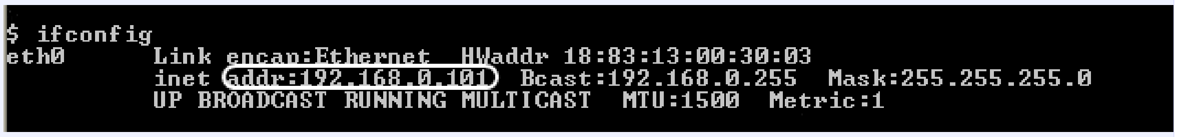
PARAMETR	VALUE
CONNECTION:	Modbus TCP/IP
MODE:	RTU
IP ADDRESS*:	192.168.0.101
PORT*:	502

Confirm the change by: OK

Modbus port was defined in access channel - MODBUS Slave:

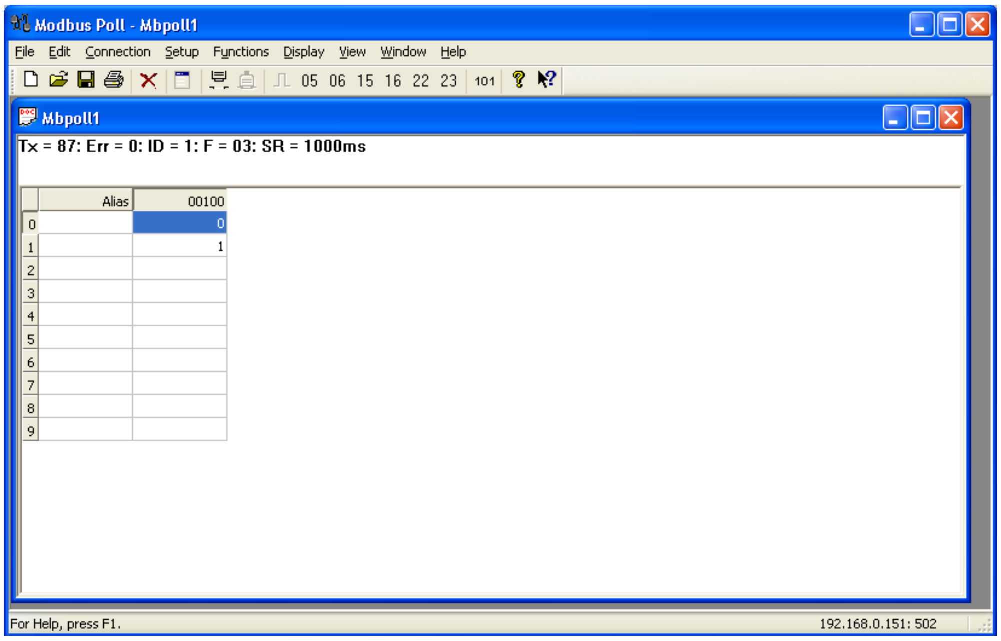
```
[...]
<port>"ET-502-TCP"</port>
[...]
```

You can get device IP address using ifconfig command on the iMod:



```
$ ifconfig
eth0       Link encap:Ethernet  HWaddr 18:83:13:00:30:03
           inet addr:192.168.0.101  Bcast:192.168.0.255  Mask:255.255.255.0
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

If you have done all the steps correctly, you should get the following screen.



Modbus Poll - Mbpoll1
File Edit Connection Setup Functions Display View Window Help
05 06 15 16 22 23 101
Mbpoll1

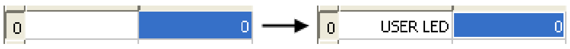
Tx = 87: Err = 0: ID = 1: F = 03: SR = 1000ms

	Alias	00100
0		0
1		1
2		
3		
4		
5		
6		
7		
8		
9		

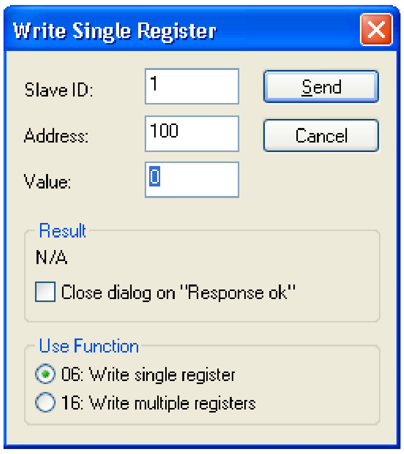
For Help, press F1. 192.168.0.151: 502

USER LED Control

In the Alias column, by double-clicking you can give label a working parameter.



The first row is the status of USER LED. You can editi cells by double clicking the left mouse button on the cell with value 0.



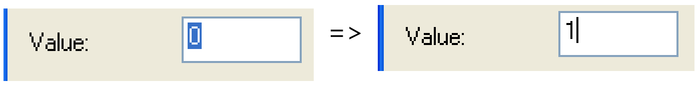
Write Single Register

Slave ID:	1	Send
Address:	100	Cancel
Value:	0	

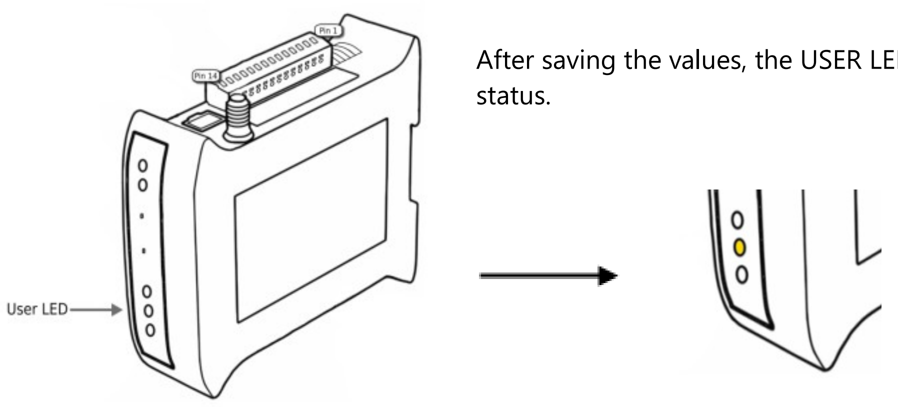
Result
N/A
☐ Close dialog on "Response ok"

Use Function
● 06: Write single register
○ 16: Write multiple registers

This will open a new window. There change the value.



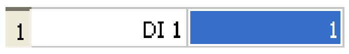
Press:

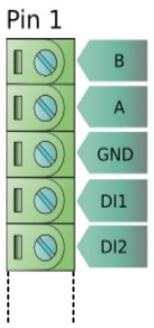


After saving the values, the USER LED change status.

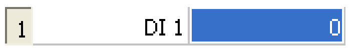
Check the status of digital input DI1

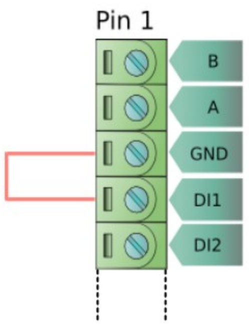
1. Initial state of DI is 1





2. After a short-circuit connectors DI1 to the ground (any wire), its status will change to 0





4.2. Example 2: Communication with modbus device

This example demonstrates how to communicate iMod with modbus RTU device.

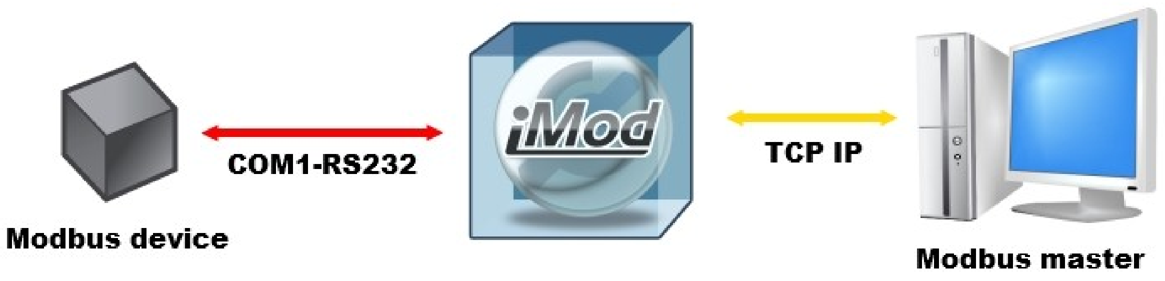


Illustration of the structure described in Example 2

Below configuration supports Modbus RTU protocol on port COM1. Source code is as follows:

```
<source-channel name="Modbus_M1">
        <protocol name="MODBUS"/>
        <port>"com0-19200-8E1"</port>
        <gap>0</gap>
        <cycle>20</cycle>
</source-channel>
```

Serial port parameters are as follows:

- COM port: 0
- baudrate: 19200 b / s
- datatbits: 8
- parity: Even
- Stop bits: 1
- type protocol: Modbus RTU

More information how the channel is defined could be fount in Section *Definitions of communication channels*

Another part of the file is <parameter> structure associated with the above-defined communication channel:

```
<parameter>
        <id>11201</id>
        <source-channel channel-name="Modbus_M1" parameter-id="11201"/>
        <access-channel channel-name="modbus_s1" parameter-id="11201"/>
</parameter>
```

Modbus RTU register the address 11201, the device connected to COM1 port is provided for the Register of converts Modbus TCP with the same address.
This example enables caching of parameter values and control an external device that supports Modbus.

Channel modbus s1 is defined similarly as in Example 1: Access to hardware resources

4.3. Example 3: Defining the channel and event notification

E-mail Notification

Based on configuration in this example, you can send e-mail when digital input is set to logical 0.

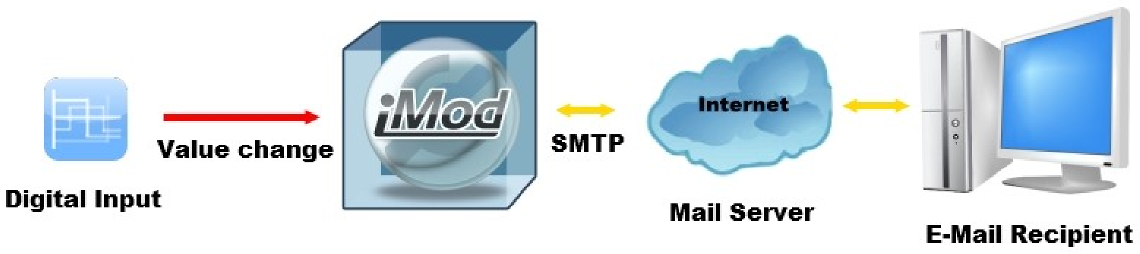


Illustration of the structure described in Example 3

In this example is defined a single parameter - the digital input DI1. iMod allows you to assign alarms (<event>) for each parameter.

<event> structure used message channel and message id to define recipients and contains.

Sample definition of a channel of information (message-channel) and the error message is as below:

```
<message-channel name="Email_sender">
        <protocol name="EMAIL">
               <property name="user" value="testnpe"/>
               <property name="password" value="123npe"/>
        </protocol>
        <port>"poczta.o2.pl"</port>
        <recipient>"techbase@a2s.pl"</recipient>
</message-channel>
```

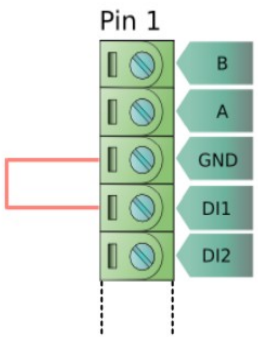
```
<message id="Mess_1">
              <![CDATA[
              "Connected to ground"
              ]]>
</message>
```

Below is defined a parameter with alert (LoAlarm) for DI2.

<event> Describes the structure in this case the event has resulted in sending a message with id "Mess_1" by channel "Email_sender" if the value is lower than the first

```
 <parameter>
        <id>102</id>
        <source-channel channel-name="npe_io" parameter-id="DI1"/>
        <access-channel channel-name="Modbus_S1" parameter-id="102" />
       <event type="OnChange">
               <message-channel channel-name="Email_sender"/>
               <message-id>"Mess_1"</message-id>
               <hysteresis>0.0</hysteresis>
        </event>
 </parameter>
```

In the case of digital input, this means that mail will be sent when the logic state changes to 0 (input connected to ground).



Force logic state 0 on the digital input DI1.

SMS Notification

Analogous to the Email Channel, could be defined the SMS channel.

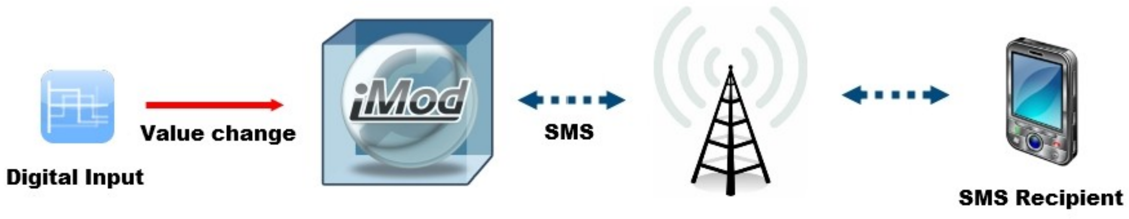


Illustration of use of SMS notification channel

Messege channel for SMS notification:

```
   <message-channel channel-name="SMS_sender">
          <protocol name="SMS"/>
        <recipient>"48123456789"</recipient>
   </message-channel>
```

Here the definition of an event notification via SMS connected with DI3:

```
 <parameter>
        <id>103</id>
        <source-channel channel-name="npe_io"
                        parameter-id="DI2"/>

        <access-channel channel-name="modbus_s1"
                        parameter-id="103" />

        <event type="LoAlarm">
              <message-channel channel-name="SMS_sender"/>
              <message-id>"Mess_1"</message-id>
              <hysteresis>0.0</hysteresis>
              <property name="trigger" value="1"/>
        </event>

</parameter>
```

SMS notification will be sent in the case when the digital input 3 change the logical value of 1 to 0

Apart from alarms of excessive or insufficient values, there is also an alarm option based on changes of a parameter by a given hysteresis or lack of such a change.

The GPRS connection must be correctly configured for the text messaging system to function correctly. More information regarding GPRS configuration can be found in the document NPE Instructions

4.4. Przykład 4: Modbus messages

The available configuration examples can be found at the following device directory:

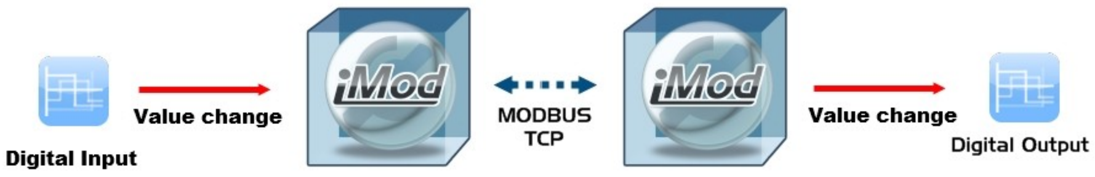
/mnt/mtd/iMod/config/examples/example4-modbus-events.xml

Message modbus type

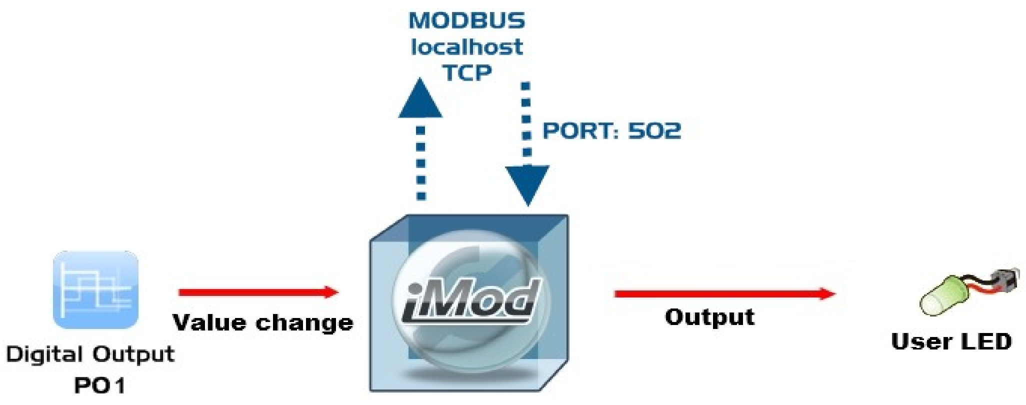
The iMod engine services a unique mechanism of event-driven modbus messages. This makes it possible to decrease transmission between two iMod devices to a bare minimum. It is also possible to force-write or force-read selected parameters based on:

- exceeding of a set value
- decrease of a parameter below a set value
- change in the parameter value by a given hysteresis

It is also possible to send a read value or write it to another modbus device, E.g.



The example shown above illustrates channel operation based on an 'onChange' event. A change on digital output PO1 will also cause a change on USER_LED.



The modbus channel is declared as

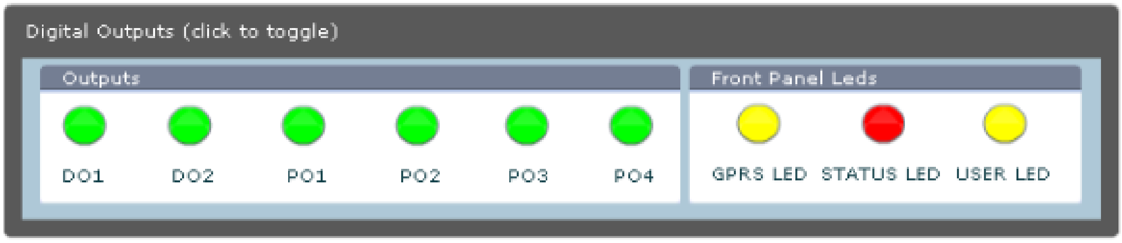
```
        <message-channel name="Modbus">
                     <protocol name="modbus"/>
                     <port>"ET-localhost-502-TCP"</port>
        </message-channel>
```

Upon a change of parameter 104, its state will be propagated to address 105, under which USER_LED is present.

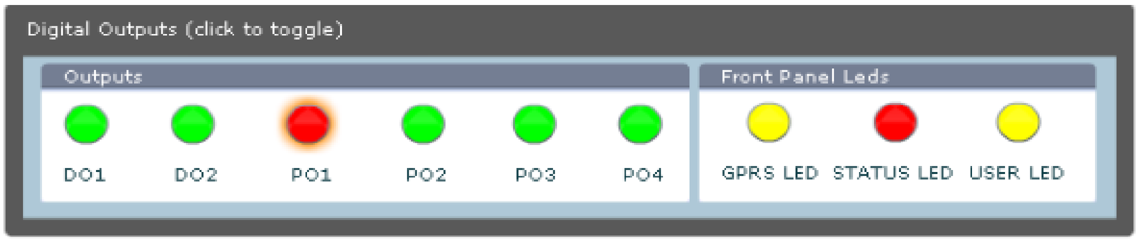
```
        <parameter>
                <id>"104"</id>
                <description>"PO1"</description>
                <source-channel channel-name="NPE_io" parameter-id="PO1"/>
                <access-channel channel-name="Modbus_S1" parameter-id="104"/>
        <event type="OnChange">
                <message-channel cannel-name="Modbus" parameter-id="105"/>
        </event>
        </parameter>
```

These are the three steps:

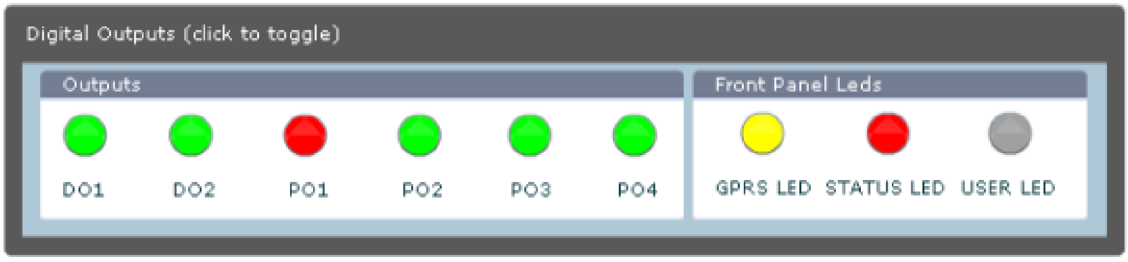
I. All PO1 inputs and USER LED are active.



II. PO1 has been changed. The value from PO1 has been sent to modbus address 105 of port 502 of the device with an ip of: Localhost.

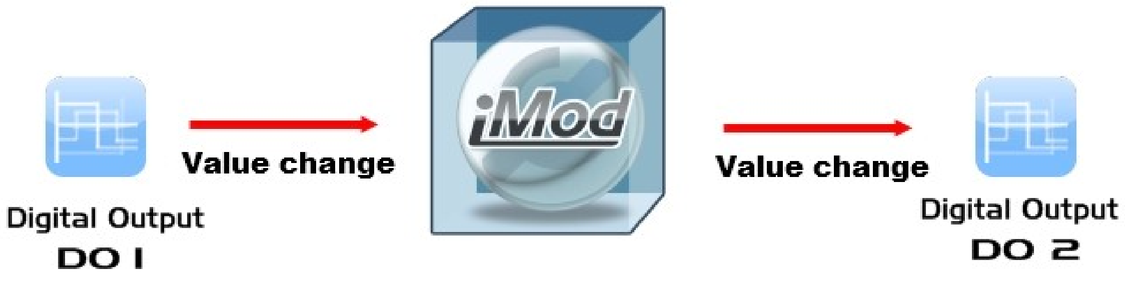


III. The iMod has read the write report to modbus address 105 (USER_LED) and carries out a change of the parameter.



Message force-write

A Force-Write type message is a modbus message which makes it possible to write. The example describes a method, in which a change in the value of DO1 will result in the change of DO2.



Declaration of the channel is carried out by granting it a unique name. 'forcewrite' should be given as the protocol. It is also possible to define several recipients. The ID parameter of the iMod to which the parameter is to be written is given as the recipient. The value of the parameter on which the event is "based on" is sent by default. However, it is possible to overwrite this by declaring the value to be written in the message. This case is not included among the examples.

