# Telemetry Module MT-202

CE

## User Manual



**telemetria.pl**

**inVentia**

# Telemetry Module MT-202 User's Manual

GSM/GPRS Telemetry Module
for monitoring and control

Class 1 Telecommunications Terminal
Equipment for GSM 850/900/1800/1900

MT-202

# MT-202

## INDEX

# 1. Introduction

Despite the explosive expansion of mobile phone networks and the per definition digital nature of transmission utilized by them, these networks were not well-suited for the transfer of digital data streams until recently. The applied technologies used traditional modem protocols. This meant that the need to use circuit-switched mode for establishing connections and transmission capabilities was limited to point-to-point connections. Thus, we had to do with the typical analogue circuit-switched links technology used in traditional telephone systems adapted to the wireless digital transmission environment. A connection established for data transmission occupied the whole voice channel. As a result, simultaneous phone calls were impossible, and the cost of transmission depended on the connection time rather than the amount of the transferred data. In this situation, the real-time monitoring of units requiring constant supervision but generating a relatively small amount of data was not possible.

The situation changed radically after GSM operators introduced data transmission services in the GPRS (General Packet Radio Services) standard. The new technology, called the 2.5G standard - meaning Two and a Half Generation, is a bridge between the to-date technologies of the Second-Generation GSM networks (2G) and the Third-Generation technology (3G - UMTS) which has been waiting for application on a wide scale. However, broadband 3G technology is still very distant. So let us deal with the solutions available right now.

In providing this user's manual, we are aware that it will not answer all your questions and address all your doubts. This is why the manual will be regularly supplemented and modified. We ask for your comments and welcome suggestions in order to make this manual more useful.

*INVENTIA Ltd.*

# 2. Module's destination

MT-202 is a specialized telemetry module optimized for application in advanced measurement and alarm systems provided with a mains power supply.

General attributes of MT-202:

- Compact design
- Programmable logical functions using internal markers logical states, timers, counters, diagnostic flags and internal registers used for SMS, email and data sending, setting internal markers and registers
- Ability to extend network with local extension modules
- Spontaneous transmission of data on occurrence of pre-defined alert states enabling application on objects requiring continuous monitoring
- GPRS packets routing also in Transparent operating mode
- Access to internal resources by standard MODBUS RTU protocol
- Local Master or Slave functionality for devices connected to serial communication port (MODBUS RTU, GazModem, NMEA 0183 protocols)
- External devices resources mapping
- Implementing of nonstandard serial protocols by user program in FlexSerial operating mode
- MT2MT buffer – direct data transmission between MT modules.

A typical application field for MT-202 are all installations requiring constant remote monitoring.

We encourage getting acquainted with the modules' configuration and modes of operation along with examples of application in different configurations described in appendices.

## 3. GSM requirements

For proper operation, the module needs a SIM card supplied by a GSM operator providing GPRS and/or SMS services.

The GPRS/enabled SIM card has to be registered in the APN with static IP addressing. The unique IP address of the SIM card is an identification for the module within the APN. This enables module-to-module and module-to-server communication within the APN structure.

A good and strong GSM signal in the place where the module's antenna is located is imperative for the proper function of the module. Using the module in places where the signal is weak may lead to interruptions in transmission and possible loss of transmitted data along with increased costs generated by transmission retries.