```
        <message-channel name="ForceWrite">
                     <protocol name="forcewrite"/>
                     <recipient>"103"</recipient>
        </message-channel>
```

In the parameter definition shown below, an element not used up till now has been shown: <init-value>. It enables setting starting values after start-up of the iMod platform.

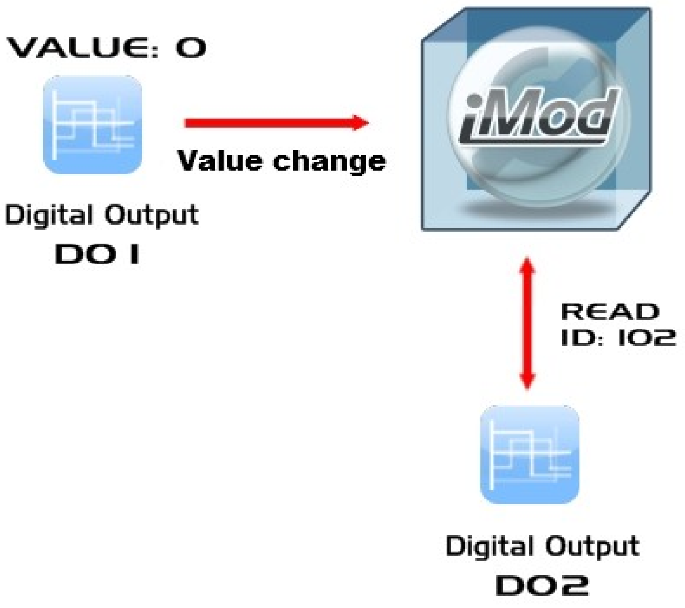
Every time this parameter is changed, it will write this value to the parameter. The recipient of the parameter has been defined during channel definition, which is why the recipient address does not need to be defined at this point.

```
<parameter>
             <id>"102"</id>
             <comment>"DO1"</comment>
             <init-value>"0"</init-value>
             <description>"DO1"</description>
             <source-channel channel-name="NPE_io" parameter-id="DO1"/>
             <access-channel channel-name="Modbus_S1" parameter-id="102"/>
             <event type="OnChange">
                    <message-channel channel-name="ForceWrite"/>
             </event>
</parameter>
```

Message force-read

The modbus message for reading values is used the least often, however, in certain applications, it is irreplacable.

This type of message makes it possible to read other parameters defined in the iMod platform.



Channel declaration takes place analogously to the other messages. It is possible to define the recipient during channel definition as well as during execution of the event in the parameter.

In this case, declaration occurs for the parameter, which is why the definition of the channel itself is only three lines long.

```
	<message-channel name="ForceRead">
		<protocol name="forceread"/>
	</message-channel>
```

Setting a 'NoChange' event type causes a forced refresh of the value of parameter 102 in the internal iMod parameter table for every reading for which USER_LED does not change its value.

```
	<parameter>
		<id>"105"</id>
		<comment>"USR LED"</comment>
		<description>"USR LED"</description>
		<source-channel channel-name="NPE_io" parameter-id="USER_LED"/>
		<access-channel channel-name="Modbus_S1" parameter-id="105"/>
		<event type="NoChange">
		<message-channel channel-name="ForceRead" parameter-id="102"/>
		</event>
	</parameter>
```

4.5. Przykład 5: Script event

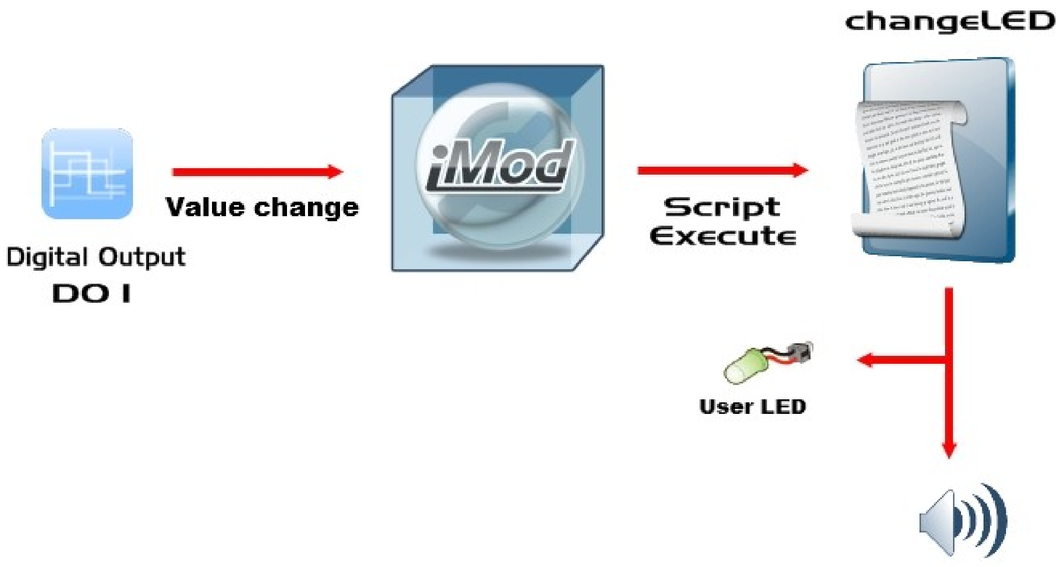
The available example of configuration can be found under the following device directory:

/mnt/mtd/iMod/config/examples/example4-modbus-events.xml

Event script

Using the iMod, the user has full access to LINUX system services. This also makes it possible to use ash scripts. More information regarding writing scripts can be found in generally available materials regarding script structure.

The platform supports script use through a special kind of message. Connected with the modmas library available for NPE, this gives a veritable sea of possibilities. For



In order to make a channel, a script message channel should be defined.

Besides the protocol, the <port> element where the file directory is input should be filled in. Then, the recipient is defined as the file name that is to be executed.

```
      <message-channel name="Script">
                   <protocol name="script" />
                   <port>"/mnt/mtd/iMod/config/examples"</port>
                   <recipient>"changeLED"</recipient>
       </message-channel>
```

The parameter appears as a typical event declaration with message channel start-up would.

```
  <parameter>
       <id>"110"</id>
       <source-channel channel-name="NPE_io" parameter-id="DO1"/>
       <access-channel channel-name="Modbus_S1" parameter-id="110"/>
             <event type="OnChange">
                   <message-channel channel-name="Script"/>
             </event>
  </parameter>
```

However, for the script to be executed correctly, the appropriate authorization for file execution must be added:
chmod 754 /mnt/mtd/iMod/config/examples/changeLED

The changeLED file appears as follows:

```
#!/bin/sh
npe +USER_LED
npe +PB10

sleep 1

npe -USER_LED
npe -PB10
```

Execution of the script will result in a sound signal and activation of the USER_LED for one second.

To execute it automatically, the value of output DO1 must be changed.

4.6. iMod platform configuration

This chapter describes the structure of the iMod platform configuration file, which completely defines the principles of the application's operation. This file is in the format of an XML standard text file.

It can be edited using any text editor as well as using dedicated tools aiding in the editing of these types of files.

The file structure has been designed so as to reflect the functioning of the application in the simplest way possible. A general description of individual blocks of this file is given below. The appendix *The structure of the application configuration file* contains a detailed description of all allowed elements of this file.

File directory

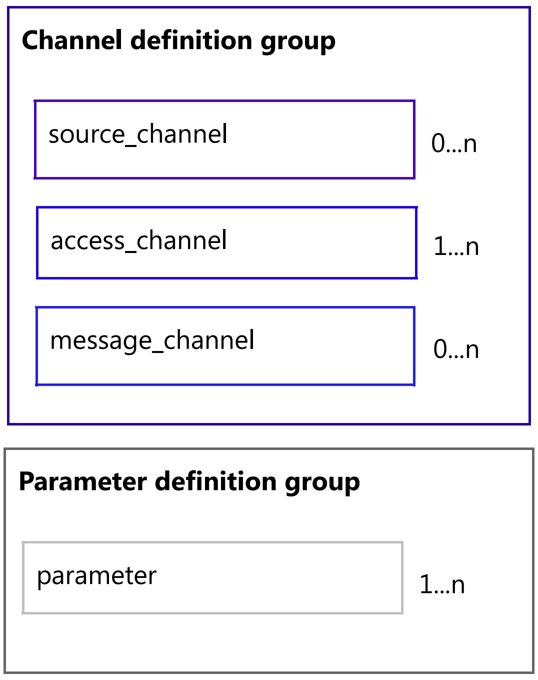
Ścieżka: /mnt/mtd/iMod/config/

Nazwa: MainConfig.xml

Configuration file block structure

The XML configuration file is comprised of blocks defining individual elements of the iMod system. Two main configuration blocks can be discerned:

- communication channel definition block
- parameter definition block.



Chapter 5 Introduction to XML

This chapter gives basic information regarding XML configuration.

When making your own configuration file, the following steps should be followed.
Define the channel group:

- from where will data be taken
- methods of access to data
- methods of informing of events

Define parameter groups:
After defining channels, parameters should be defined

- parameter address=
- parameter data source
- method of parameter access
- parameter type (real32, int16, int32)
- assigning of a specific event (OnChange, NoChange, LowAlarm, HiAlarm)
- assigning a specific message to the event (E-mail, text message, Script, SQL, Modbus)

5.1. Communication channel definitions

Source channels

Element name: **<source-channel>**

Definition of the channel being the input data source for the iMod platform.

Example of a source channel definition:

```
<source-channel name="npe_io">
      <protocol name="HARDWARE"/>
      <gap>0</gap>
      <cycle>2</cycle>
       <delay>300ms</delay>
</source-channel>
```

The above example shows a configuration of the information source as device hardware resources (HARDWARE).

The following protocols are also available:

- MODBUS
- MBUS
- HARDWARE
- 1-WIRE
- SNMP

GAP & CYCLE & Delay

The <cycle> parameter defines the frequency of updating modbus address series. The <gap> parameter specifies the length of pauses between subsequent reading cycles.
The<delay> delay parameter defines the period between polling of subsequent modbus addresses in the cycle. The whole values are given in seconds. The delay parameter is optional, and in its absence, a default value of 0ms is set.
When making the configuration file, special attention should be paid to the appropriate selection of values for these parameters. System operation should be monitored after 30 minutes from the start of operation according to the target configuration using the *uptime* command. The second and third value of the uptime command should not exceed a value of 1.0.

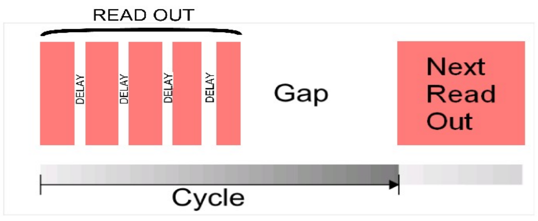


Figure illustrating the relationship between Cycle and Gap

In special cases, the cycle time can be lengthened by an unspecified value caused by a lengthened reading time. In such a case, gap remains unchanged.

PORT

If a modbus slave device is specified as the data source, the <port> element also appears.
An example of a data source declaration from a TCP modbus is shown below.

```
<source-channel name="Modbus_slave">
        <protocol name="MODBUS" />
        <port>”ET-192.168.0.101-502-TCP”</port>
        <gap>0</gap>
        <cycle>2</cycle>
</source-channel>
```

MULTIADDRESSING

The main slave modbus ID can be assigned during channel definition. It will be default for all parameters.

```
<source-channel name="Modbus_slave">
        <protocol name="MODBUS" />
       <port>C4-115200-8N1</port>
       <gap>0</gap>
        <cycle>2</cycle>
        <property name=”device-id” value=”5” />
</source-channel>
```

The ID can also be defined (overwritten) for the parameter:

```
<parameter>
      <id>103</id>
      <description>"Device 2 00"</description>
      <source-channel channel-name="MODBUS_M1”>
       <property name=”device-id” value=”7” />
       </source-channel
</parameter>
```

Data access channels

Element name: **<access-channel>**

Definition of the channel enabling data access through the iMod platform.

An example of an access channel definition is shown below:

```
<access-channel name="modbus_s1">
        <protocol name="MODBUS"/>
        <port>"ET-502-TCP"</port>
         <property name=”device-id” value=”1”/>
</access-channel>
```

The above example demonstrates an access channel configuration using a MODBUS protocol. The <port> parameter defines, that it is to take place by means of Ethernet TCP using port 502.

Message channels

Element name: <message-channel>

The definition of the channel for messages sent by the iMod platform in response to events defined by the user. A channel of this type can also be called a notification or alarm channel.

An example of a message channel definition is shown below:

```
<message-channel name="Email_sender">
        <protocol name="EMAIL">
               <property name="user" value="testnpe"/>
               <property name="password" value="123npe"/>
        </protocol>
        <port>"poczta.o2.pl"</port>
        <recipient>"techbase@a2s.pl"</recipient>
</message-channel>
```

The above example shows a message channel configuration using e-mail.

- The <property name> parameter sets the user name and email account password used for sending messages.
- The <port> parameter specified the SMTP server name.

• The <recipient> parameter specifies the recipient of the message, in this case, the e-mail address. There can be more than one recipient.

The message channel can also implement text messages. The example below shows a channel called "sms sender." This channel is responsible for sending text messages to the telephone number „123465789"

```
<message-channel channel-name="SMS_sender">
        <protocol name="SMS"/>
        <recipient>"48123456789"</recipient>
</message-channel>
```

Parameter definitions

Element name: <parameter>

The <parameter> structure defines a single iMod parameter. It contains relations with channels: the source channel and access channel can be defined.

The parameter declaration can also contain definitions of events related to it (e.g. alarms). Events are related to message channels through which they are sent. This will be shown in the section *Błąd: Nie znaleziono źródła odwołania*.

Below, a definition for a parameter related to the hardware resources channel through digital input No. 1 is given. This parameter is available using the MODBUS protocol in reading mode under address 100.

```
<parameter>
        <id>100</id>
        <description>"Digital input 1"</description>

        <source-channel channel-name="npe_io"
                        parameter-id="DI1"/>

        <access-channel channel-name="modbus_s1"
                        parameter-id="100"/>
</parameter>
```

The parameter has an internal unique identifier 100 and is describes as *Digital input 1*.
Methods for ascribing communication channels for oarameter definitions are described below in further detail.

Assigning channels to parameters

The <parameter> structure should contain elements defining the relationship of the parameter with:

- the source channel, or the relationship with resources
- the access channel
- optionally, the parameter defining the event

Assigning the source channel

Element name: <source-channel>

The parameter data source is defined using the <source channel> element.

In the example, the data source is defined as read-only digital input DI1.

Possible values for parameter-id for the **Hardware** channel are as follows:

Digital Input	Digital Output	Analog Input	Other
DI1	DO1	AI1	USER_LED
DI2	DO2	AI2	YEAR
DI3	PO1	AI3	MONTH
DI4	PO2	AIV	DAY
DI5	PO3		HOUR
DI6	PO4		MINUTE
DI7			
DIW			

For other channel types, other values are possible. A more specific list of all parameter ids for the source channel can be found in the chapter:
source-channel structure

In older modbus devices, the parameter type (READ/WRITE/HOLDING) defines the space from which a parameter is taken. In that case, there is a possibility for defining such a space by adding the appropriate propertis-u to the parameter:

```
<property name=”varspace” value=”output” />
```

The following values are accepted:
– output (0x03)
– input (0x04)
– coil (0x01)

Assigning data access channels

Element name: <access-channel>

The definition assigns one or many data access channels to a given parameter.

Example of definition:

```
<access-channel channel-name="modbus_s1"
                   parameter-id="100"
                   access="fullaccess"/>
```

The above fragment gives access to a defined parameter through the MODBUS protocol at address 100. The *fullaccess* attribute signifies, that the Modbus client using this channel (SCADA) has the right to change this parameter. This is the default value. Access can be limited by changing to READ.

Assigning an event

Element name: <event>

This definition assigns one or many events related to message channels and message content to a given parameter.

Example of definition:

```
<event type="HiAlarm">
        <message-channel channel-name="Email_sender"/>
        <message-id>"Mess_1"</message-id>
        <hysteresis>0.0</hysteresis>
        <property name="trigger" value="0"/>
</event>
```

The above fragment defines messaging in the form of an email (alarm) in the case when a parameter exceeds a value of 0 (HiAlarm type) – e.g. a digital input reaches a logical value of 1. As a result of the fulfillment of the condition, an e-mail with content defined in the *Mess_1* identifier message will be sent.
Example of a message content definition:

```
<message id="Mess_1">
          <![CDATA[
          "Connected to ground"
          ]]>
</message>
```

The message may contain macros beginning with "REG_". They are converted to current attribute values of a given parameter such as its value („REG_VALUE"), name („REG_NAME"), or unit symbol („REG_UNIT"). The value in brackets is the identifier of a given parameter. E.g.:

```
 <message id="Mess_1">
              <![CDATA[
              "TIME
              Rejestr REG_NAME[4001]
              zmienił wartość na REG_VALUE[4001] REG_UNIT[4001]."
              ]]>
        </message>
```

There are also other types of events, e.g. LoAlarm, OnChange or NoChange. More information on this subject can be found in chapter *7.1.7.event structure*

There are also six types of alarms

- Text messages
- E-mail
- Logger
- Script
- Modbus event
- log4j

Detailed instructions for using these alarm types can be found in further chapters of these instructions.

5.2. Data logger

iMod platform has built-in functionality of the data logging on an SD card. Access to the following files, can be obtained through ftp client. Data files are saved in csv format.

Enabling write data

If you want to enable logging, set the appropriate attribute: acquisition_period.
Acquisition_period determines how often data is saved. The value is given in seconds.

```
<imod version="1.1.25" acquisition_period="60">
```

Example above configuration enables data stored in every 60 seconds.

The specified acquisition_period must be an integer number.

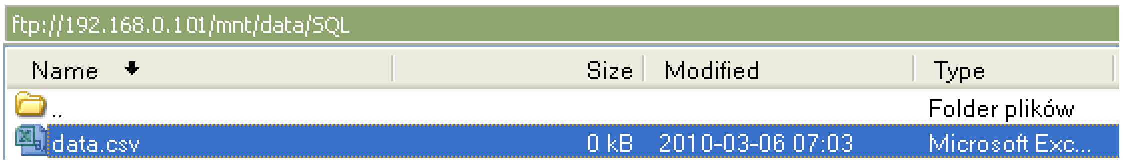
The choice of parameters to be recorded

The collection of data specific parameter, be included by adding the element <parameter> entry:

```
 <data-logging>true</data-logging>
```

Place of saved data (data.csv)

Date are saved into */mnt/data/SQL/* with *.csv extensin.*



ftp://192.168.0.101/mnt/data/SQL

Name	Size	Modified	Type
..			Folder plików
data.csv	0 kB	2010-03-06 07:03	Microsoft Exc...

CSV - format data is stored in text files. Supported by such programs as MS Excel and Open Office Calc. By default is used as a separator is semicolon.

Write to the file format data.csv

Data.csv file consists of values separated by semicolons. In this way, data can be interpreted by the spreadsheet program in the form of columns.

Column headings are created on the basis of the value of other elements in the tag <parameter>.

- Element <id> forms the first part of the header.

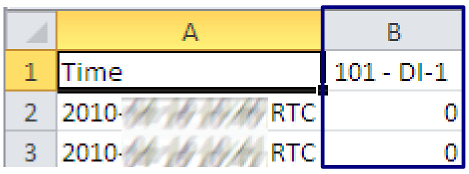
- Element <description> forms the second part of the header.

These elements are separated from each other by a space and hyphen.

Sample MainConfig.xml entry in the file:

```
<parameter>
            <id>
                101
            </id>
             <data-logging>true</data-logging>
            <description>"DI-1" </description>

            <source-channel channel-name="npe_io"
             parameter-id="DI1"/>

            <access-channel channel-name="modbus_s1"
             parameter-id="101"/>
</parameter>
```

create a column with the following header (B):



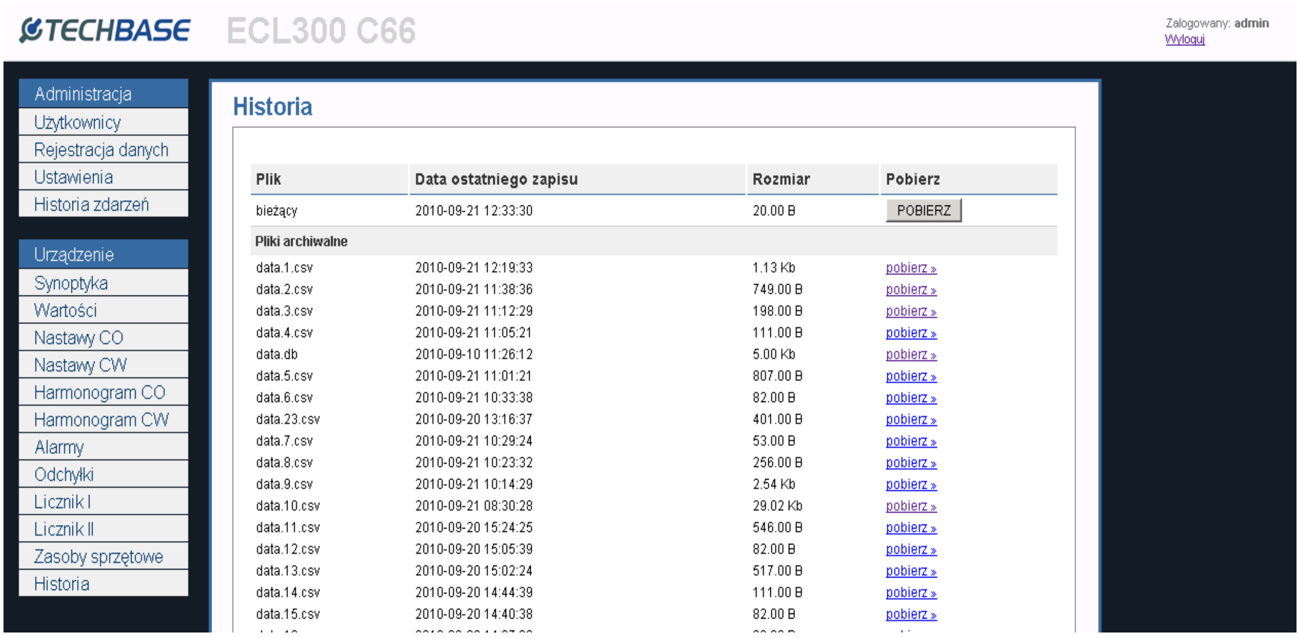
	A	B
1	Time	101 - DI-1
2	2010-[redacted] RTC	0
3	2010-[redacted] RTC	0

WARNING!

Changes in value are not included <data-logging> before restarting the application iMod platform. At the time of the re-launch will create a new file .csv taking into account the changes.

CSV files are also available through a WWW template pre-installed in the iMod.

It is enough to enter the device IP into a browser and go to the History tab:



Historia

Plik	Data ostatniego zapisu	Rozmiar	Pobierz
bieżący	2010-09-21 12:33:30	20.00 B	POBIERZ
Pliki archiwalne			
data.1.csv	2010-09-21 12:19:33	1.13 Kb	pobierz »
data.2.csv	2010-09-21 11:38:36	749.00 B	pobierz »
data.3.csv	2010-09-21 11:12:29	198.00 B	pobierz »
data.4.csv	2010-09-21 11:05:21	111.00 B	pobierz »
data.db	2010-09-10 11:26:12	5.00 Kb	pobierz »
data.5.csv	2010-09-21 11:01:21	807.00 B	pobierz »
data.6.csv	2010-09-21 10:33:38	82.00 B	pobierz »
data.23.csv	2010-09-20 13:16:37	401.00 B	pobierz »
data.7.csv	2010-09-21 10:29:24	53.00 B	pobierz »
data.8.csv	2010-09-21 10:23:32	256.00 B	pobierz »
data.9.csv	2010-09-21 10:14:29	2.54 Kb	pobierz »
data.10.csv	2010-09-21 08:30:28	29.02 Kb	pobierz »
data.11.csv	2010-09-20 15:24:25	546.00 B	pobierz »
data.12.csv	2010-09-20 15:05:39	82.00 B	pobierz »
data.13.csv	2010-09-20 15:02:24	517.00 B	pobierz »
data.14.csv	2010-09-20 14:44:39	111.00 B	pobierz »
data.15.csv	2010-09-20 14:40:38	82.00 B	pobierz »

5.3. Writing to a database

If the iMod is only being used as a proxy/router/gateway modbus and WWW visualization is not used, it is possible to deactivate the writing of values to the database in order to optimize the device.

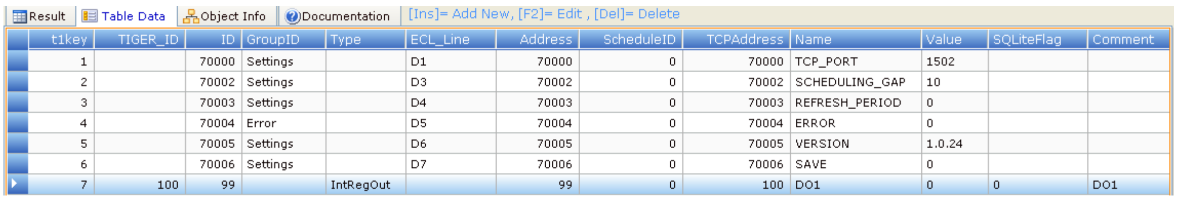
This is realized by adding *parameter-db="false"* the the iMod element e.g.:

```
<imod version="1.0.0" parameter-db="false">
```

Deactivation of this option also results in faster start-up of the platform.

The database can be found on the ramdisk partition, and it is an SQLite3 format base. It can be viewed using the SQLite2009Pro program.

From information included in the table: DATA is used by the pre-installed www page template.



t1key	TIGER_ID	ID	GroupID	Type	ECL_Line	Address	ScheduleID	TCPAddress	Name	Value	SQLiteFlag	Comment
1		70000	Settings		D1	70000	0	70000	TCP_PORT	1502		
2		70002	Settings		D3	70002	0	70002	SCHEDULING_GAP	10		
3		70003	Settings		D4	70003	0	70003	REFRESH_PERIOD	0		
4		70004	Error		D5	70004	0	70004	ERROR	0		
5		70005	Settings		D6	70005	0	70005	VERSION	1.0.24		
6		70006	Settings		D7	70006	0	70006	SAVE	0		
7	100	99		IntRegOut		99	0	100	DO1	0	0	DO1

The SQLite flag column informs the iMod engine of the fact, that the parameter was changed and that it should be overwritten on the output.

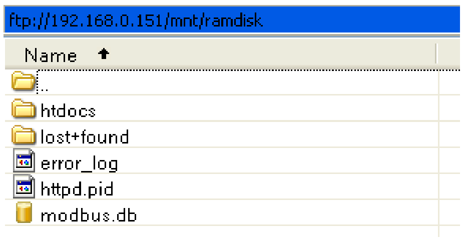
IMod writes to the database only in case of reading when the value has changed.

5.4. WWW template

A WWW template is pre-installed in the device, which, after modifications, can serve as its own visualization of slow-variable processes.

Apache2, with PHP, SQL, and AJAX support has been used as a web-server. The WWW pack, also called APACHE PACKAGE, comprises two directories. One, the data source *htdocs_src* is created in the place of the pack's installation, and the second *htdocs,* is created in the ramdisk where temporarily used files are stored.

In order to immediately observe changes introduced to WWW, they should be made in /mnt/ramdisk/htdocs. However, for these changes to be written, changes should be moved to the directory <apache_installation_directory>/htdocs_src.



ftp://192.168.0.151/mnt/ramdisk

```
Name
..
htdocs
lost+found
error_log
httpd.pid
modbus.db
```

Htdocs directory

The apache installation directory can be checked by using the command:
# getenv APACHE_VERSION

5.5. NxDYNAMICS – FLEX technology interface

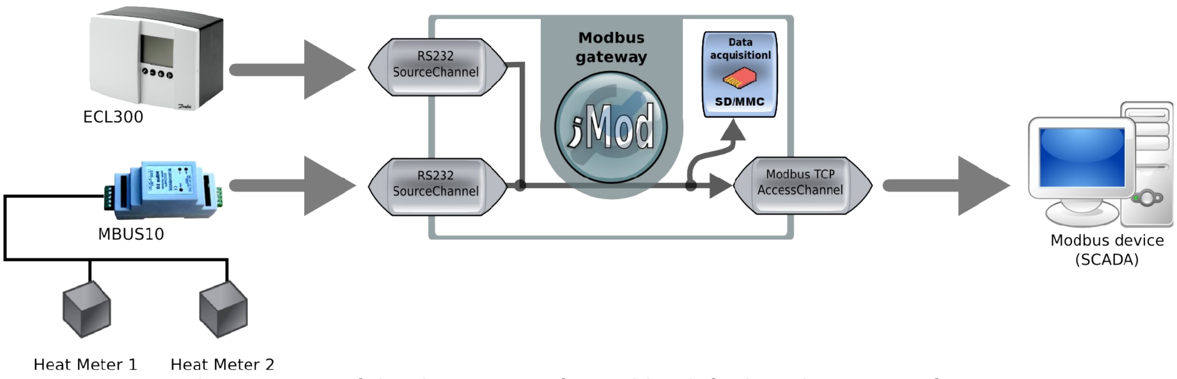
There is also an alternative way of building visualizations using the innovative FLEX technology, which decreases transfer to a minimum, ensuring incomparable aesthetics and safety.

More information on this subject can be found at www.a2s.pl

Chapter 6 TRM Tutorial*

This chapter describes the functionality of all devices with a series of TRM.

Below picture shows a simplified structural diagram of the device is configured by default.



The structure of the device is configured by default with a series of TRM

6.1. Description of the structure of the application configuration

The first element is the file structure of a communication channel configuration Modbus TCP:

```
<access-channel name="modbus_s1">
        <protocol name="MODBUS"/>
        <port>"ET-502-TCP"</port>
        <device-id>1</device-id>
</access-channel>
```

This channel acts as a Modbus TCP slave device on port Modbus address 502 equal to the first

Another element is the structure that supports communication channel configuration protocol service ECL300C66 regulator.

```
<source-channel name="Modbus_M1">
        <protocol name="ECL300C66S"/>
        <port>"C3-1200-8O1"</port>
        <gap>0</gap>
        <cycle>30</cycle>
</source-channel>
```

This channel controller enables communication with the serial port COM3. Transmission parameters are 12008O1. Reading value is held every 30 seconds without any additional break between readings.

The third communication channel supports MBUS protocol on port COM1:

```
<source-channel name="MBUS_HMeters">
        <protocol name="MBUS"/>
        <port>"C1-2400-8E1"</port>
        <gap>0</gap>
        <cycle>60</cycle>
</source-channel>
```

* Chapter describes the functionality of the converter TRM, not apply to an iMod

Transmission parameters are 24008E1. Reading is the value of every minute with no additional break between readings.

Elements named <parameter> determine the conversion of data read from the slave devices.

Caution is advised when modifying elements <parameter>. They are adapted to map the internal registers and control ECL300 MBUS'owych heat meters. More information in the Extras.

6.2. Description of application configuration file structure

The first file element is the configuration structure of the Modbus TCP type communication channel:

```
<access-channel name="modbus_s1">
        <protocol name="MODBUS"/>
        <port>"ET-502-TCP"</port>
        <property value=”device-id” value=”1” />
</access-channel>
```

This channel plays the role of a Modbus TCP slave device on port 502 with a Modbus adddress of 1.

The next element is the configuration structure of the communication channel servicing the service protocol of the ECL300C66 regulator.Kolejnym elementem jest struktura konfiguracyjna kanału komunikacji obsługującego protokół serwisowy regulatora ECL300C66.

```
<source-channel name="Modbus_M1">
        <protocol name="ECL300C66S"/>
        <port>"C3-1200-8O1"</port>
        <gap>0</gap>
        <cycle>30</cycle>
</source-channel>
```

This channel enables communication with the regulator on serial port COM3. Transmission parameters are 12008O1. Value readings are carried out every 30 s with no additional pauses between readings.

The third communication channel services the MBUS protocol on the COM1 port:

```
<source-channel name="MBUS_HMeters">
        <protocol name="MBUS"/>
        <port>"C1-2400-8E1"</port>
        <gap>0</gap>
        <cycle>60</cycle>
</source-channel>
```

Tranmission parameters are 24008E1. Value readings are carried out every minute with not additional pause between readings.

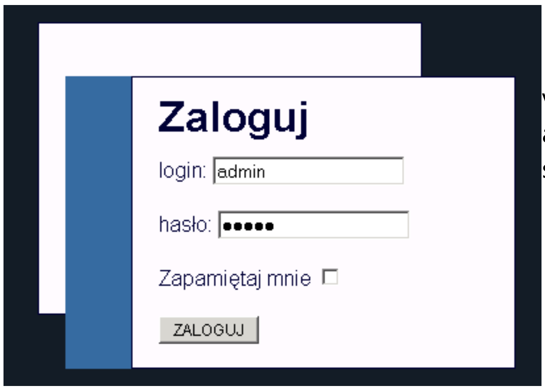
Elements with the name <parameter> define the method of conversion of data read by slave devices.

Change of Address of heat meters

Addresses MBUS heat set in /mnt/mtd/iMod/config file/MBUSMap.xml.
Specify them markers <a2s:Counter addr="XXX">, where XXX is the address counter.

6.3. Managing WWW users

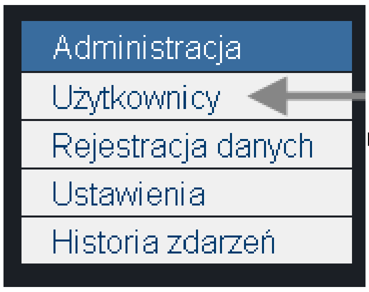
Default user (www)



The TRM converter possesses an integrated visualization available through www browsers. Easy access configuration to the interface by means of a special Panel is possible.

Default settings allow login using:
login: admin

password: admin



The administration Panel contain the “Users” field, which makes it possible to configure access to WWW.

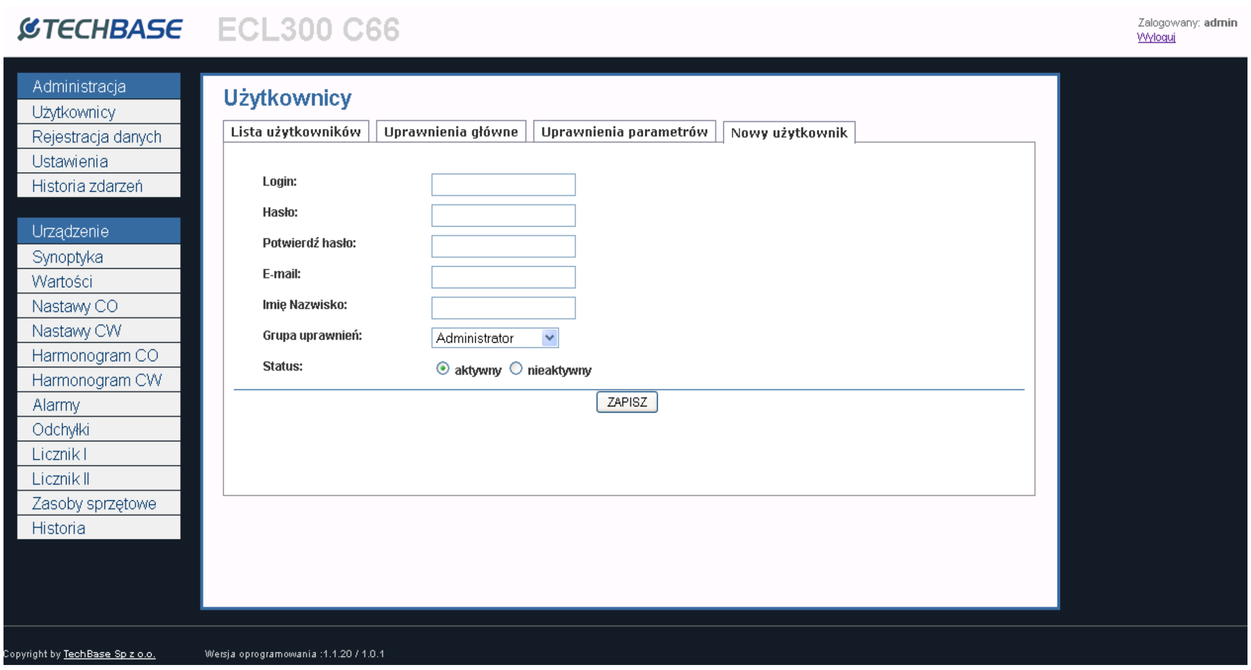
To be able to change the default password, a second user with 'administrator' authorization must be created, after which the default account can be deleted or edited.

Adding users (www)

When adding users, a short form is to be filled out.

All fields are required. In the e-mail field, any address can be input, as long as it has the correct format.

Every user must be assigned to an appropriate authorization group.



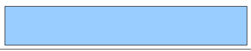
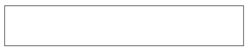
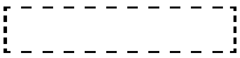
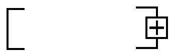
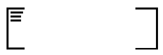
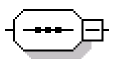
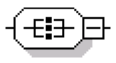
User authorization (www)

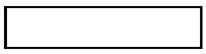
To set user authorization, two special tabs must be used: M*ain authorization, Parameter authorization.* Access to every element of the WWW interface can be configured from there at will for every user group.

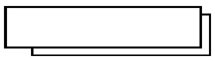
Chapter 7 XML file structure

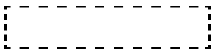
7.1. The structure of the application configuration file

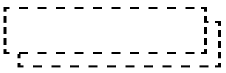
The table below presents the names used in the remainder of the description of the XML structure.

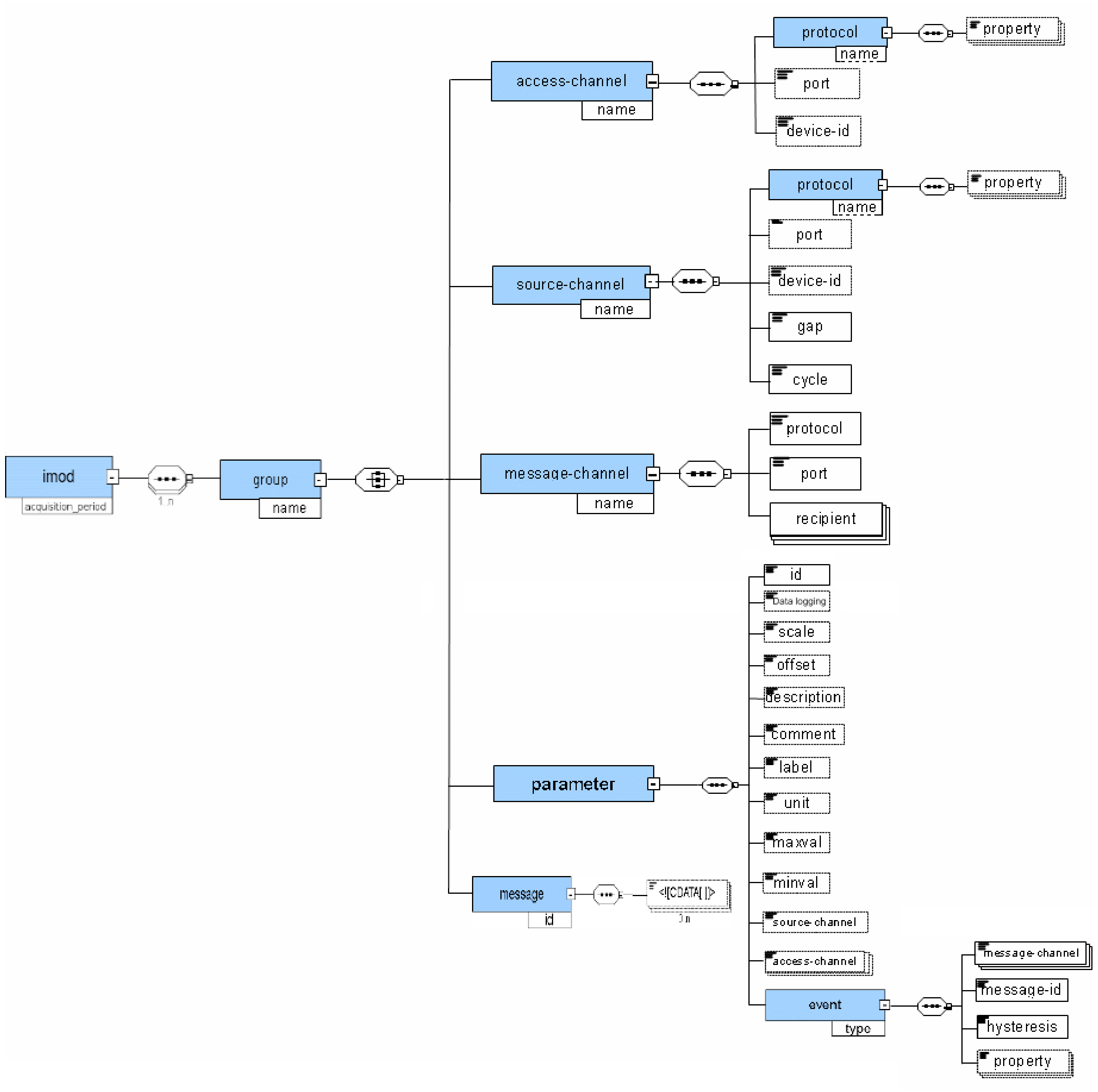
SYMBOL	Element type	Description
	Main	Icludes the definition of other elements. May have additional attributes added.
	Required	Must be at least the first
	Optional	Optional element that can not occur at all in the definition.
	Complex	Contains another elements
	Simple	Does not contain any other elements
SEQUENCE ELEMENTS		
	Any	The order of the definition of the elements can be random.
	Sequential	Elements must appear in a specific order as presented in the documentation.
ELEMENT AMOUNT		

	Required	Min. occurences = 1;
		Max. occurences = 1;


1...n
	Required	Min. occurences = 1;
		Max. occurences = no limits;


0..1
	Optional	Min. occurences = 0;
		Max. occurences = 1;

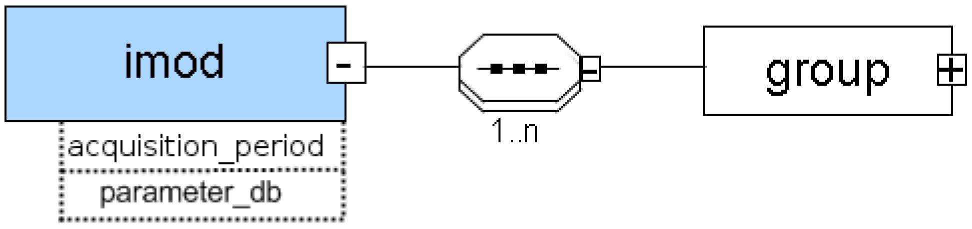

0...n
	Optional	Min. occurences = 0;
		Max. occurences = no limits;



XML class tree

The main element

The main element of the file. May contain several groups (group) configuration parameters.



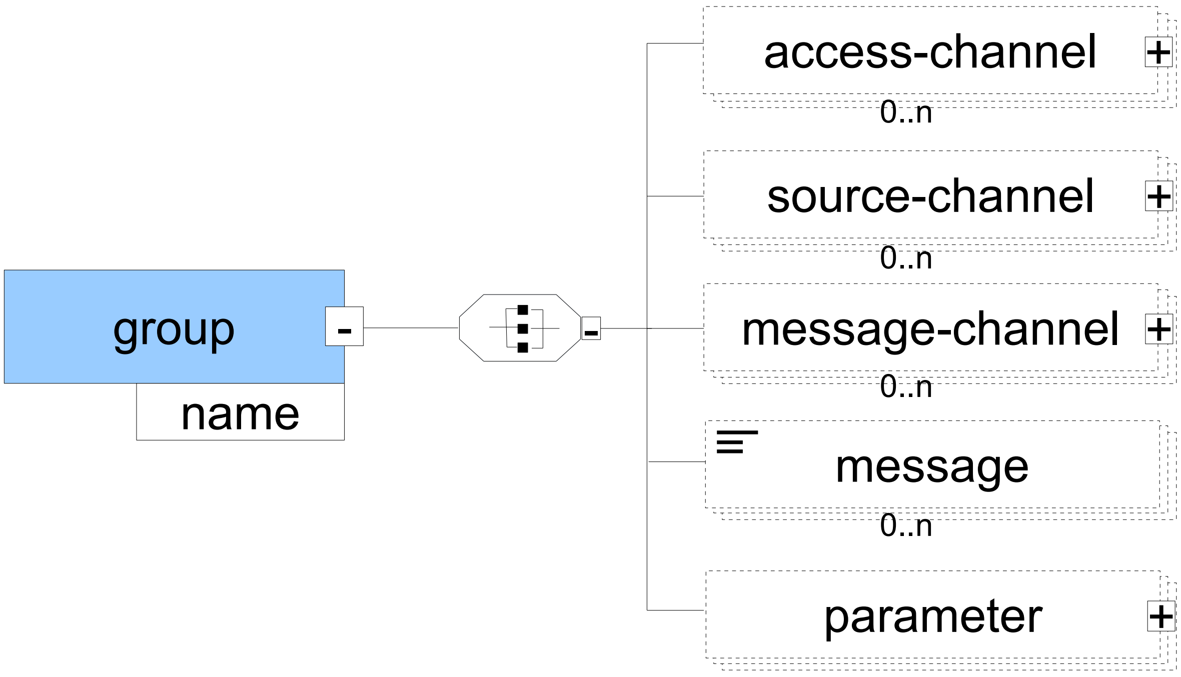
Atribute		
Atribute name	**Type**	**Description**
version	version type	Software version number
acquisition_period	acquisition_period type	The frequency of archiving data in minutes. Do not implement this attribute disables archiving of data
parameter_db	true/false	Activates/deactivates SQL plugin

Elements		
Item Name	**Type**	**Description**
group	Błąd: Nie znaleziono źródła odwołania	Group-defined configuration parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<imod version="1.0.0"  parameter-db="false" acquisition_period="1">
 <group name="Slave_1_parameters">
 [...]
 </group>
</imod>
```

group structure

Group configuration parameters. Each group has its own name, which helps to increase the readability of the configuration file