## 4. Module's design

### 4.1. Topography
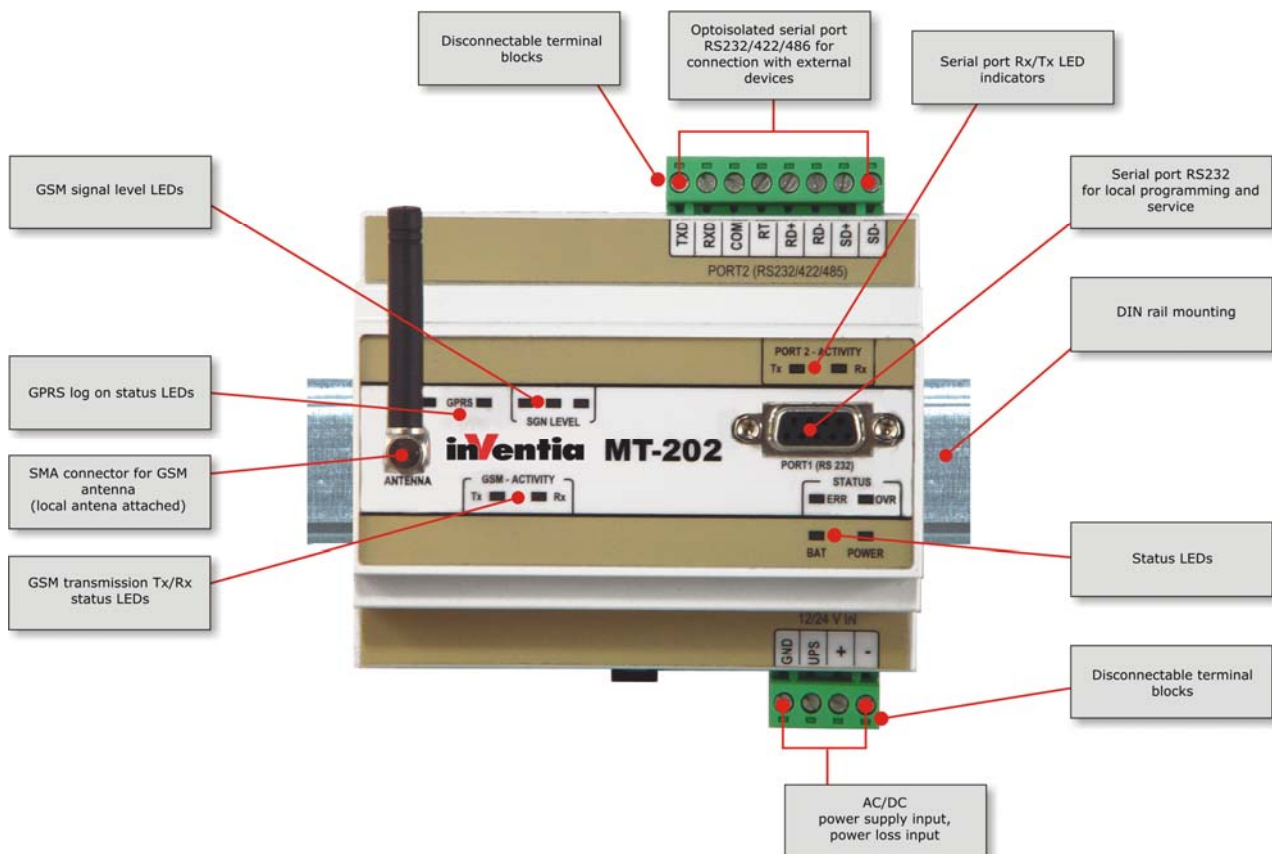


### 4.2. Resources

**MT-202** module's resources

| Serial **PORT 1** | 1 | standard RS232 - configuration and Modbus RTU Slave (ID1, 9600, hardware handshake [CTS/RTS]) |
|---|---|---|
| Serial **PORT 2** | 1 | standard RS232/422/485 - different communication protocols |

## 4.2.1. Serial ports

**MT-202** Telemetry Module is equipped with two serial ports **PORT1** and **PORT2**.

**PORT1** works only in **RS232** mode and is dedicated to local configuration of parameters. In order to perform local configuration, connect this port to a PC-class computer with running MTM program. Operating parameters of this port are not modifiable and the length of the connecting cable should not exceed 3 m. **This port is not isolated!**

**PORT1** can also be used as Modbus RTU Slave port without need of changing modules configuration. Module's Modbus ID on this port is always 1 (options in configuration applies only to **PORT2**). Transmission parameters are:
- speed of 9600bps,
- 8 data bits,
- no parity,
- 1 stop bit,
- hardware handshake (RTS/CTS).

This port is ideal for connecting external graphical or text panel supporting Modbus RTU Master.