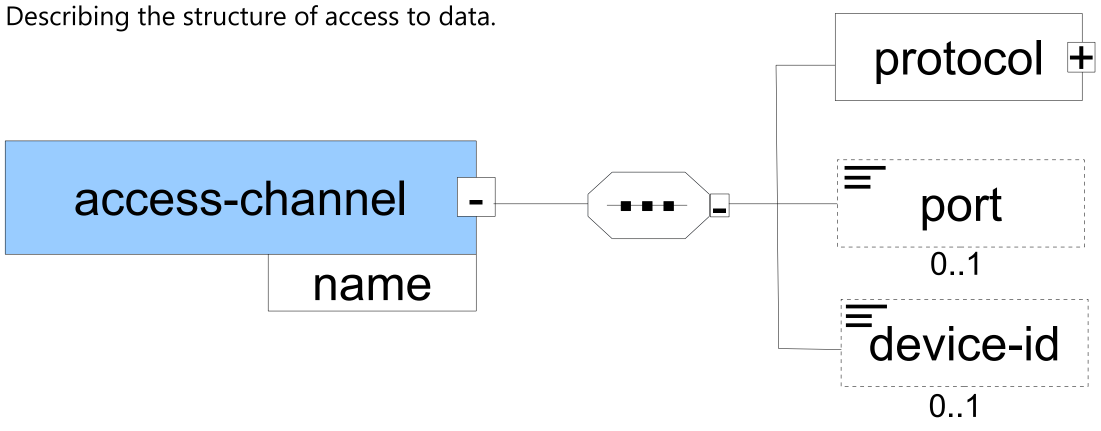
Atribute		
Atribute name	**Type**	**Description**
name	String	Name of the group parameters

Elements		
Item Name	**Type**	**Description**
access-channel	access-channel structure	Definition way of access to data
source-channel	source-channel structure	Definition of source for data
message-channel	message-channel structure	Definition ways of notifications
message	message type	Definition of contain of notification
parameter	parameter structure	Definition of parameter

```
<group name=”channels”>
   <source-channel [...]
   <access-channel [...]
   <message-channel [...]
</group>

<group name=”parameters”>
   <parameter>[...]
</group>
```

access-channel structure

Describing the structure of access to data.



Attribute		
Atribute name	**Type**	**Description**
name	String	Channel name used as an identifier

Elements		
Item Name	**Type**	**Description**
protocol	protocol structure	The type and parameters of the protocol
port	port type	Definition of the port channel
device-id	device-id type	Device ID

'property' element		
Element name	**Values**	**Description**
<property name="type" value="[wartość]" />	RTU/ASCII/MBUS	Communication protocol on access channel. Given values available only for MODBUS type access channels.
<property name="device id" value="[id]" />	integer	Device ID

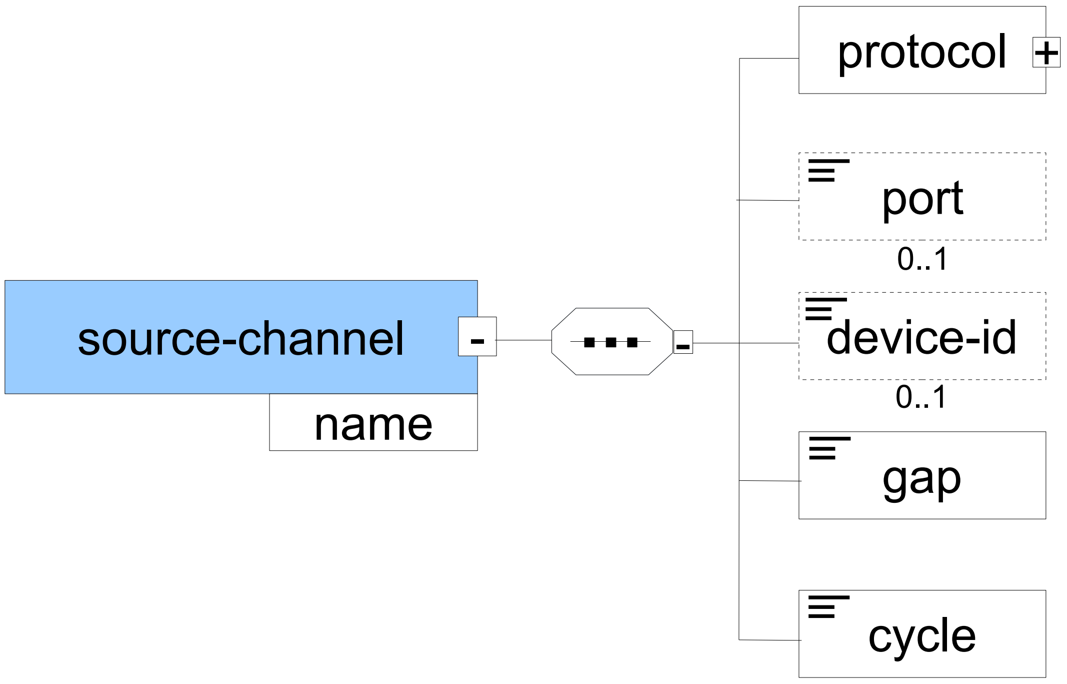
`<access-channel name="Modbus_S1">`
`     <protocol name="MODBUS"/>`
`     <port>"ET-502-TCP"</port>`
`</access-channel>`
	Access using Modbus protocol using device IP on TCP port 502. Device Modbus ID = 1

`<access-channel name="Modbus_S1">`
`     <protocol name="MODBUS"/>`
`     <property name="type" value="RTU" />`
`     <port>"com3-115200-8N1"</port>`
`     <property name="device-id" value="1"/>`
`</access-channel>`
	Access using Modbus protocol using port RS-485 with set parameters.

`<access-channel name="Modbus_S1">`
`     <protocol name="MODBUS"/>`
`     <property name="type" value="ASCII" />`
`     <port>"com0-115200-8N1"</port>`
`</access-channel>`
	Access using Modbus protocol using port RS-232 (com0) with set parameters.

source-channel structure

Channel structure describing the data source.



Attribute		
Atribute name	**Type**	**Description**
name	String	Channel name used as an identifier

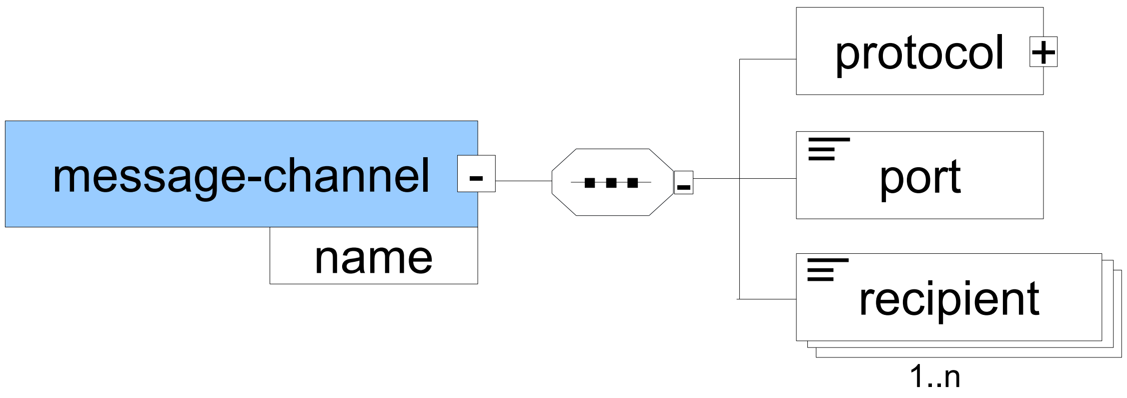
Elements		
Item Name	**Type**	**Description**
protocol	protocol structure	The type and parameters of the protocol
port	port type	Definition of the port channel
Property	Property type	Dodatkowe właściwości
gap	Integer	In the case where a regular integer is given, it refers to a value in seconds. These values can be changed by adding a suffix to the number h, ms, m. e.g. 300ms.
cycle	Integer	
delay	Integer	

'property' element		
Element name	**Values**	**Description**
<property name="type" value="[value]" />	RTU/ASCII/MBUS	Communication protocol on access channel. Given values are available only for MODBUS access channels.
<property name="device id" value="[id]" />	String	Device ID address

`<source-channel name="NPE_Hardware_Resources"> <protocol name="Hardware" /> <gap>0</gap> <cycle>2</cycle> </source-channel>`	Reading NPE computer hardware with parameters: 0s. Gaps between readings. 2 seconds for reading of the whole group.
`<source-channel name="NPE_Hardware_Resources"> <protocol name="Hardware" /> <gap>55m</gap> <cycle>1h</cycle> </source-channel>`	Reading NPE computer hardware resources with parameters: 55m gap between readings with reading refresh every hourOdczytywanie
`<source-channel name="Modbus_Slave"> <protocol name="MODBUS" /> <port>"ET-192.168.0.201-502"</port> <gap>0</gap> <cycle>30</cycle> <delay>300ms</delay> </source-channel>`	Reading using Modbus-TCP, with pauses between polling of subsequent modbus addresses every 300 milliseconds, with 30 secods for the entire reading cycle. Data collected from device TCP protocol with IP address 192.168.0.201.
`<source-channel name="Modbus_Slave"> <protocol name="MODBUS"> <property name="type" value="RTU"/> </protocol> <port>"C1-115200-8N1"</port> <gap>0</gap> <cycle>30</cycle> <delay>100ms</delay> </source-channel>`	Reading using Modbus-RTU, with pauses between polling of subsequent modbus addresses every 100 milliseconds, with 30 seconds for the entire reading cycle. Data collected from port RS-232(RX1/TX1/GND).
`<source-channel name="Modbus_Slave"> <protocol name="MODBUS"> <property name="type" value="ASCII"/> </protocol> <port>"C3-115200-8N1"</port> <gap>0</gap> <cycle>30</cycle> <delay>100ms</delay> </source-channel>`	Reading using Modbus-ASCII with pauses between polling of subsequent modbus addresses every 100 milliseconds, with 30 seconds for the entire reading cycle. Data collected from second RS-232 port (RX3/TX3/GND).
`<source-channel name="Modbus_Slave"> <protocol name="MODBUS"> <property name="type" value="RTU"/> </protocol> <port>"C4-9200-7E1"</port> <gap>0</gap> <cycle>30</cycle> <delay>1s</delay> <property name="device-id" value="1"/> </source-channel>`	Reading using Modbus-RTU, with pauses between polling of subsequent modbus addresses every 1 second, with 30 seconds for the entire reading cycle. Data collected from device connected to port RS-485 (A/B) with ID number 1.

message-channel structure

The structure of the channel messages. Allows you to send messages related to specific events.



Atribute		
Atribute name	**Type**	**Description**
name	String	Channel name used as an identifier

Elements		
Item Name	**Type**	**Description**
protocol	protocol structure	The type and parameters of the protocol
port	port type	Port Definition
recipient	recipient type	Recipient of a message ID

```
<message-channel channel-name="SQL_saver">
                    <protocol name="SQL"/>
                    <port>"/mnt/data/SQL/data.db"</port>
                    <recipient>"alarmy"</recipient>
               </message-channel>
```

SQL

```
<message-channel channel-name="Unit_updater">
                    <protocol name="logger"/>
                    <port>"file"</port>
                    <recipient>"info"</recipient>
          </message-channel>
```

LOGGER

```
 <message-channel channel-name="Script_exec">
                     <protocol name="script"/>
                     <port>"/mnt/mtd/iMod/config"</port>
                     <recipient>"difference"</recipient>
 </message-channel>
```

SCRIPT

```
<message-channel channel-name="email_sender">
                      <protocol name="EMAIL">
          <property name="user" value="testnpe"/>
          <property name="password" value="123npe"/>
                      </protocol>
             <port>"poczta.o2.pl"</port>
            <recipient>"techbase@a2s.pl"</recipient>
 </message-channel>
```

E-mail

```
<message-channel channel-name="sms_sender">
      <protocol name="sms"/>
      <recipient>"48123456789"</recipient>
</message-channel>
```

Text Message

```
<message-channel channel-name="Gateway">
     <protocol name="forcewrite">
         <property name="gateway" value="true"/>
      </protocol>
</message-channel>
```

Modbus forcewrite

```
<message-channel channel-name="Gateway">
     <protocol name="forceread">
         <property name="gateway" value="true"/>
      </protocol>
</message-channel>
```

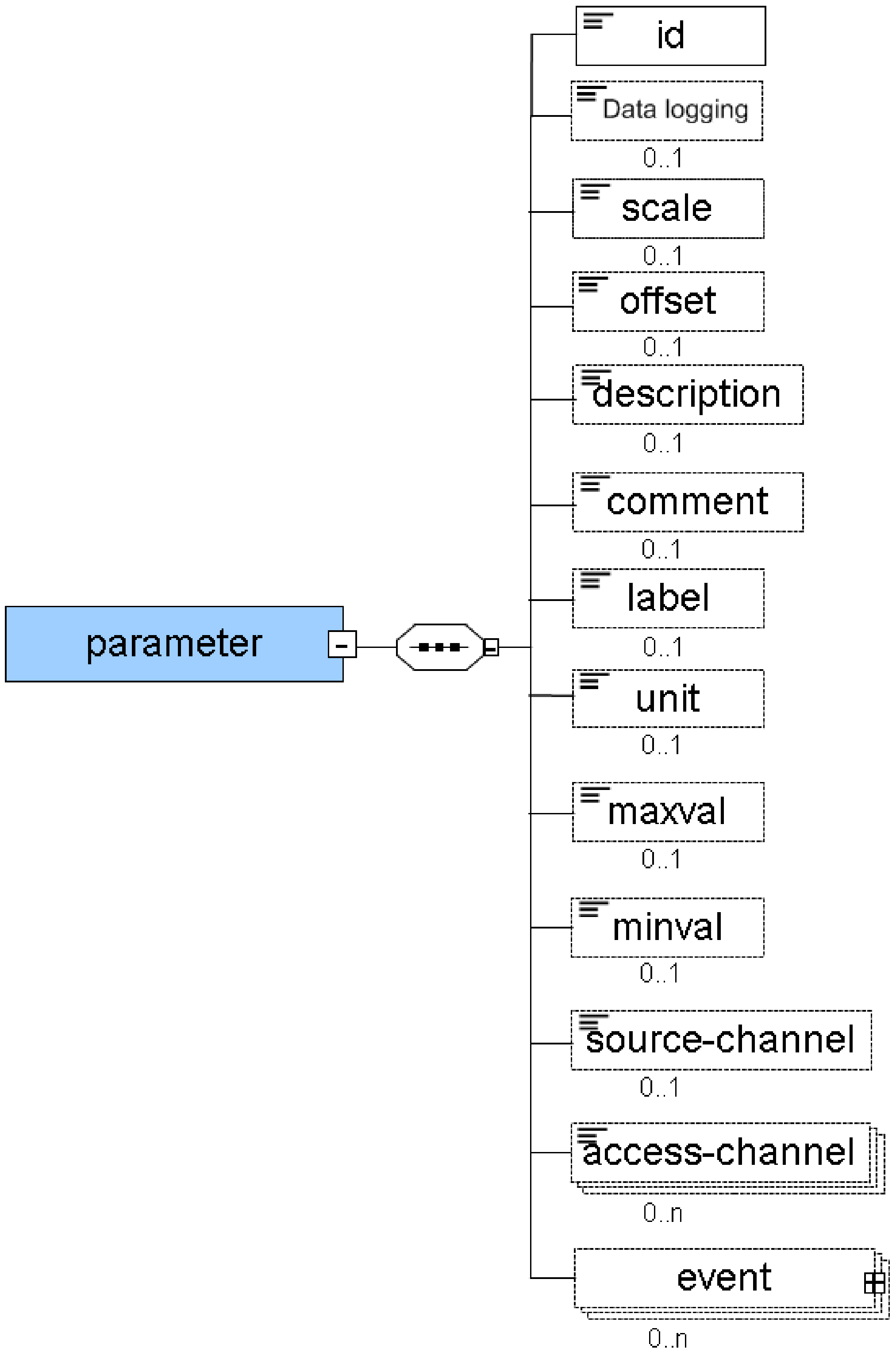
Modbus forceread

```
<message-channel channel-name="Modbus_sender">
         <protocol name="modbus"/>
         <port>"ET-192.168.0.151-502-TCP"</port>
         <property name="device-id" value="1" />
 </message-channel>
```

Modbus [push]

parameter structure

The structure definition of the parameter. Constitutes an intermediary between data sources and channels of access channels.



Element			
Element name	**Type**	**Description**	**Required value/defoult?**
id	id type	Unique identifier parameter	YES
scale	scale type	Scale factor values for	NO/1
offset	offset type	Transfer coefficient values for	NO/0
description	description type	Parameter description	NO/undeftxt
comment	comment type	Comment	NO/undeftxt
label	label type	Label	NO/undeftxt
unit	unit type	Symbol Unit	NO/undeftxt
maxval	maxval type	The maximum value	NO/undeftxt
minval	minval type	The minimum value	NO/undeftxt
data-logging	data-logging type	If an element is to be recorded on SD.	NO/FALSE
source-channel	source-channel-type	The connection to the channel data source	NO/none
access-channel	access-channel-type	Relationship with the channel access	NO/none
event	event structure	Event Definition	NO/none

`<parameter>`
`   <id>101</id>`
`   <data-logging>true</data-logging>`
`   <description>"DI-1" </description>`
`   <source-channel channel-name="npe_io"`
`             parameter-id="DI1"`
`             parameter-type="WRITE"/>`
`</parameter>`
	Hardware
	Definition of digital output #1 parameter, with data registration to a CSV file. The column heading will have the title "DI-1".
	This parameter will be possible to overwrite.

`<parameter>`
`   <id>101</id>`
`   <source-channel channel-name="ModbusTCPSlave"`
`             parameter-id="10"/>`
`</parameter>`
	Modbus TCP
	int16 type parameter definition (default value) read from slave Modbus with a device address of 1 (default value).
	However the 10 modbus register is read.
	This value is write type (by default).

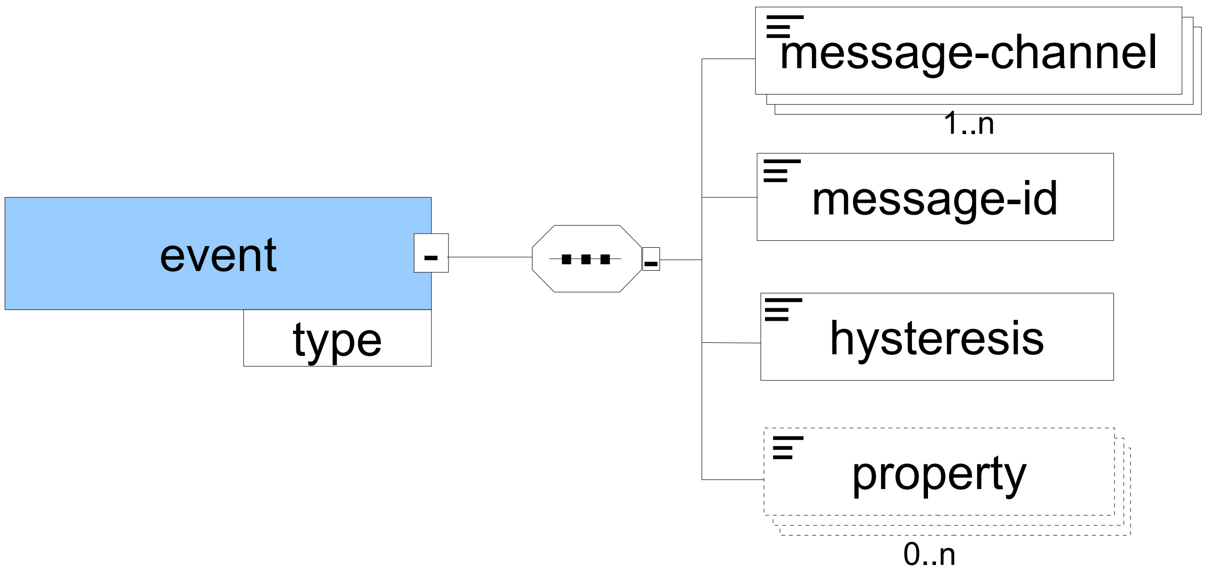
`<parameter type="float>`
`  <id>101</id>`
`  <source-channel channel-name="ModbusRTUSlave"`
`             parameter-id="5:10"`
`             parameter-type="READ"/>`
`</parameter>`
	Modbus RTU
	Float type parameter definition
	(variable decimal) read from slave Modbus with device address 5 and its 10 register.
	This is a read-only parameter.

`<parameter>`
`<id>"THERM1_1"</id>`
`<description>"THERMOMETER_DS18B20"</description>`
`<source-channel channel-name="OneWire" parameter-id="28CFF1C9020000:temperature"/>`
`</parameter>`
	OneWire

`<parameter type="int32">`
`      <id>"4621"</id>`
`      <scale>10e-2</scale>`
`      <comment>"INST_VAL"</comment>`
`      <unit>"m^3"</unit>`
`      <description>"VOLUME"</description>`
`      <source-channel channel-name="MBUS_com0"`
` parameter-id="2-2"/>`
`      <access-channel channel-name="Modbus_SMBUS"`
`parameter-id="104"/>`
`</parameter>`
	MBus

event structure

Event-driven structure definition. With the channel set up a system alert messages, and Event-driven communications.



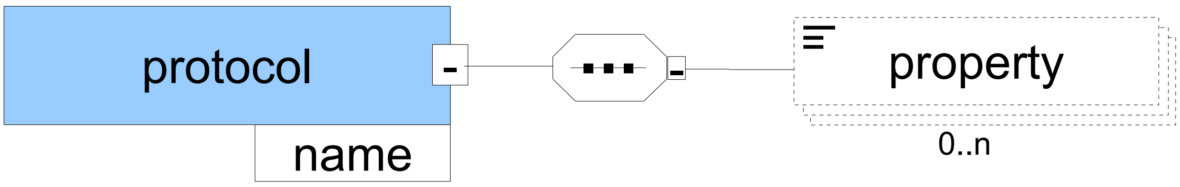
Atribute		
Atribute name	**Type**	**Description**
type	event type	Event Type

Elements		
Item Name	**Type**	**Description**
message-channel	message-channel type	Communication channel associated with the event
message-id	message-id type	Message id
hysteresis	hysteresis type	Hysteresis
property	property type	Event parameters defining type (Acceptable values are trigger or id).

protocol structure

The structure definition of the protocol and its parameters.

Defines a protocol which is used to implement access or source Chanell. This is an essential element in the definition of the channel. There are six types of protocol: Modbus, Mbus, Hardware, SMS, Email, SQL.



Atribute		
Atribute name	**Type**	**Description**
name	protocol-name type	Protocol type

Elements		
Item Name	**Type**	**Description**
property	property type	Additional parameter of protocol

version type

Software version number

Used by	Base type	Permissible empty value ?	Regular expression	Description
The main element	String	NO	[0..9]+\.[0..9]+\.[0..9]+	The three integers separated by dots, eg 0.1.18

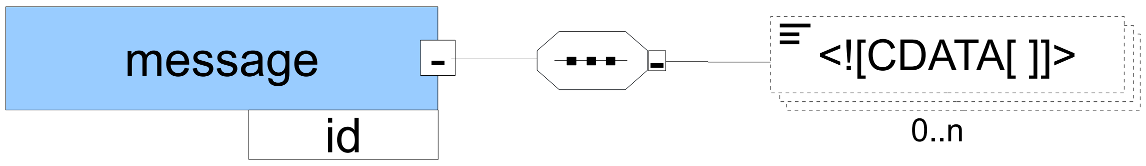
acquisition_period type

The frequency of archiving data in minutes. None of this attribute disables archiving of data

Used by	Base type	Permissible empty value ?	Regular expression	Description
The main element	Integer	NO	[1..9]+[0..9]*	Positive integer

message type

The content of messages placed in the section <! [CDATA [ ]]>.



Atribute name	Type	Description	Format
id	id_type	ID currently used in other XML elements	String

Used by	Base type	Permissible empty value ?	Regular expression	Description
Błąd: Nie znaleziono źródła odwołania	String	NO		A string with end of line markers and specific markers indicating the current value of the parameter: REG_NAME [id] - the name of the parameter d REG_VALUE [id] - the current value of the parameter d REG_UNIT [id] - the unit symbol for the parameter d REG_LABEL [id] - the label of the parameter d

port type

Type specifying the nature of the communication channel port and its parameters. The parameter can be described in terms of such passages.

Used by	Base type	Permissible empty value ?	Regular expression	Description
access-channel structure source-channel structure	String	NO	^ET-([a-zA-Z_0-9]+)(-([0-9]{1,5}?))?-(TCP|UDP)$ ^C([1-9])-(300|600|1200|1800|2400|4800|9600|19200|38400|57600|115200|230400)-([5-8])([OEN])([1-2])-(A|R)	The exact description of the possible values are presented below.
message-channel structure	String	NO	EMAIL	*Address mail server*
			SMS	*GSM modem serial port*
			SQL	The path to the SQLite database file

Channel type	Protocol name type	port type: A-B-C							
		A		B		C			
		Value	Description	Value	Description	Value	Description		
source/access channel	MBUS, MODBUS, HARDWARE	ET	Data will be sent by Ethernet or GPRS	1-65535	Port number	TCP	Dane przesyłane za pomocą protokołu TCP		
						UDP	Dane przesyłane za pomocą protokołu UDP		
		comX	Use com port where X it is number from 0-3*	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400	Speed of transmission	8/7 N/E/O/M/S 1/2 np. 8N1-R	Parametry portu COM	R - RTU A - ASCII	Rodzaj transmisji**

Channel type	Protocol name type	Port type for message channel	
		wartość	opis
message-channel structure	EMAIL	E-mail sever for example *poczta.o2.pl*	SMTP email server
	SQL	*/mnt/data/SQL/data.csv*	Path to SQLite database

Table 5: Format of a channel-port

**- Assuming no symbolic links and multiplexing ports*

***- Only if Modbusmessage-port protocol type*

device-id type

A unique identifier for the channel.

Used by	Base type	Permissible empty value ?	Regular expression	Description
access-channel structure source-channel structure	Integer	No		Positive integer

gap type

Forced break between cycles to read data [ms]

Used by	Base type	Permissible empty value ?	Regular expression	Description
source-channel structure	Integer	No	[1..9]+[0..9]*	Positive integer

cycle type

The time between successive cycles of reading data [s]

Used by	Base type	Permissible empty value ?	Regular expression	Description
source-channel structure	Integer	No	[1..9]+[0..9]*	Positive integer range 1-65535

recipient type

Id recipient notification

Used by	Base type	Permissible empty value ?	Regular expression	Description
message-channel structure	String	No		Positive integer range 1-65535

Message Channel Protocol Type *	Description of the format type of recipient
EMAIL	Recipient's email address
SMS	Recipient's phone number
SQL	Name of the table in the database

* the value of name attribute in the element protocol

id type

Unique id parameter.

parameter structure	Integer	No	[1..9]+[0..9]*	Positive integer range 1-65535

scale type

The scale is a real number which is a multiplier for parameter read out from the protocol.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	Float	No	[1..9]+[0..9]*(\b[0-9]+\.([0-9]+\b)?|\.[0-9]+\b)	Real number with sign

offset type

The number added to the parameter read out from the protocol.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	Float	No	[1..9]+[0..9]*(\b[0-9]+\.([0-9]+\b)?|\.[0-9]+\b)	Real number with sign

description type

Description sent to the database, creating a column header. Csv file

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	String	Nie		String

comment type

Comments placed in the database. Xml synonymous with commentary.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	String	No		string

label type

The label passed to the database used by the web interface.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	String	No		string

unit type

Mark units in which values are measured.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	String	No		string

minval type

It occurs in the element event. It is used to determine a minimum value.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	Float	No	[1..9]+[0..9]*(\b[0-9]+\.([0-9]+\b)?|\.[0-9]+\b)	Real number with sign

maxval type

It occurs in the element event. Used to determine the maximum value.

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	Float	No	[1..9]+[0..9]*(\b[0-9]+\.([0-9]+\b)?|\.[0-9]+\b)	Real number with sign

source-channel-type

Attribute name	Description	Format
channel-name	The name of the parameter associated with the channel data source	String
parameter-id	Depending on the type of protocol identifier for the channel data source parameter value	Description contains Table 5: Description of permitted attribute values for the parameter-id
parameter-type	Description contains Table: Description of the value of the parameter-type attribute	

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	None	yes (no value)	None	No value

protocol structure source-channel	Description of the format attribute, parameter-id
MODBUS	The total value of 1-65535 - Modbus register address to read
MBUS	The total value of 1-65535. Mapping the address to the value of a parameter contains a file MBUSMap.xml MBUS
HARDWARE	AI1 - AI3 - read values from the analog inputs AI1 - AI3 AIV - reading values from the analog input AIV DI1 - DI7 - read digital inputs DI1 - DI7 DIW - read digital input DIW DO1 - DO2 - read status and control of digital outputs DO1 - DO2 PO1 - PO4 - read status and control of digital outputs PO1 - PO4 USER_LED - read status and control LED USER YEAR - year reading the system date MONTH - month to read the system date DAY - Reading of the month the system date HOUR - hour reading the system date MINUTE - minutes reading the system date

Description of permitted values for the attribute parameter-id

access-channel-type

Nazwa atrybutu	Opis	Format
channel-name	The name of the parameter associated with the channel access	String
parameter-id	Depending on the type of protocol data access channel ID parameter value	Description contains Table: Description of allowed parameter values for the attribute-id
parameter-type	Description contains Table: Description of the value of the parameter-type attribute	

Used by	Base type	Permissible empty value ?	Regular expression	Description
parameter structure	none	Yes (no value)	None	No value

Channel Protocol Type of data source	Description of the format attribute, parameter-id
MODBUS	The total value of 1-65535 - Modbus register address to read

Description of permitted attribute values for the parameter-id

The value of parameter-type attribute	Description of the format attribute, parameter-type
HOLD or READ	The value of read-only
WRITE	The value to read and write

Description of the value of the parameter-type attribute

event type

Used by	Base type	Permissible empty value ?	Regular expression	Description
event structure	Ciąg znaków	No	none	Description contains below tabel.

Available value	Description
LoAlarm	Event generated when the current value of the parameter is less than the set threshold. The threshold is set by an element Property: <property name="trigger" value="32.0"/> The value attribute is set threshold
HiAlarm	Event generated when the current value of the parameter is greater than the set threshold. The threshold is set by an element of Property: <property name="trigger" value="32.0"/> The value attribute is set threshold

Allowed values for the type of event type

message-channel type

Atribute name	Description	Format
name	The name of the parameter associated with the communications channel	String

Used by	Base type	Permissible empty value ?	Regular expression	Description
event structure	String	Yes (no value)	'trigger' or 'id'	In the case of not properly expression value is set to "0"

message-id type

Used by	Base type	Permissible empty value ?	Regular expression	Description
event structure	String	NO	none	Message ID which is to be sent with the activation event

hysteresis type

Used by	Base type	Permissible empty value ?	Regular expression	Description
event structure	Float	NO	none	Hysteresis Activation Event

property type

Atribute name	Description	Format
name	Name property	String
value	Value of property	String

Used by	Base type	Permissible empty value ?	Regular expression	Description
event structure protocol structure	String	Yes (no value)	none	No value

protocol-name type

Protocol name for the type of communication channel (channel structure).

Allowed values	Description	For the types of channels:
MODBUS	Defines the Modbus protocol, and opens a connection modbus master.	source-channel
MBUS	MBUS protocol is used	
HARDWARE	With access to hardware resources	
MODBUS	Określa protocol modbus modbus slave mode.	access-channel
EMAIL	Specifies the notification via e-mail	message-channel
SMS*	Specifies the notification via SMS. Required GPRS Module	
SQL	Used to communicate with interefejsem www.	

Used by	Base type	Permissible empty value ?	Regular expression
protocol structure	String	Yes (No value)	No

data logging_type

Used by	Base type	Permissible empty value ?	Regular expression	Description	Used by
parameter structure	string	false	No (no value)	[true|false]	definition of data recording (or data will be recorded)

Chapter 8 Other information

This chapter describes examples of **iMod** platform configurations.

8.1. Supported protocols

The iMod platform supports the following protocols by default:

Modbus TYPE	***ASCII***	***RTU***	***TCP***
RS-232	+	+	-
RS-485	+	+	-
TCP	-	-	+
UDP	-	-	+

- Mbus (conversion to any Modbus type)
- *SNMPv3

*this protocol will be added soon

8.2. Protocol scanning

The iMod has integrated mechanisms for scanning Mbus and 1-wire protocols. They can be used using the command:

imod scan mbus

imod scan onewire

The device can be expanded with scanning of additional protocols. In order to do this, use the SDK being created in the JAVA language. More information on this subject can be found in the technical section.

8.3. Connecting MBUS meters

Heat-meters-ARM

After physically connecting the MBUS meter to the iMod device, check that the connection is correct. This is done using the application:

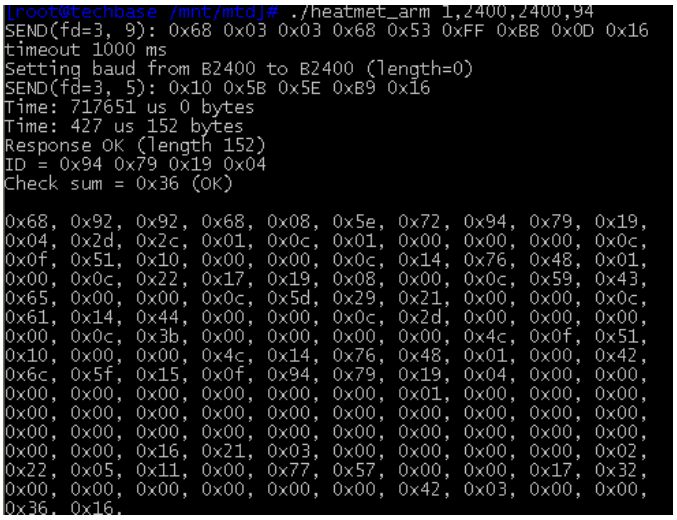
heatmet_arm

It is to be copied to the device and the following commands are to be executed:

```
$ chmod 754 heatmet_arm
$ ./heatmet_arm 1,2400,2400,-1
```

Heatmet_arm <com_number>,<check_baudrate>,<set_baudrate>,<scan mode/ID>
Give the COM port number to which the meter has been connected as the first parameter.
The second value is that of the baudrate used by the meter during its operation.
The third value is the communication speed set on the meter.
The fourth parameter can accept a value from the range of : -1 to 255. -1 scans all address in search of a meter. Instead of the -1 value, it is recommended to enter the ID of the connected meter.

If you receive a CHECK sum message (OK):



```
[root@techbase /mnt/mtd]# ./heatmet_arm 1,2400,2400,94
SEND(fd=3, 9): 0x68 0x03 0x03 0x68 0x53 0xFF 0xBB 0x0D 0x16
timeout 1000 ms
Setting baud from B2400 to B2400 (length=0)
SEND(fd=3, 5): 0x10 0x5B 0x5E 0xB9 0x16
Time: 717651 us 0 bytes
Time: 427 us 152 bytes
Response OK (length 152)
ID = 0x94 0x79 0x19 0x04
Check sum = 0x36 (OK)

0x68, 0x92, 0x92, 0x68, 0x08, 0x5e, 0x72, 0x94, 0x79, 0x19,
0x04, 0x2d, 0x2c, 0x01, 0x0c, 0x01, 0x00, 0x00, 0x00, 0x0c,
0x0f, 0x51, 0x10, 0x00, 0x00, 0x0c, 0x14, 0x76, 0x48, 0x01,
0x00, 0x0c, 0x22, 0x17, 0x19, 0x08, 0x00, 0x0c, 0x59, 0x43,
0x65, 0x00, 0x00, 0x0c, 0x5d, 0x29, 0x21, 0x00, 0x00, 0x0c,
0x61, 0x14, 0x44, 0x00, 0x00, 0x0c, 0x2d, 0x00, 0x00, 0x00,
0x00, 0x0c, 0x3b, 0x00, 0x00, 0x00, 0x00, 0x4c, 0x0f, 0x51,
0x10, 0x00, 0x00, 0x4c, 0x14, 0x76, 0x48, 0x01, 0x00, 0x42,
0x6c, 0x5f, 0x15, 0x0f, 0x94, 0x79, 0x19, 0x04, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x16, 0x21, 0x03, 0x00, 0x00, 0x00, 0x00, 0x02,
0x22, 0x05, 0x11, 0x00, 0x77, 0x57, 0x00, 0x00, 0x17, 0x32,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x42, 0x03, 0x00, 0x00,
0x36, 0x16,
```

This means that you may proceed to the next step. Sometimes meters require multiple pollings before they start to respond at the proper rate and CheckSUM reaches and OK status.

If this is not the case, check if the LEDs on the Mbus-10 signalize obtaining a response from the meter.

Mbus.jar

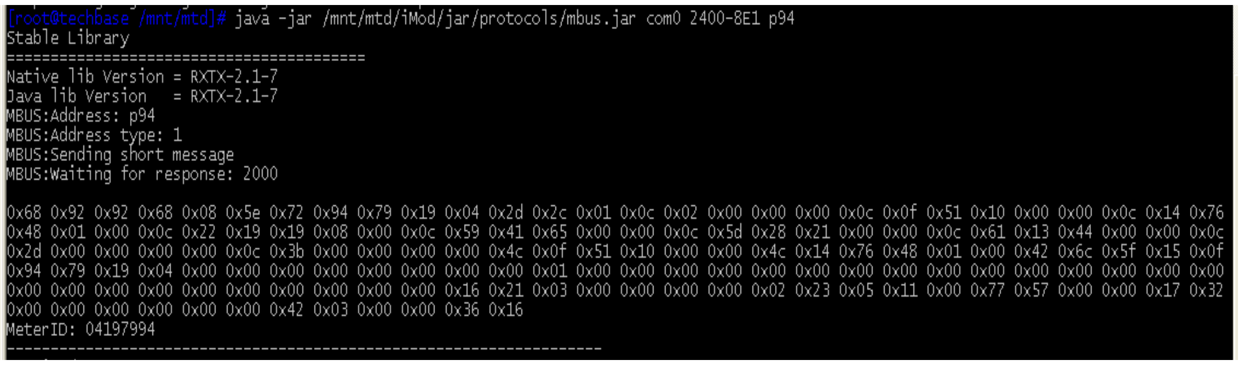
The next step is to check if the Mbus library detects the meter.

Example of query:

```
$ java -jar /mnt/mtd/iMod/jar/protocols/mbus.jar com0 2400-8E1 p94
```

com0 – port number to which the meter is connected to
Available ports:
com0 (RS-232 #1)
com1 (RS-232 #2)
com3 (RS-485)
2400 – connection baudrate.
8E1 – connection properties
p94 – address type (primary – p / secondary – s). 94 – device id. Any value within the range of 0 to 255.

If the library supports the meter, it will display frame decoding:



```
[root@techbase /mnt/mtd]# java -jar /mnt/mtd/iMod/jar/protocols/mbus.jar com0 2400-8E1 p94
Stable Library
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
MBUS:Address: p94
MBUS:Address type: 1
MBUS:Sending short message
MBUS:Waiting for response: 2000

0x68 0x92 0x92 0x68 0x08 0x5e 0x72 0x94 0x79 0x19 0x04 0x2d 0x2c 0x01 0x0c 0x02 0x00 0x00 0x00 0x0c 0x0f 0x51 0x10 0x00 0x00 0x0c 0x14 0x76
0x48 0x01 0x00 0x0c 0x22 0x19 0x19 0x08 0x00 0x0c 0x59 0x41 0x65 0x00 0x00 0x0c 0x5d 0x28 0x21 0x00 0x00 0x0c 0x61 0x13 0x44 0x00 0x00 0x0c
0x2d 0x00 0x00 0x00 0x00 0x0c 0x3b 0x00 0x00 0x00 0x00 0x4c 0x0f 0x51 0x10 0x00 0x00 0x4c 0x14 0x76 0x48 0x01 0x00 0x42 0x6c 0x5f 0x15 0x0f
0x94 0x79 0x19 0x04 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x16 0x21 0x03 0x00 0x00 0x00 0x00 0x00 0x02 0x23 0x05 0x11 0x00 0x77 0x57 0x00 0x00 0x17 0x32
0x00 0x00 0x00 0x00 0x00 0x00 0x42 0x03 0x00 0x00 0x36 0x16
MeterID: 04197994
--------------------------------------------------------------------
```

When the library detects the addresses, the scanning phase can be initiated.

iMod scan mbus

The newest iMod version supports the process of scanning the MBUS protocol on all serial ports. In order to activate this mechanism, the following command must be executed:

```
$ imod scan mbus
```

At this point, the device will begin scanning with a default meter response time-out: 2000 ms. After scanning, the MbusScan.xml log file will be created.

This file contains an example of a configuration reading all parameters available for scanning.

MbusScan.xml

The structure of the MbusScan.xml file is consistent with the structure of the MainConfig.xml configuration file. However, the file generated by default requires processing by the user for the purpose of obtaining a specific configuration.

Access channel

An exemplary*access-channel* has been generated in the file:

```
<access-channel name="Modbus_SMBUS">
          <protocol name="MODBUS"/>
          <port>"ET-1502-TCP"</port>
 </access-channel>
```

This means that Mbus parameters will be accessible at TCP port: 1502
It is recommended to use the default TCP port: 502, defined by default in the iMod configuration, during merging:

```
   <access-channel name="Modbus_S1">
         <protocol name="MODBUS">
             <property name="type" value="rtu"/>
         </protocol>
         <port>"ET-502-TCP"</port>
   </access-channel>
```

Next, carry out refactorization in parameters:
Modbus_SMBUS -> Modbus_S1

SourceChannel

The file has generated an exemplary source-channel:

```
<source-channel name="MBUS_com0">
          <protocol name="MBUS"/>
          <port>"com0-2400-8E1"</port>
          <property name="device-id" value="51-MODEL_MULTICAL601"/>
          <property name="device-id" value="94-MODEL_MULTICAL"/>
          <gap>0</gap>
          <cycle>60</cycle>
</source-channel>
```

The above definition signifies that two meters have been detected:

Multical 601 with ID 51

Multical 66C with ID 94.