> **NOTICE!**
> **The first configuration has to be performed locally, via PORT1 in order to provide the module with basic GPRS communication parameters like PIN code and APN name.**

**PORT2** is capable of operating in **RS232/422/485** modes and serves communication with external data sources. The interface and operating mode is selected during module's configuration. This port is optoisolated.

## 4.2.2. Real time clock

**MT-202** Module is equipped with **astronomical time clock** (**RTC**).

The clock is a base for defining working cycles of module, timers and time stamps for measurement results recorded in registers. Imprecise clock setting results in faulty time stamping and subsequent loss of vital information. For that reason, it is recommended to set the clock to UTC time instead of the local time zone of the module's placement.

> **CAUTION!**
> **The module's RTC clock does not automatically adjust to summer/winter time. To avoid loss of data during manual time adjustment, UTC time is recommended .**

> **CAUTION!**
> **The RTC clock is powered from an internal battery, and as long as it is operational there is no need to reset the time after power-off. Since the clock precision is not absolute, periodical time adjustment may be necessary.**

Setting the time is described in configuring mode documentation for the **MTM** program.

### 4.3. Internal resources

#### 4.3.1. Registers

**MT-202**T elemetry module has in it's internal resources 16 bit input registers and 16 bit internal registers. Remote access to these areas is possible using standard Modbus commands.

Internal Registers are not reset at power off.
Input Registers are reset at power on.

Module's 16 bit registers store unsigned values in range 0-65535. In order to increase the range of stored values, pairs of 16 bit registers were reserved to create 32 bit registers storing signed values for use in user/defined internal program.

#### 4.3.2. Virtual registers

**MT-202** Telemetry module feature 16 bit Virtual Registers. They reflect input (VREG_BIx) and output (VREG_BOx) bit spaces. Using virtual registers gives easy access to bit groups and copying data between bit spaces and registers. Access to registers is possible through module's internal program using standard programming functions described in the chapter Programming.

#### 4.3.3. Clocks

**MT-202 Telemetry Module** is equipped with 4 programmatic clocks divided in two groups with different functionalities.
2 programmable **Asynchronous clocks** TMR1, TMR2, enable cyclical time measuring for up to 100 days (8 640 000 s) and 2 programmable **Synchronous clocks** TMR3, TMR4, enable cyclical time measuring from 1 min to 24 h with synchronization with module z RTC clock.

Properly configured clock counts the time setting after each completed period, for one program cycle, a flag in binary inputs space (respective *TMR1, TMR2, TMR3, TMR4*). These flags can be used in internal controlling program or for triggering transmission on event.

#### 4.3.4. MT2MT Buffer

**MT2MT** buffer enables creation of system where MT-101/102/202 modules may exchange information (Internal Registers) with each other without any relaying instance.
Data transmission from one module to another goes like this:
1. In sending module the event/triggered sending of the buffer has to be defined.
2. In receiving module switch the **MT2MT** buffer on and define it's placement and size so that it encompasses the area of sent registers.
3. Upon reception of event-triggered data frame, registers from event-reporting module are copied into receiving module's registers and MT2MT_x bit informing about modification of MT2MT buffer with data from respective IP is set. (Bit number reflects the index of IP address in GPRS/Authorized numbers in configuration). MT2MT_x bits are set for 1 program cycle immediately after reception and recording it in MT2MT buffer.
4. Data transmission in this system copies registers of sending module into the exact same register in receiving module. When designing communication between larger number of modules, separate register areas have to be sent and appropriately large areas define for **MT2MT receiving buffer** in receiving modules.

#### 4.3.5. Parameters

In **MT-202** module max. 128 constant parameters that are loaded into module's memory during initialization of the module can be defined.

These constants are stored in 16 bit registers and have numerical values ranging from 0 – 65535.

Constant parameters may be used for parameterization of user programs.

Defined parameters are not reset after power loss or module is reset.

### 4.3.6. System variables

**MT-202** has system variables connected to GSM/GPRS connection state and power supply. Variables state is reflected by flags that can be used as transmission triggers or in internal control programs.