Reading will take place every 60 seconds (<cycle>60</cycle>)

ID

In the case where the meter parameter has already been implemented in the TRM template, the MBUSScan.xml file will contain an appropriate ID for the parameter that has been adapted for WWW.

```
<parameter type="int32">
        <!--Actual value: 06605251 (BCD:8)-->
        <id>"4519"</id>
        <comment>"INST_VAL"</comment>
        <unit>"(unkown)"</unit>
        <description>"FABRICATION_NUMBER"</description>
        <source-channel channel-name="MBUS_com0" parameter-id="51-0"/>
        <access-channel channel-name="Modbus_SMBUS" parameter-id="100"/>
 </parameter>
```

If, however, the ID is not recognized, it will be generated according to the appropriate algorithm - <PROTOCOL>_<MODEL>_<ID>_<OFFSET>

```
<parameter type="int32">
        <!--Actual value: 0 (Integer)-->
        <id>"MBUS_06605251_51_12"</id>
        <scale>10e7</scale>
        <comment>"INST_VAL"</comment>
        <unit>"J"</unit>
        <description>"ENERGY"</description>
        <source-channel channel-name="MBUS_com0" parameter-id="51-12"/>

        <access-channel channel-name="Modbus_SMBUS" parameter-id="124"/>
  </parameter>
```

SCALE

The scale parameter is collected from the device by default and written for basic units ([J][W][m^3], etc.). Value scaling is carried out in the following manner.

```
<scale>10e7</scale>
```

Signifies: VALUE * 1*10^7

The value can be scaled however, by changing the parameter:

```
<scale>10e-2</scale>
```

Then, if the value was expressed in the unit [J], it will be presented in GJ after scaling.

COMMENT

```
<comment>"INST_VAL"</comment>
```

This parameter defines whether it is a parameter with a value specifying a maximum or minimum type.

UNIT

An element specifying the unit. After scaling, the unit label is to be changed manually.

DESCRIPTION

Denotes a description of the value read from the meter. It can occur that the meter will have several parameters with the same description. In that case, compare read values during scanning.

```
<!--Actual value: 4960 (Integer)-->
```

They can be compared to the values on the meter display. On this basis, it can be determined which value is the desired value.

Parameter: Source & access

```
<source-channel channel-name="MBUS_com0" parameter-id="94-7"/>
```

The above parameter description signifies that, using the MBUS_com0 channel, the value from the device with ID = 94, parameter 7 will be read.

```
<access-channel channel-name="Modbus_S1" parameter-id="4521"/>
```

The read value will be accessible using the MODBUS protocol (defined in the MODBUS_S1 channel) with the address: 4521.

Warning

In the case of very complicated meters, the number of parameters may exceed the iMod's output. In the version that is currently available, this error is not detected, which is why the user is obliged to test the operation of expected functions after merging the MBusScan.xml configuration with the MainConfig.xml file.

Meter information on WWW

Mbus meter information may be displayed on WWW.

The tabs responsible for displaying this information are as follows:
licznik1.php and licznik2.php available in the device directory mnt\ramdisk\htdocs.

WWW Mechanism:
A list of parameters is created in the PHP language on the WWW page. For meter No. 1, the entry appears as follows:

```
$sql_list = list_variable(4,array(4619,4505,4503,4507,4501,4622));
```

For meter No. 2:

```
    $sql_list = list_variable(7,array(4619, 4525,4521));
```

The numbers given are ID values from the DATA table. This table is included in the MODBUS.DB database also found in the RAMDISK directory.

Based on these values, the following table is created:

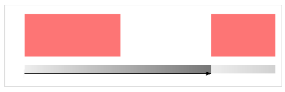
```
<table>
  <tr>
     <th>L</th>
     <th>Parametr</th>
     <th>Wartość</th>
     <th>Jednostka/Opis</th>
  </tr>
```

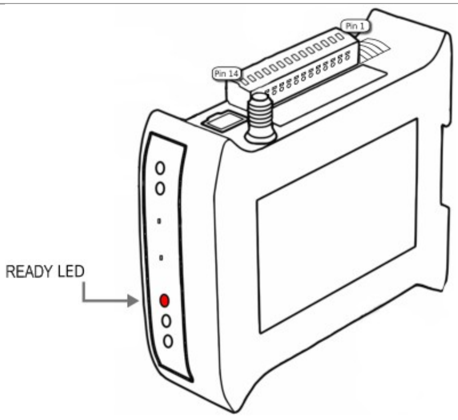
Which is then displayed with values read from the database.
For the correct values to appear on the page, the proper <ID> must be assigned to them

8.4. Software manager

4. Verifying installation

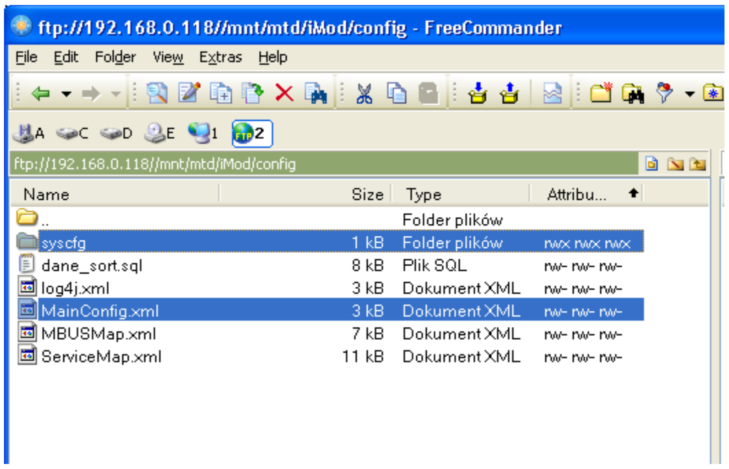
In order to make certain that the iMod platform is functioning correctly, observe the Status LED [RDY], which should flash with a frequency of about 1 Hz.

	Slow flashing (about 1Hz)



The platform configuration file will appear in the directory /mnt/mtd/iMod/config:
/mnt/mtd/iMod/config/MainConfig.xml

as well as a symbolic link to the NPE system configuration file
/mnt/mtd/iMod/config/syscfg


ftp://192.168.0.118//mnt/mtd/iMod/config - FreeCommander

ftp://192.168.0.118//mnt/mtd/iMod/config

Name	Size	Type	Attribu...
..		Folder plików	
syscfg	1 kB	Folder plików	rwx rwx rwx
dane_sort.sql	8 kB	Plik SQL	rw- rw- rw-
log4j.xml	3 kB	Dokument XML	rw- rw- rw-
MainConfig.xml	3 kB	Dokument XML	rw- rw- rw-
MBUSMap.xml	7 kB	Dokument XML	rw- rw- rw-
ServiceMap.xml	11 kB	Dokument XML	rw- rw- rw-

If you want to verify the iMod and a default configuration is set, read the chapter *Verification of Example*

8.5. Checking iMod platform version

Checking the iMod version is carried out using the *getenv* command:

```
$ getenv IMOD_VERSION
IMOD_VERSION=1009231539 installdir /mnt/sd/iMod xmlversion=1.0.1
```

8.6. Managing users (telnet)

User and *root* (super user) accounts are available by default in the system. If the device is to be used by many persons, it is worth adding additional accounts and groups. Thanks to this, the system administrator will be able to grant specific authorization so as to limit access by unauthorized users to selected files and directories.

Management of accounts in the NPE system is made possible by the following packs:

- adduser – add new user
- addgroup – add new user group
- saveusers – saves users and groups in the directory /mnt/mtd

A detailed description of using individual packs can be found in the table below.

Name	**Options**	**Description**
adduser *[options] [name]*	- h <*directory*> - g <*description*> - s <*layer name*> - G <*group name*> - D - H	Use "home" directory Optional account description Select shell layer Assign user to group Account without passwords Account without home directory
addgroup *[options] [name]*	- g	Group ID number
saveusers		Save users

Always remember to save user accounts after every change using the *saveusers* command. If not, after a system restart, the created accounts will be lost.

8.7. Cron and stability scripts

The iMod platform has two integrated watchdogs – safety mechanisms.

The Linux system gives access to a mechanism of scheduled script execution. Configuration is carried out using the 'crontab' file in the /mnt/mtd directory.

The default configuration includes calling up of two scripts:

1. iMod watchdog

```
*/2  * * * *          /mnt/sd/iMod/respawn
```

Every two minutes, iMod operation is checked. This is a guarantee of the platform's availability. It is possible to change the frequency of verification of the condition.

2. Watchspace

```
5 * * * *          /mnt/sd/iMod/watch_space
```

Every five minutes a script checking the amount of available space on the DATA partition is run. If there is too little space, old .log files will be deleted. If there is still too little space (less than 15%), the script will begin deletion starting from the oldest CSV files.

3. Cron and custom scripts

It is possible to expand the cron table with your own scripts.

In order to do this, add an entry according to the appropriate principles. The cron table consists of six columns.

- 1st column (range *0-59*) denotes minutes.
- 2nd column (range *0-23*) denotes the hour.
- 3rd column (range *0-31*) denotes the day of the month.
- 4th column (range *0-12*) denotes the month. (0 and 1 are January)
- 5th column (range *0-7*) denotes the day of the week (0 and 7 are Sunday)
- 6th column specifies the command to be executed for a given row.

More examples of CRONTAB application can be found at:

http://pl.docs.pld-linux.org/uslugi_cron.html in the chapter **System Tables.**

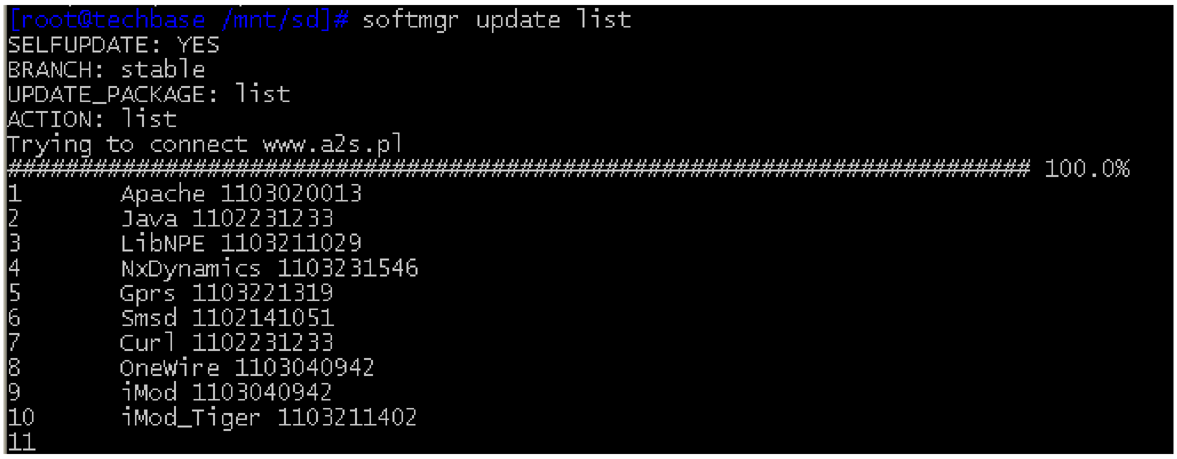
8.8. Pack manager (softmgr)

The pack manager is installed with the libraries for managing hardware resources (LIBNPE).

In order to use it, the MAC address must be authorized in the manufacturer's server. If the device is not authorized to download updates, technical support should be contacted.

An active device internet connection is necessary for using the pack manager.

The following packs are available in the pack manager:



```
[root@techbase /mnt/sd]# softmgr update list
SELFUPDATE: YES
BRANCH: stable
UPDATE_PACKAGE: list
ACTION: list
Trying to connect www.a2s.pl
######################################################################## 100.0%
1       Apache 1103020013
2       Java 1102231233
3       LibNPE 1103211029
4       NxDynamics 1103231546
5       Gprs 1103221319
6       Smsd 1102141051
7       Curl 1102231233
8       OneWire 1103040942
9       iMod 1103040942
10      iMod_Tiger 1103211402
11
```

Using the manager:

USAGE: softmgr {update [apache|java|nx|libnpe|gprs|sms|curl|imod|onewire] | list}

E.g. updating the imod pack:

# softmgr update imod

APPENDIX A (useful shell commands)

cd	Change directory to the one given
pwd	Displays the current path
ls -l	Lists files
mv <plik> <plik>	Moves file
rm <plik>	Deletes file
chmod +x <plik>	Grants execution authorization
Sdon/sdoff	Mount/Dismount SD card
e2fsck	SD card repair program
Uptime	System load
ps	Running processes
GETENV <PACKAGE_NAME>_VERSION	Checks version of installed packs
ifconfig	Checks network settings
udhcp	Refreshes dhcp
tail -f <plik>	Displays live-log for the given file
mc	Midnight-commander
mcedit <plik>	File editor
vi	File editor
settime	Sets time on the device
npe	ANSI C hardware resource support
npe ts	Writes the set time to the device
imod start	Platform start
imod debug	Platform start in debug mode
imod trace	Platform start in trace mode

Appendix B (troubleshooting)

The iMod is for integrating systems. The nature of these matters requires an analytical approach to resolving problems. In understanding of the needs of integration systems, this appendix has been prepared and should help in diagnosing problems that may occur during integration of systems using the iMod device.

Linux system

Messages

A log file from the Linux system itself can be found in the device. It is in the directory /mnt/mtd/messages. It can be viewed in real time using the command:

$ tail -f /mnt/mtd/messages

This log contains the status of the GPRS connection if the Autoreconnect service has been activated. The iMod watch-dog also adds entries on whether the iMod engine is running.

Uptime

After configuring the iMod and waiting 30 minutes, system load should be checked. This is done using the command:

$uptime

As a result, three number values are obtained, which denote the system load level. Strict attention should be paid so that the second and third values do not exceed 1.0.

Kernel

For the system to operate in a stable fashion, it is necessary to have the newest stable system kernel. Every so often, an optimized system containing numerous fixes and updates is issued. It is worth observing the information given to partners of TechBase Solution Partner.

GPRS

Making and maintaining a stable GPRS connection to the iMod platform is relatively simple. It does however, require experience regarding the organization of such connections.

The error met with most often is:

PPP interface not established.

Script exit code: 16

This signifies incorrect login data sent to the APN.

The Device is not visible in the network

Check if the LEDs on the RJ-45 terminal are signaling correct operation.

Use the NPE searcher application (available at a2s.pl – NPE product catalog card).

If the device is still not detected, check if a device with the same MAC address is registered to the router.

In the case of further problems with finding the device in a network, contact technical support.

iMod

iMod log level

The iMod platform has several levels of writing logs. The first of these is the default INFO level. However, there are two deeper levels enabling more precise entries to be activated. These are the commands:

imod debug

and

imod trace

In order to view what is happening with the device in real-time, use the command:

tail -f /mnt/data/logs/iMod.log

iMod.log

Information on the operation of the iMod platform can be found in the file iMod.log. It contains occurrences and executions of appropriate events.

This file can be downloaded to a computer for analysis.

Device communication error

If there is a problem with communication with the device, check:

- system version (uname -a, getenv IMOD_VERSION)
- system load (uptime)
- message log file (for checking the operation of the GPRS connection)
- iMod.log file.

If the iMod.log file contains entries regarding a reading failure, go to the stderr.log file.

This file contains information regarding sent queries (HEX dump) and returned responses. For information to be completed in the stderr.log file, the iMod must be started in a mode of at least DEBUG.

First, it should be verified if the connection error does not result from a lack of communication between HOST – iMod or iMod – Slave.

SMS sent

To send a text message, you must have a correctly configured internet connection. In addition, the number in the MainConfig.xml file should have 11 characters, with the first two being the country code. If not, the sms may not be sent. To check if the connection works, enter the command:

$ifconfig

If the connection is made, the PPP0 interface will be visible in the network interfaces.

In the case where text messages are still not being sent, a text message should be sent from the console using the *smssend* command.

If the text message is still not sent, the following command should be executed:

gprs modem_info_refresh

Restart the device and try to send the text message again.

Factory settings reset does not help

A factory settings reset causes a clean LINUX system with no additional services to start with default configuration. The previous configuration is saved in the files:

syscfg.bak and *rcs.bak*.

For the iMod to start again, these files must be restored.

If, after their restoration, the device still does not start, the error lies in the start-up files and technical support must be contacted for further assistance.

Appendix C (1-wire module)

Integration of 1-Wire bus with the iMod platform

This chapter has the purpose of presenting a method of using the iMod to collect data from a scattered system of sensors and executive systems connected to the 1-Wire bus.

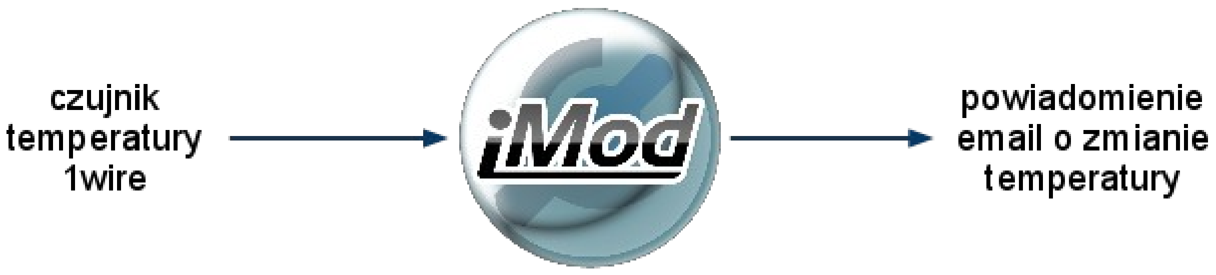
The entire material has been separated into two parts, in which first, the first steps of servicing the 1-Wire protocol in the iMod system will be discussed. The second part of the article will be dedicated to an expanded configuration supporting two (or more) NPE devices, which will be a slightly modified configuration from the first part.

In order to fully understand all the terms used in this description, we recommend becoming acquainted with basic information as to what the iMod platform is and how it works.

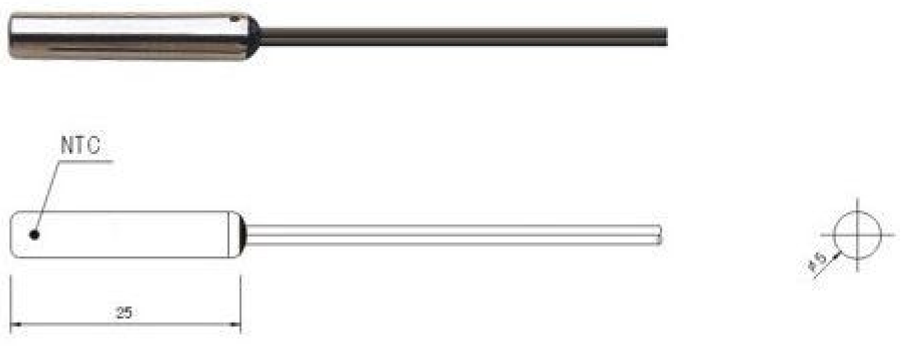
The article has the purpose of presenting the simplicity and scaling capacity of 1-Wire bus support in the iMod system used in a scattered measuring and control network comprised of NPE devices.

1. First step – simple configuration

The first example of using the 1- Wire bus will use a single digital temperature sensor connected to an NPE device. Thanks to this, the user will have a chance to become acquainted with the method of defining parameters reflecting the connected sensors.

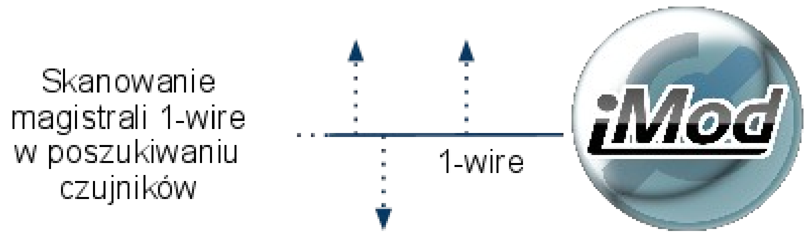

czujnik temperatury 1wire
iMod
powiadomienie email o zmianie temperatury

This example uses a Dallas DS18B20 sensor, which reads temperatures in range from -55 ^{0}C to +125 ^{0}C with a programmable resolution of 9-12 bits.
The tested sensor has been placed in magnesium housing and connected using a 2 meter 3 conductor cable.



Knowledge of unique numbers identifying individual sensors assigned during the production process will be be indispensable for creating correct parameter definitions.

In order to facilitate detection of individual sensors and the addresses assigned to them – a mechanism searching the bus for connected and active 1-Wire interface systems has been created.

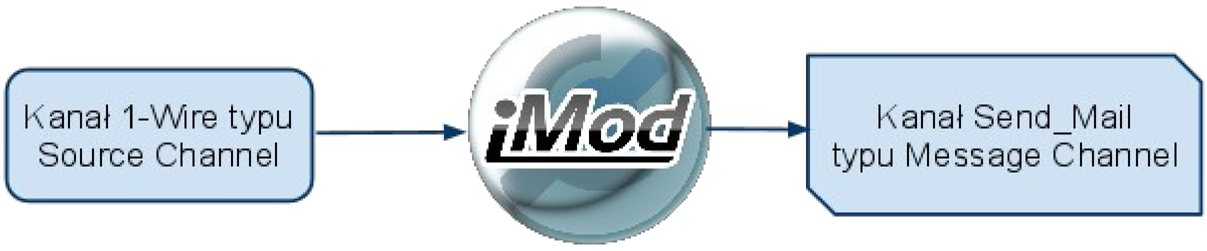


Starting the ***imod*** application in the system console with the ***scan onewire*** parameters will initiate the bus search process. In the case of finding and detection of sensors, a template configuration file will be generated and will serve as the first discussed elementary configuration. The XML file containing parameters will be saved in the iMod application directory.

```
<parameter>
      <id>"THERM_1"</id>
      <description>"THERMOMETER_DS18B20"</description>
      <source-channel channel-name="OneWire" parameter-
id="28A935CA020000:temperature12"/>
</parameter>
```

Due to the automatic mechanizm for generating parameters for one sensor, even several dozen separate definitions can be created for one sensor, which is why after scanning, the desired parameters should be selected, and on the basis of those parameter, the target configuration file fulfilling the requirements of a given project should be created.

A scheme illustrating the configuration fo the iMod platform for this example has been presented below:


Kanał 1-Wire typu Source Channel
iMod
Kanał Send_Mail typu Message Channel

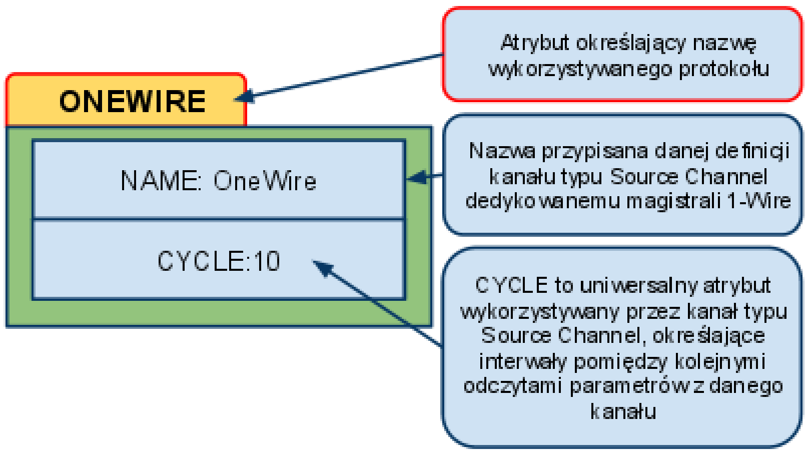
Two transmission channels have been used:

- source channel for data collected from the 1-wire bus
- Message channel for transmission of e-mail notifications

Configuration parameters for individual channels are discussed in detail below.

A. Defining 1-Wire source channel

Sensors connected to the 1-Wire bus (due to the access method) are detected in the iMod system as parameters connected with the source channel. Defining a source channel responsible for communication with the bus requires the specification of group of necessary attributes, which have been presented on the scheme.


ONEWIRE
NAME: OneWire
CYCLE:10
Atrybut określający nazwę wykorzystywanego protokołu
Nazwa przypisana danej definicji kanału typu Source Channel dedykowanemu magistrali 1-Wire
CYCLE to uniwersalny atrybut wykorzystywany przez kanał typu Source Channel, określające interwały pomiędzy kolejnymi odczytami parametrów z danego kanału

A graphical representation of the source channel structure in the XML configuration file has also been presented on the listing below for comparison.

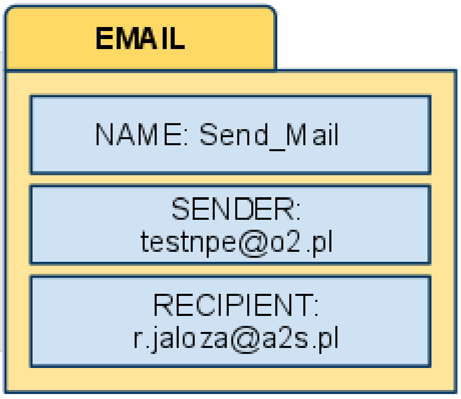
```
<source-channel name="OneWire">
            <protocol name="ONEWIRE"/>
    <gap>0</gap>
    <cycle>10</cycle>
 </source-channel>
```

The simplest configuration utilizes default parameters assigned to the 1-Wire channel. Thanks to this, the ONEWIRE channel can be defined using several lines. The method for overwriting default values will be presented with creation of an expanded configuration.

B. Message channel definition

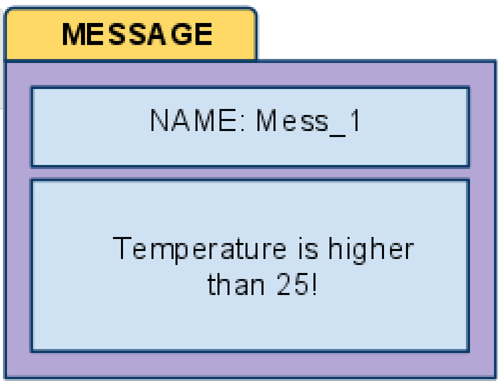
In order to show an example of interaction between parameters, the channel responsible for sending e-mail messages will be defined. An example of a method for defining a message channel has been shown below.

```
<message-channel name="Send_Mail">

<protocol name="EMAIL">
<property name="user" value="testnpe"/>
<property name="password" value="123npe"/>
</protocol>
<port>"poczta.o2.pl"</port>
<recipient>"r.jaloza@a2s.pl"</recipient>

</message-channel>
```

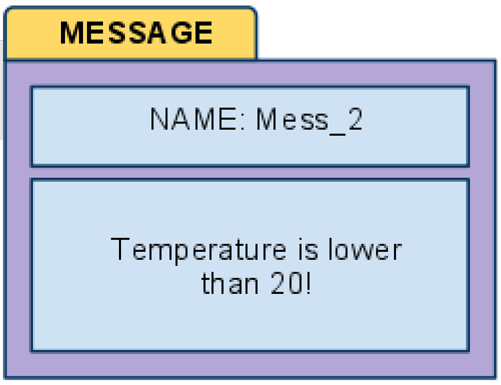


For the needs of the message channel, two parameters, being the definition of the message sent to a specified addressee, will be added. The method for their implementation has been presented in the graphical representation below.

```
<message id="Mess_1">
<![CDATA["Temperature is higher than 25!"]]>
</message>
```



```
<message id="Mess_2">
<![CDATA["Temperature is lower than 20!"]]>
</message>
```



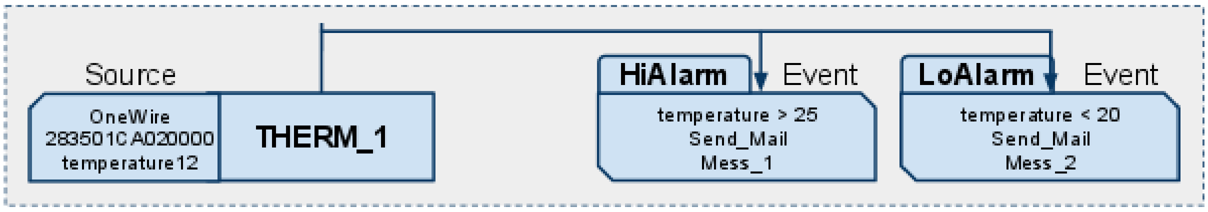
C. How to tie sensor resources to the iMod application?

Sensors connected to the bus are detected using a unique identifier assigned during the production process. In the iMod system, the parameters of 1-Wire sensors are also discerned using a unique sensor ID number (in our case *283501CA020000*) and resolution of temperatur measurement (sampling) (temperature12).

These attributes are necessary for correct detection of the type of system connected to the bus and the resource to be accessed by the iMod system. Each family of sensors possesses its own attributes assigned to specific internal resources.

In order to demonstrate the interaction and reaction of the system to a change of the resource – in our case, temperature change, two events have been assigned to the parameter, which will be called up after exceeding specific temperature thresholds. In effect, this will cause the sending of the first message as an e-mail.

The graphical representation of the parameter has been given below:



A fragment of the configuration file defining the event connected with the iMod application parameter giving access to temperature sensor resources appears as follows:

```
 <event type="HiAlarm">
           <message-channel channel-name="Send_Mail"/>
            <message-id>"Mess_1"</message-id>
            <property name="trigger" value="25.0"/>
</event>
```

HiAlarm and LoAlarm type events are identically defined – the only difference is in the interpretation of the trigger attribute.

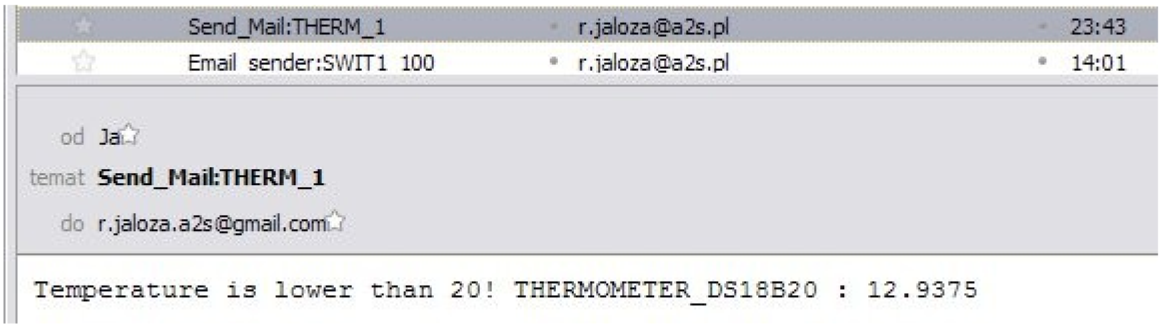
D. Example of operation

The configuration file created from parameters and definitions of the channels described above can be copied directly to the iMod application directory and run. The next part will present examples of the effects of operation of a configuration configured in such a way.

The temperature read from the sensor was less than 20 degrees, which was detected by the iMod system, and according to the settings, an e-mail was sent. The application log recorded the following information regarding this event:

```
23:43:51,552 | INFO  | OneWire | Added to Send_Mail's queue event count: 1 id: THERM_1
23:43:59,131 | INFO  | Send_Mail | Email sent
23:43:59,137 | INFO  | Send_Mail | Transmitted Send_Mail's queue event id: THERM_1 (queue size: 0)
```

The received message has been shown below:



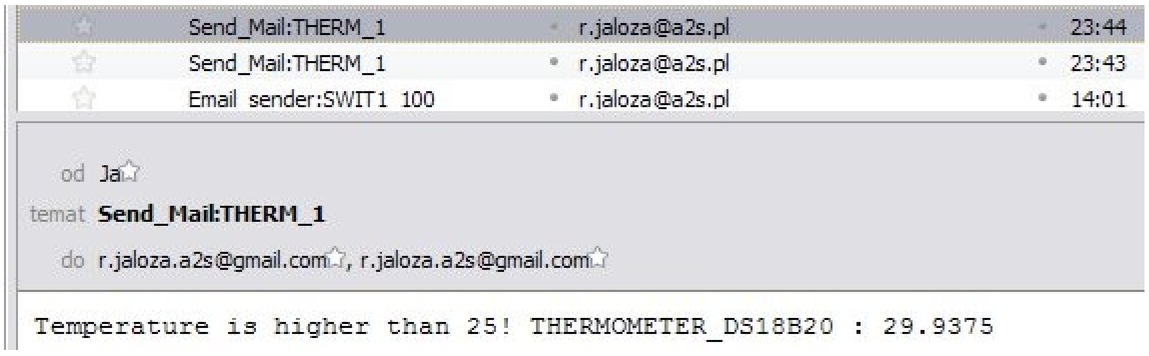
```
Send_Mail:THERM_1        r.jaloza@a2s.pl    23:43
Email sender:SWIT1 100   r.jaloza@a2s.pl    14:01

od Ja
temat Send_Mail:THERM_1
do r.jaloza.a2s@gmail.com

Temperature is lower than 20! THERMOMETER_DS18B20 : 12.9375
```

An analogous situation occurred when the ambient temperature of the sensor was increased to above 25^{0}C.

An entry appeared in the application log:

```
23:44:51,762 | INFO  | OneWire | Added to Send_Mail's queue event count: 1 id: THERM_1
23:44:52,216 | INFO  | Send_Mail | Email sent
23:44:52,222 | INFO  | Send_Mail | Transmitted Send_Mail's queue event id: THERM_1 (queue size: 0)
```

The message received in the inbox is as follows:



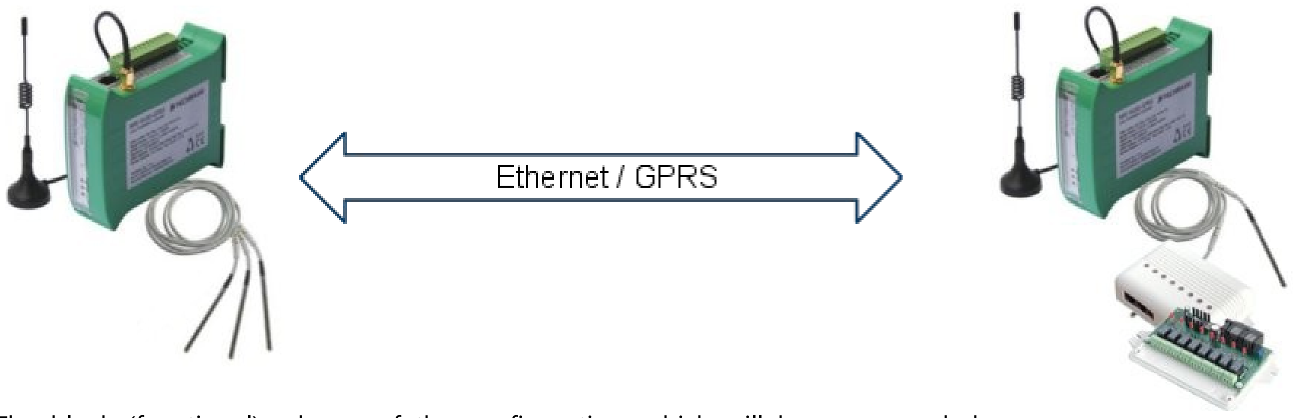
```
Send_Mail:THERM_1        r.jaloza@a2s.pl    23:44
Send_Mail:THERM_1        r.jaloza@a2s.pl    23:43
Email sender:SWIT1 100   r.jaloza@a2s.pl    14:01

od Ja
temat Send_Mail:THERM_1
do r.jaloza.a2s@gmail.com, r.jaloza.a2s@gmail.com

Temperature is higher than 25! THERMOMETER_DS18B20 : 29.9375
```

The above examples have shown the functioning of a simple configuration using a single temperature sensor connected to the NPE device with the iMod application.

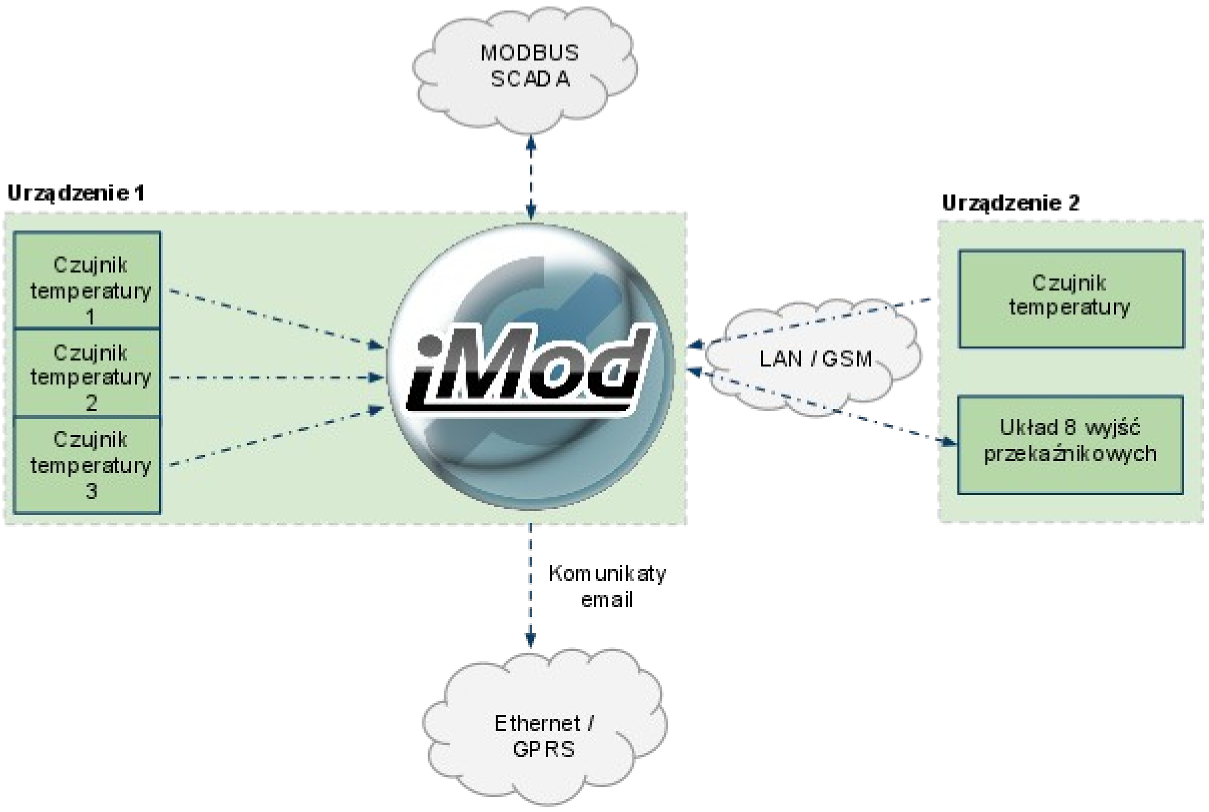
2. Step two – Expanded configuration

The first simple configuration presented the basic functioning which can be easily used and expanded by additional parameters representing sensors connected to the device with the iMod application as well as to other devices available in the network.

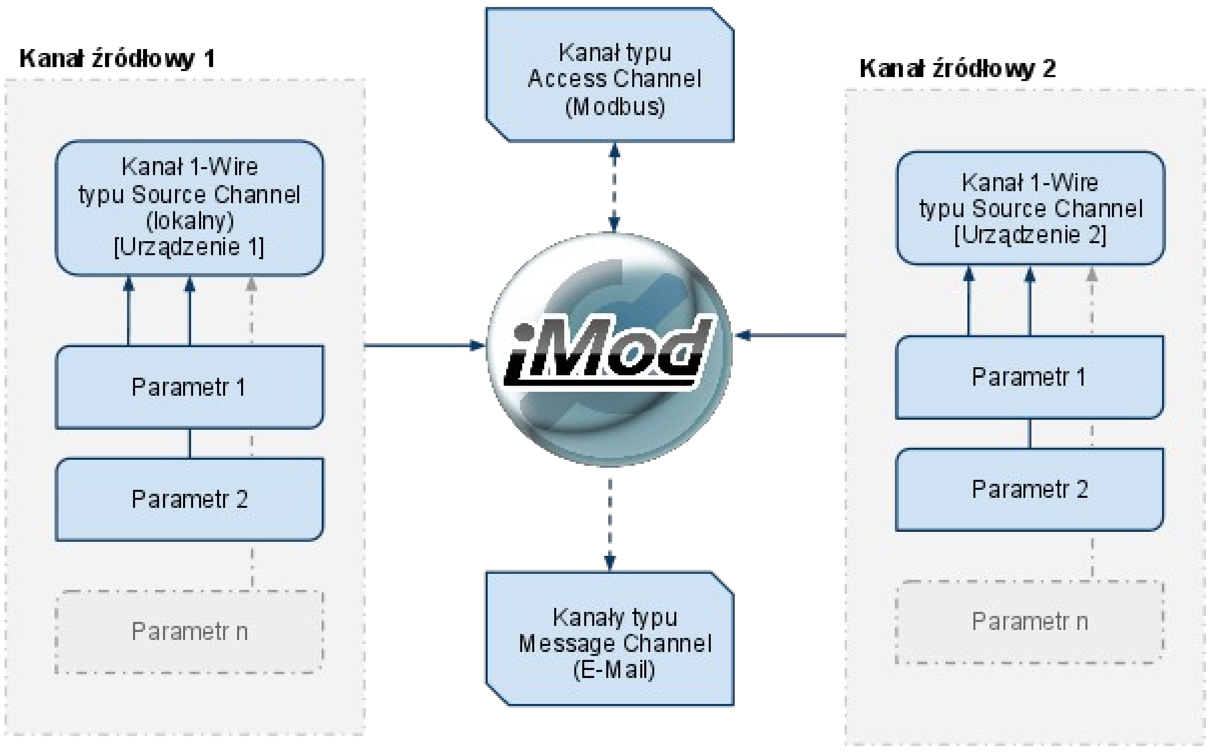
The topology used in this second example has been presented on the figure below. The scattered system is comprised of three temperature sensors connected to a device without the iMod application as well as a system of transmitter outputs along with a single temperature sensor connected to a second device with an installed iMod application.



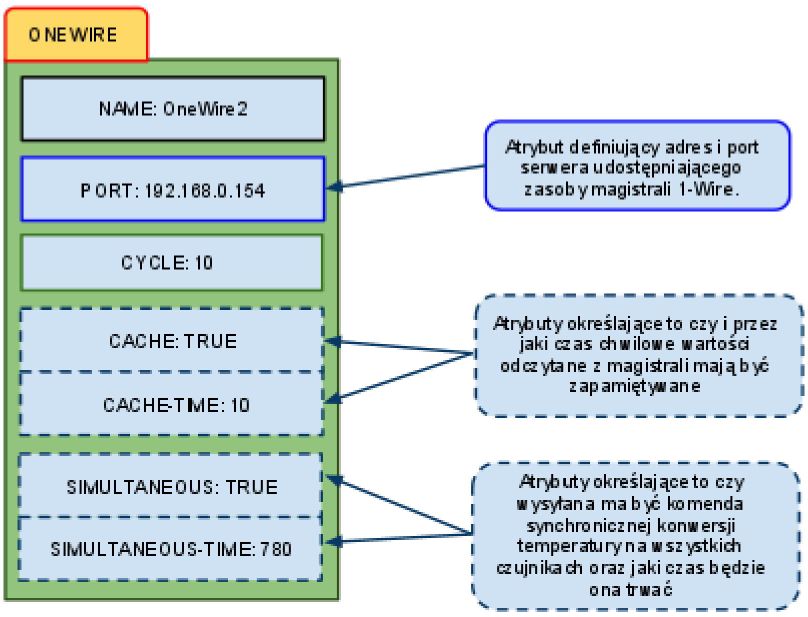
The block (functional) scheme of the configuration, which will be an expanded example of the one from the first part – is shown below:



The functional scheme of the constructed configuration is shown below:


Kanał źródłowy 1
Kanał 1-Wire typu Source Channel (lokalny) [Urządzenie 1]
Parametr 1
Parametr 2
Parametr n
Kanał typu Access Channel (Modbus)
iMod
Kanały typu Message Channel (E-Mail)
Kanał źródłowy 2
Kanał 1-Wire typu Source Channel [Urządzenie 2]
Parametr 1
Parametr 2
Parametr n

Giving access to resources connected to another device requires adding another ***Source channel*** , which will communicate with sensors connected to the second NPE device available in the LAN network.