FS1_ups = 1 - loss of potential at module's UPS pin
FS1_gprs = 1 - informs upon log off from GPRS network

Full list of system variables is placed in Memory map chapter in Appendices.

## 4.4. LED diodes

**LED** indicators placed on **MT-202** front panel are convenient during module start up phase.
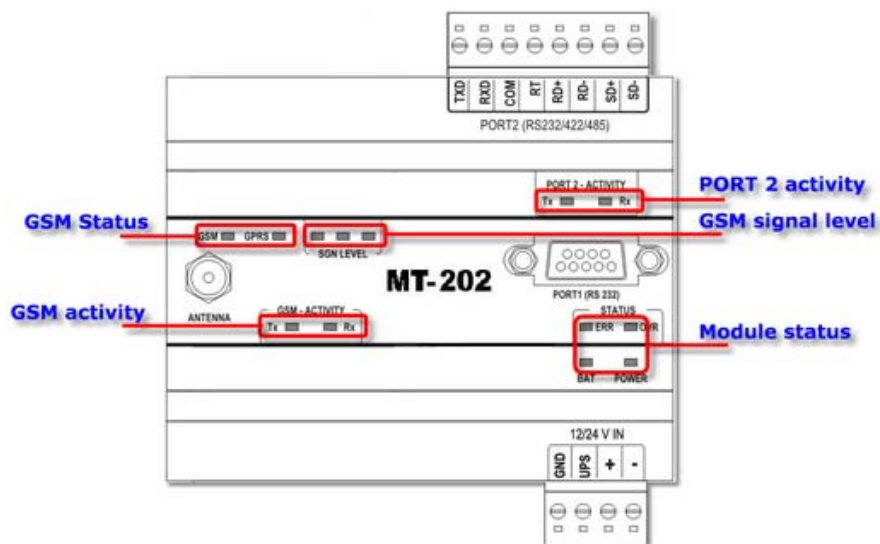


Detailed description of signaling patterns is placed in chapter LED signaling.

## 4.5. SIM card

**MT-202** telemetry module is equipped with standard miniature SIM card holder for connecting card to GSM modem.

If use of GPRS transmission is intended the SIM card should have GPRS option and possibility of login to APN for assigning static IP addresses activated. In absence of static IP address, use of the module for GPRS transmission is impaired.

Proper placement of the SIM card is imperative for module's operation. The module accepts only SIM cards operating in low potential technology 3,3V.

## 4.6. Antenna

Attachment of antenna is essential for proper operation of MT-202 telemetry module. SMA socket is placed on module's front panel. The attached antenna has to secure appropriate radio signal level enabling login to GSM network.

The type and placement of antenna has significant influence on module's sender/receiver circuits. GSM signal level is reflected by **SGN LEVEL LEDs** on module's front panel . When GSM signal level is not sufficient for reliable operation LED is not lit. In that case the use of a directional antenna should be considered.
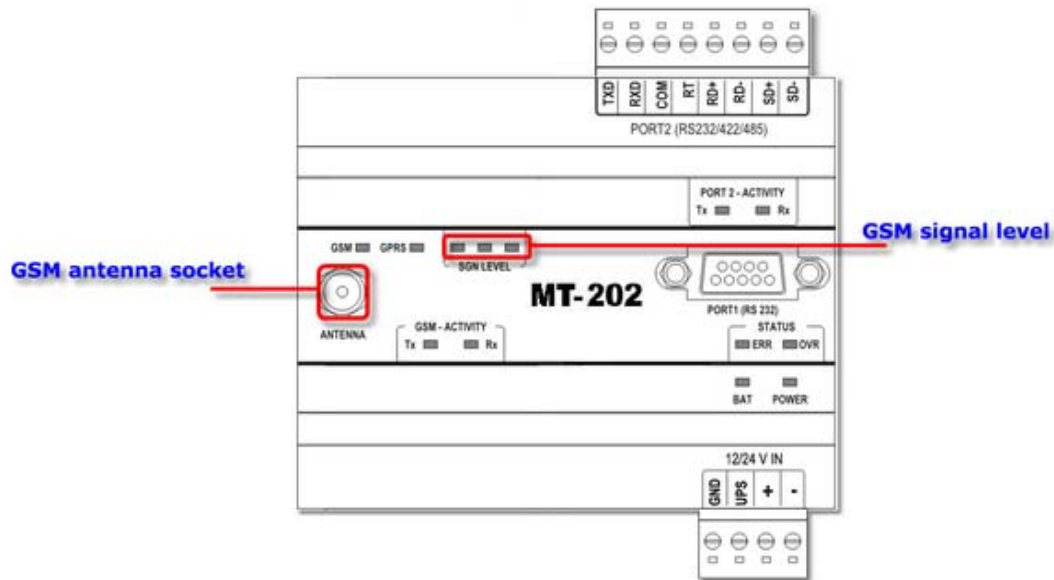