ONEWIRE
NAME: OneWire2
PORT: 192.168.0.154
CYCLE: 10
CACHE: TRUE
CACHE-TIME: 10
SIMULTANEOUS: TRUE
SIMULTANEOUS-TIME: 780
Atrybut definiujący adres i port serwera udostępniającego zasoby magistrali 1-Wire.
Atrybuty określające to czy i przez jaki czas chwilowe wartości odczytane z magistrali mają być zapamiętywane
Atrybuty określające to czy wysyłana ma być komenda synchronicznej konwersji temperatury na wszystkich czujnikach oraz jaki czas będzie ona trwać

Optional attributes have been designated by a broken line. If they are skipped, the iMod systemwill change them with default values defined for a specific parameter.

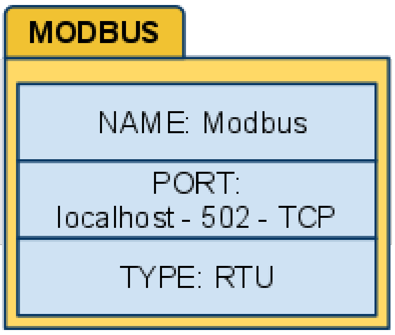
The structure of the added channel in the XML file appears as follows:

```
<source-channel name="OneWire2">
        <protocol name="ONEWIRE"/>
        <port>"ET-192.168.2.154"</port>
                <property name="cache" value="true"/>
                <property name="cache-time" value="10"/>
                <property name="simultaneous" value="true"/>
                <property name="simultaneous-time" value="780"/>
        <cycle>10</cycle>
</source-channel>
```

One of the assumptions of the second example is giving access to parameters connected with the iMod application for SCADA type applications communicating using the Modbus protocol. Due to the fact, that queries incoming from the SCADA application are asynchronous to the configuration, an ***Access channel*** giving access to selected parameters has been added.

An example of the defined channel is shown below::

```
<access-channel name="Modbus">
<protocol name="MODBUS">
<property name="type" value="rtu"/>
</protocol>
<port>"ET-502-TCP"</port>
</access-channel>
```

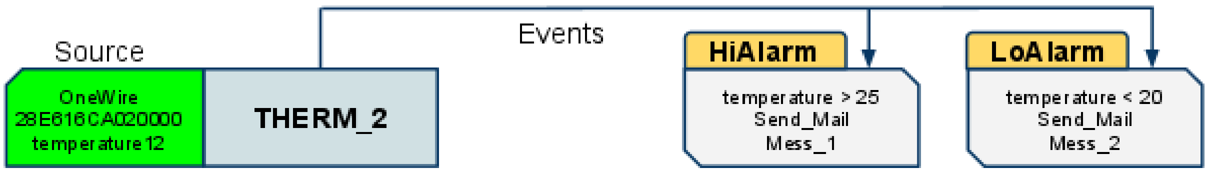


When a specific parameter is tied to the Modbus channel, it will be available for other applications on TCP/IP port number 502 under a specific ID.

A. How to expand existing parameters?

An expanded configuration requires modification of the existing configuration and addition of additional parameters. Connection of the resources of sensors connected to another device in the network is as simple as in the first example. New parameters are to be assigned to the defined Source Channel.

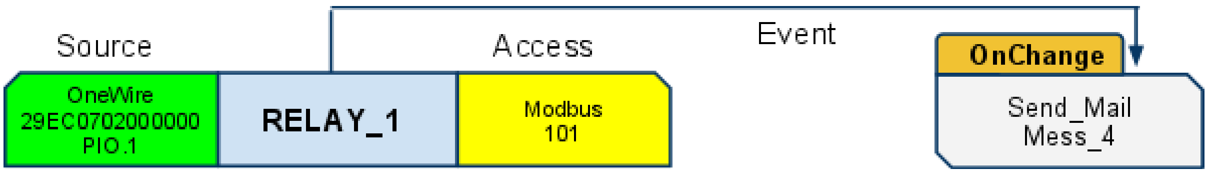
Below, an exemplary parameter responsible for temperature readings of the sensor connected to another device in the network is shown with the events assigned to it.



The definition of an exemplary parameter written in the XML file appears as follows:

```
<parameter>
                    <id>"THERM_2"</id>
                    <description>"THERMOMETER_DS18B20"</description>
                    <source-channel channel-name="OneWire2"
                                                    parameter-
id="28E616CA020000:temperature12"/>

                    <event type="HiAlarm">
                  <message-channel channel-name="Send_Mail"/>
                  <message-id>"Mess_1"</message-id>
                    <property name="trigger" value="25.0"/>
                    </event>

<event type="LoAlarm">
                  <message-channel channel-name="Send_Mail"/>
                  <message-id>"Mess_2"</message-id>
                    <property name="trigger" value="20.0"/>
                    </event>
</parameter>
```

One of the basic properties of the transmitter I/O module is the capability of asynchronous setting of port states. In order to take advantage of the module's potential, a unique identifier in the Modbus network connected with the Access Channel has been assigned to parameters reflecting individual transmitter outputs.



A fragment of the implementation has been shown below:

```
<parameter>
                         <id>"RELAY_1"</id>
                         <description>"SWITCH_DS2408"</description>
                         <source-channel channel-name="OneWire2"
                                                     parameter-id="29EC0702000000:PIO.1"/>
                         <access-channel channel-name="Modbus" parameter-id="101" />
                         <event type="OnChange">
                    <message-channel  channel-name="Send_Mail"/>
                    <message-id>"Mess_4"</message-id>
                      <property name="trigger" value="0.0"/>
                         </event>
</parameter>
```

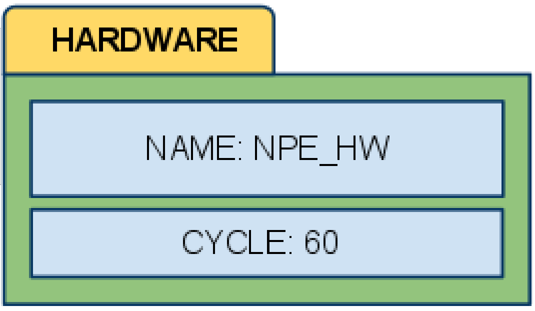
Analogously to previous examples, in the case of a change of state of a specified output, an e-mail will be sent.

3. Step three – advanced interactions between parameters

Finally, capabilities of creating rules of interaction between parameters will be presented. Thanks to this, for example, a specified sequence of output ports can be set in the case of exceeding a specified temperature value.

In the last configuration, NPE device resources (digital outputs and LEDs) will also be used. An example of a source channel definition has been shown below:

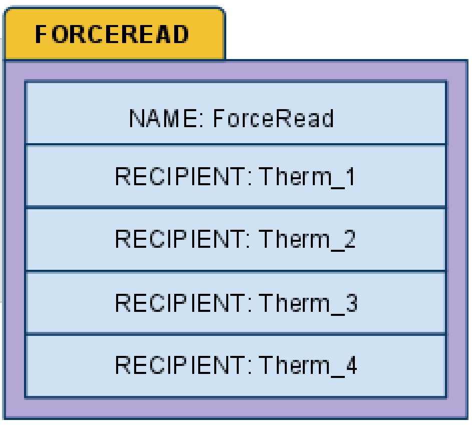
```
<source-channel name="NPE_HW">
<protocol name="HARDWARE"/>
<cycle>5</cycle>
</source-channel>
```



A message channel is usually used to send messages in specific situations. Another application of these channels can be forcing reading or writing of a specific value to a given parameter.

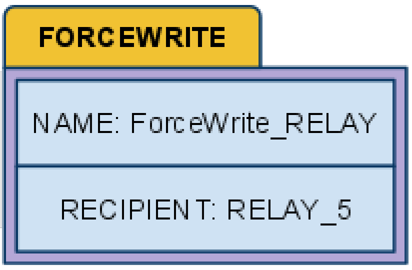
- Forcing reading of selected parameters is realized using FORCEREAD. The parameters to be read are added as *recipient* attributes.

```
<message-channel name="ForceRead">
<protocol name="FORCEREAD"/>
<recipient>"THERM_1"</recipient>
<recipient>"THERM_2"</recipient>
<recipient>"THERM_3"</recipient>
<recipient>"THERM_4"</recipient>
</message-channel>
```



- Forcing writing of a specific value to a selected parameter is realized using FORCEWRITE.

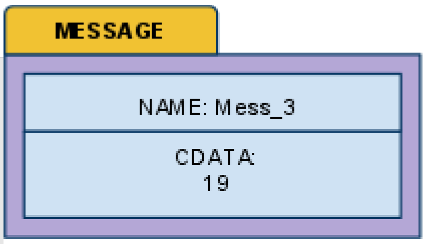
```
<message-channel name="ForceWrite_RELAY">
<protocol name="FORCEWRITE"/>
<recipient>"RELAY_5"</recipient>
</message-channel>
```



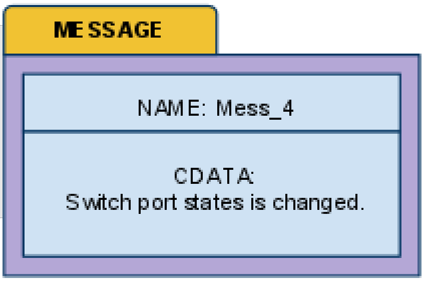
The universality of the ForceWrite solution makes it possible to tie the parameters to the hardware resources, set a specific output sequence and change a specific digital output or USER_LED.

For the needs of the forced actions described above, definitions of additional messages were added. The first of them defines a sequence of digital outputs of the transmitter module (as a bit representation of a number from 0-255), and the second informs of a change of a specified digital input on the transmitter.

```
<message id="Mess_3">

<![CDATA["19"]]>

</message>
```



```
<message id="Mess_4">
<![CDATA[
"Switch port states is changed. "
]]>
</message>
```



For example, in the final configuration, parameters were assigned the following functions:

●***RELAY_1*** – Parameter related to port PIO.1 of the transmitter. Change in the parameter state (*PIO.1* output) causes a change in the USER_LED state. The set value is identical to the state of the port, e.g. change in the state of port *PIO.1* from 0 to 1 will cause the USER_LED to turn on.

●***RELAY_2*** - Parameter related to port PIO.2 of the transmitter. Change in the parameter state (*PIO.2 state*) causes a specific output port sequence of the transmitter output system to be set, as defined in the content of message *Mess_3*.

●***RELAY_3*** – Change of the parameter state(*PIO.3* output) causes a change in the state of digital output *DO1* of the NPE device. The set value is identical to the change of the state of port PIO.3, e.g. change in the state of port *PIO.3* from 0 to 1 will cause the setting of an analogous state on output *DO1.*

●***RELAY_4* –** Change of the parameter state (*PIO.4* output) will cause forced reading of parameters assigned to a ForceRead type message channel. In effect, the current temperature value will be read from sensors defined as *THERM_1, THERM_2, THERM_3, THERM_4.*

●***RELAY_5* –** Change of the parameter state (*any output)* will cause an e-mail to be sent using the *Send_mail* channel.

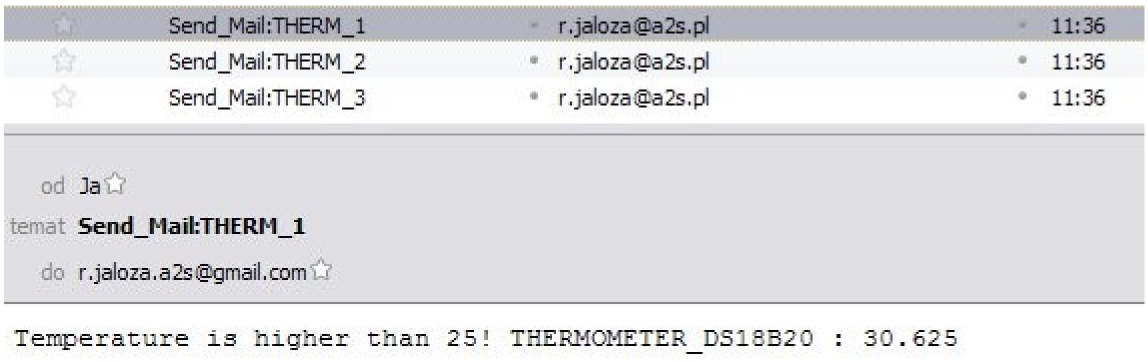
A. Example of functioning

Examples and interactions between defined parameters have been shown below:

- Raising sensor temperatures *THERM_1, THERM_2 i THERM_3* above the 25^{O}C threshold New etnries have appeared in the log:

```
11:36:14,203 | INFO | OneWire | Added to Send_Mail's queue event count: 1 id: THERM_1
11:36:14,624 | INFO | OneWire | Added to Send_Mail's queue event count: 2 id: THERM_2
11:36:15,624 | INFO | OneWire | Added to Send_Mail's queue event count: 3 id: THERM_3
```

- Sent messages have appeared in the inbox:



Send_Mail:THERM_1 r.jaloza@a2s.pl 11:36
Send_Mail:THERM_2 r.jaloza@a2s.pl 11:36
Send_Mail:THERM_3 r.jaloza@a2s.pl 11:36

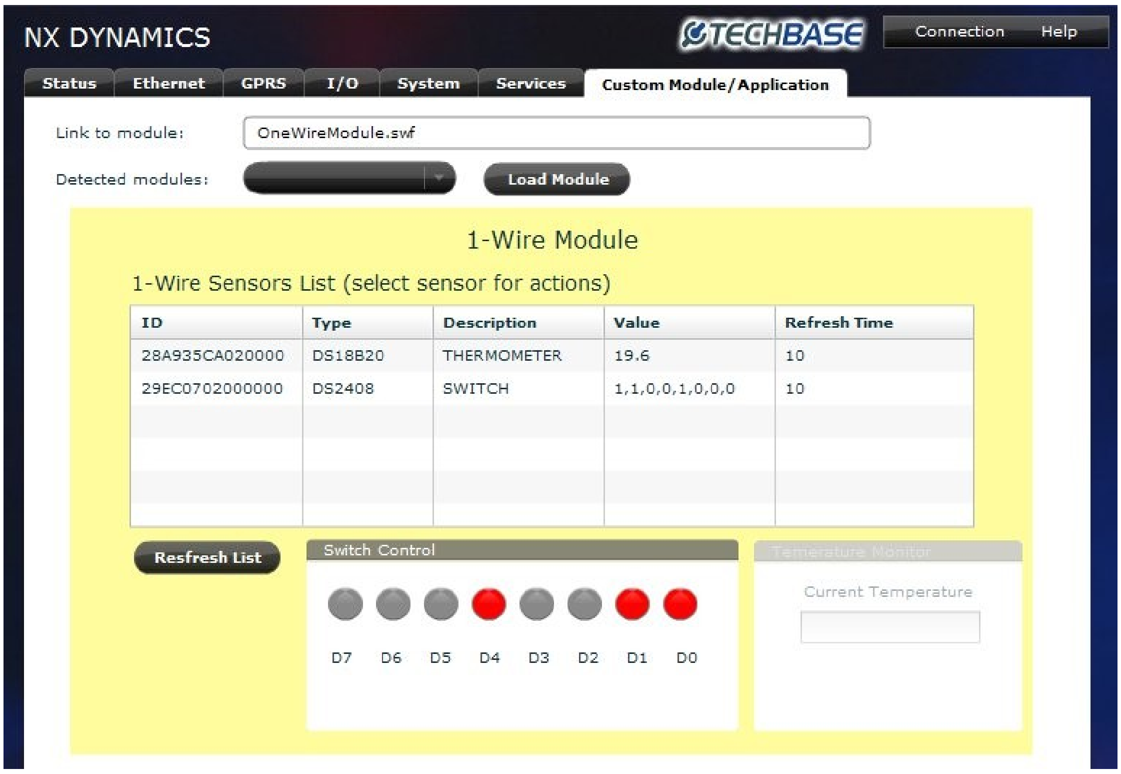
od Ja
temat **Send_Mail:THERM_1**
do r.jaloza.a2s@gmail.com

```
Temperature is higher than 25! THERMOMETER_DS18B20 : 30.625
```

- Lowering the temperature of sensor *TEMP_3* below 20^{O}C caused the effect:

```
22:33:47,428 | INFO  | OneWire | Added to ForceWrite_RELAY's queue event count: 1 id: THERM_3
22:33:52,632 | INFO  | ForceWrite_RELAY | Wrote 1-Wire register: RELAY_5 Value: 19
22:33:52,649 | INFO  | ForceWrite_RELAY | Transmitted ForceWrite_RELAY's queue event id: THERM_3 (queue size: 0)
```

The input sequence set on I/O ports of the transmitter output system is visible in the application NX DYNAMICS:



Link to module: OneWireModule.swf

Detected modules:

1-Wire Module

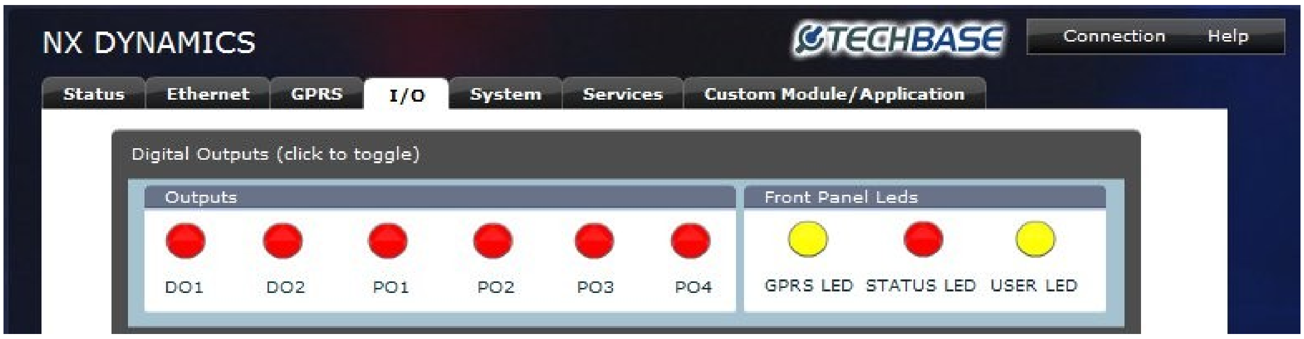
1-Wire Sensors List (select sensor for actions)

ID	Type	Description	Value	Refresh Time
28A935CA020000	DS18B20	THERMOMETER	19.6	10
29EC0702000000	DS2408	SWITCH	1,1,0,0,1,0,0,0	10

Change in the state of input RELAY_1 caused a change of the LED

```
22:33:46,386 | INFO  | OneWire2 | Added to ForceWrite_LED's queue event count: 1 id: RELAY_1
22:33:46,454 | INFO  | ForceWrite_LED | Transmitted ForceWrite_LED's queue event id: RELAY_1 (queue size: 0)
```

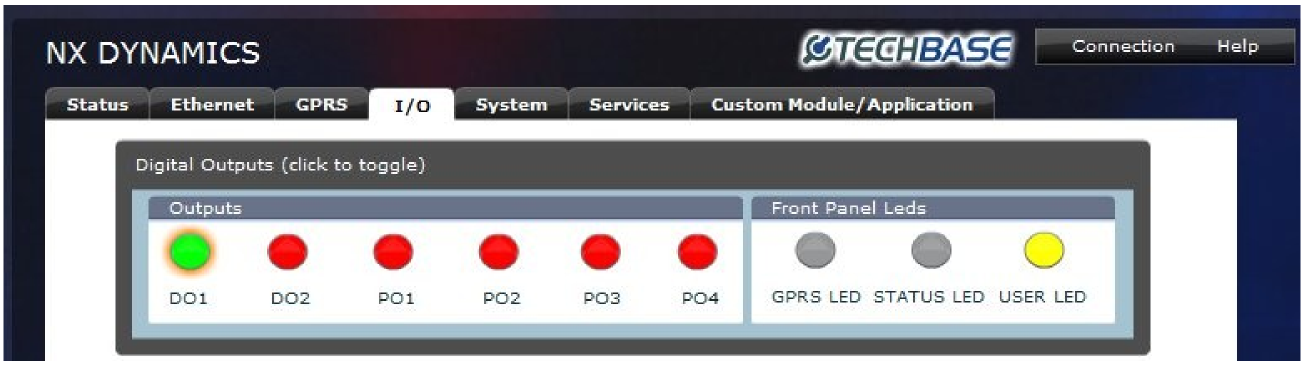
Change in the state of the USER_LED to active is also visible in the NX DYNAMICS application:



Change in the state of I/O port RELAY_3 caused a change in digital output DO1:

```
00:41:48,084 | INFO  | OneWire2 | Added to ForceWrite_DO1's queue event count: 1 id: RELAY_3
00:41:48,103 | INFO  | ForceWrite_DO1 | Transmitted ForceWrite_DO1's queue event id: RELAY_3 (queue size: 0)
```

NX DYNAMICS Application



4. Summary

The above examples of a simple yet expanded configuration have shown methods of usage and the capabilities of cooperation of the 1-Wire bus with the iMod application.

Use of a TCP/IP server application that gives access to the resources of the 1-Wire bus aids easy and efficient scaling of the system and capabilities of creating subnets consisting of many NPE devices with connected sensors and a single iMod application collecting measurement from all network branches.