## 4.7. Power supply

**MT-202** may be powered by 18...26,4 V (AC) or 10,8...36 V (DC).

<div style="border:1px solid; background:#ffffcc; padding:10px; text-align:center; color:red;">

**NOTICE!**
**Exceeding the range of power supply may cause faulty operation or damage the module!**

</div>

The module may work with auxiliary battery supply securing operation for some time after main supply failure. In order to discriminate whether the module is powered from battery or from main supply the module has a binary input marked **UPS**, where the signal informing that main supply has failed. Main supply voltage drop below 10,8 V raises the *FS1_ups* system flag. The flag may be used for rules processing. The input may be used to signal main supply failure and switch over to battery supply.

## 4.8. Housing

MT-202 module is encapsulated in standard housing made of plastic compliant with safety requirements and protecting the module in standard operating environment.

The applied solution complies with standard industrial requirements for DIN rail mounting.

# 5. Connections scheme

Tis chapter shows standard configurations of connections securing proper operation of **MT-202** module's integral inputs in all available operating modes.

## 5.1. Communication ports

Telemetry module is equipped with 2 communication ports for different applications.
They are :

**PORT 1** (RS232 – configuration, Modbus RTU Slave [ID1])
- The not optically-isolated Port is used for configuration
- Connection point to point to PC via RS-232

DB-9 connector (female)

| Pin | description |
|---|---|
| 1 - | |
| 2 – TXD | transmitter output |
| 3 – RXD | receiver input |
| 4 - | |
| 5 – GND | ground |
| 6 - | |
| 7 – CTS | handshake input |
| 8 – RTS | handshake output |
| 9 - | |
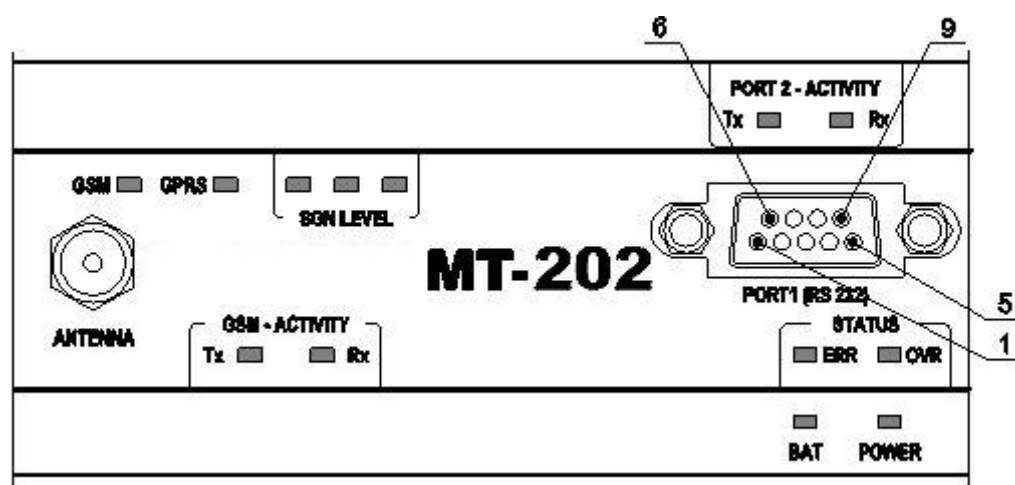


**PORT 2** (RS232/422/485 – communication)
- The Port is optically-isolated, used for data exchange.
- Maximal isolating voltage 60Vrms.
- Operating mode selected by configuration application.

Description

| pin | Description |
|---|---|
| TXD | transmitter output |
| RXD | receiver input |
| COM | interface common ground (GND) |

| RT | terminator – if necessary connect to RD+ |
|---|---|
| RD+, RD- | for RS485 (transmitter, receiver), for RS422 (receiver) |
| SD+, SD- | for RS422 (transmitter) |
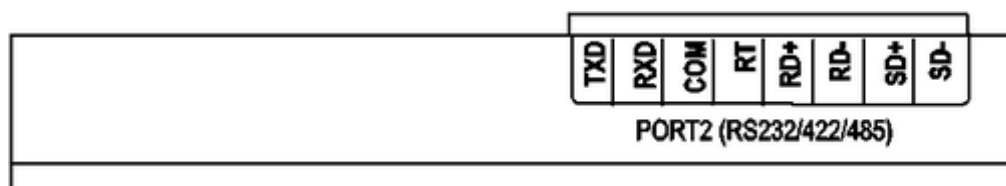


For RS232 mode the length of the cable should not exceed 15 m.

**NOTICE!**
- Supply cables length < 10 m
- Signal cables length < 30 m
- For longer cables it is recommended to use external overvoltage protection.

## 5.2. Power supply

The power supply is connected to „+" and „-" terminals (preserving polarisation only when supplying direct current).

| Pin | Description |
|---|---|
| GND | Module's ground * |
| UPS | Input – power supply state signaling. Active state for voltage > 10,8V<br>When not employed , shortcut with +. |
| + | Positive pole of power supply** |
| - | Negative pole of power supply** |
| \* - Not recommended in standard appliances since it may increase emitted noise.<br>\*\* - For AC supply polarisation does not apply. | |

Example:

Connection diagram with battery backup
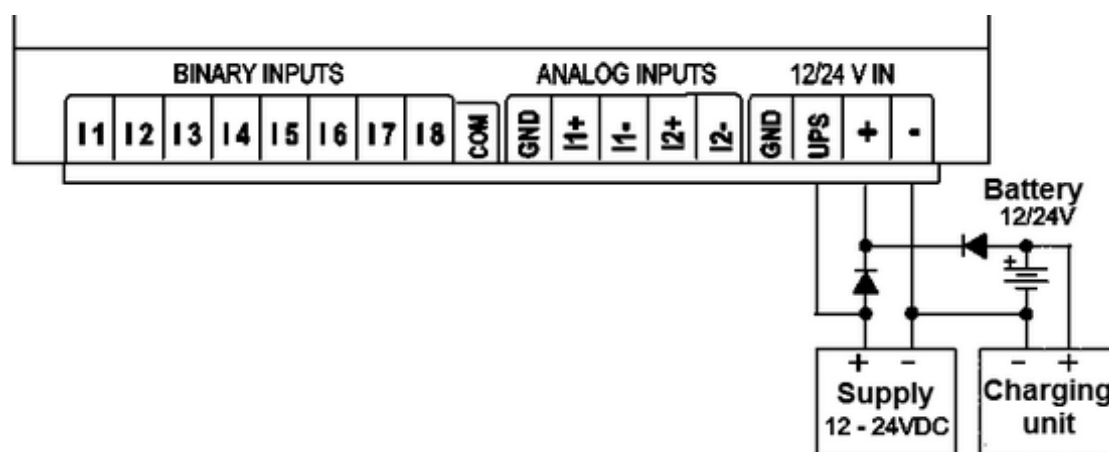
## 6. Starting the module

Starting **MT-202** module requires few basic activities.

Recommended sequence:

1. GSM antenna attachment.

2. Configuration of basic operating parameters
3. Insertion of SIM card
4. Restart of the module

### 6.1. Connecting antenna

Connecting external GSM antenna is an essential condition for proper module operation. The type of antenna depends on the desired mounting type and power of GSM signal in antenna placement area. As previously mentioned, the antenna has to be attached at every power up because it is a necessary transmitter load and absence endangers the module's transmitter part. Even with no SIM inserted, the module exchanges information with available GSM networks in order to secure possibility of sending emergency calls (112).
The antenna is connected to **MT-202** module via SMA connector placed on the front panel of device.
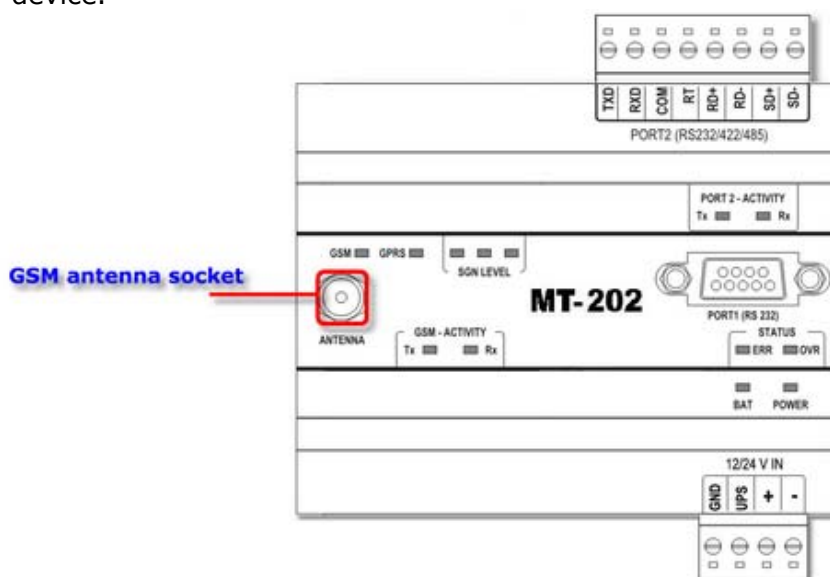
The choice of antenna type depends on GSM signal propagation at place where the module is installed. In most cases, a standard small size antenna is sufficient. Where GSM signal is weak using multi-segment directional antenna may be necessary.

## 6.2. First configuration

First configuration of **MT-202** is necessary for setting up basic parameters making logging to GSM network possible and, optionally, servicing GPRS.

<div style="border:2px solid black; background:#ffffcc;">

**NOTICE!**
**Since a new module or a module configured for other circumstances may not have necessary data for proper login to GSM network, it is necessary to perform the first configuration in local mode by serial RS232 cable connected to Port 1**

</div>

To configure the module, connect it via RS232 cable to a computer running **MTManager**. Comprehensive information about installation, use and attachment of **MTM** program to configured modules can be found in **MTM User Manual**.

Logging into GSM/GPRS network requires basic information about the SIM card in use and optionally about the APN that the module is going to operate within when GPRS mode is turned on.

The parameters are:
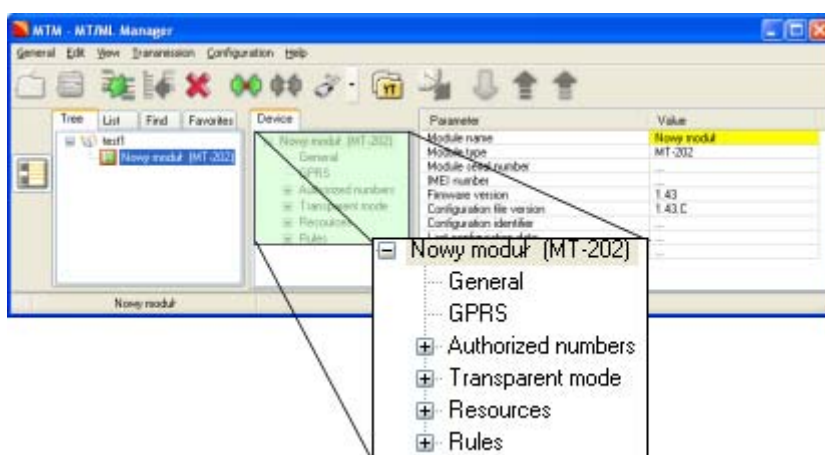


In **General** group:

> *PIN code for SIM card*
> > insert PIN code of the SIM card intended for the module, unless the card is configured not to ask for PIN code.
> *Use GPRS*
> > **Yes - if** SMS and GPRS packet transmission is intended
> > **No** - if the module is to work in SMS mode only.

In **GPRS** group - visible if *Use of GPRS* is set to **Yes**:

> *APN Name*
> > insert name of APN, in which GPRS transmission is to take place.

> *APN User Name*
> > insert user name (if required by Network Operator)

*APN login password*
      insert password (if required by Network Operator)

<div style="border:2px solid black; background:#ffff99; padding:10px; text-align:center; color:red;">

**NOTICE!**
**Upon each writing of new configuration into MT-202, the module performs full RESET**

</div>

These parameters are all that is needed to be able to log in GSM/GPRS network. One has to remember that modules with basic configuration can not send any information. Therefore, after verifying that module logs in to GPRS network, one has to perform full configuration of module parameters enabling all full intended functionality of the module.

## 6.3. Inserting SIM card

One of the fundamental conditions for proper operation of the module is inserting a **SIM** card enabling module to send SMS and/or packet data in GPRS network.

The best way to do it is when the power supply disconnected. Theoretically the SIM card may be inserted into the module before the first configuration is performed, but note that two attempts of entering wrong PIN code the cause module to stop attempting to log into network. Should this happen, the module has to be unlocked.

<div style="border:2px solid black; background:#ffff99; padding:10px; text-align:center; color:red;">

**NOTICE!**
**Earlier models of MT-202 do not have the function preventing triple attempt of inserting wrong PIN code resultingn permanent locking of access to SIM card and subsequent permanent locking of module. Should this happen, the SIM card has to be unlocked using the PUK code.**
**The details of the procedure unlock the card is described in chapter Problem solving**

</div>

Insert **SIM** card in cradle and slide it into the slot as depicted below until the gentle click is heard.
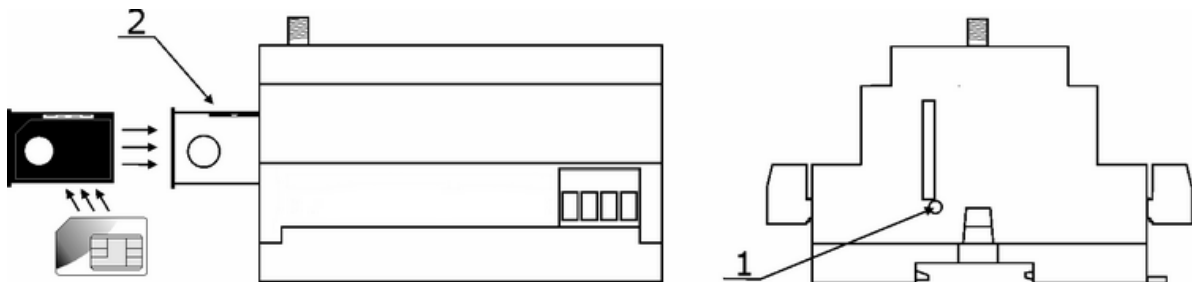


Properly inserted **SIM** cards secure reliable connection with module's terminals.

## 6.4. Start up

After performing the first configuration and inserting SIM card, proceed to start up. Switch the power supply off and on again to reset the module.

<div style="border:2px solid black; background:#ffff99; padding:10px; text-align:center; color:red;">

**NOTICE!**
**If the SIM card was inserted to the module during first configuration resetting the module is unnecessary as every writing of new configuration forces RESET of the module.**

</div>

Wellconfigured MT-101 module logs into GPRS network within 20 -30 seconds. The login sequence is displayed by diagnostic LEDs at the front panel of the module. Explanation of sequence is described in details in LED signaling chapter.

In case of difficulties when logging into GPRS, verify configured parameters and pay attention to LEDs indicating GSM signal level.
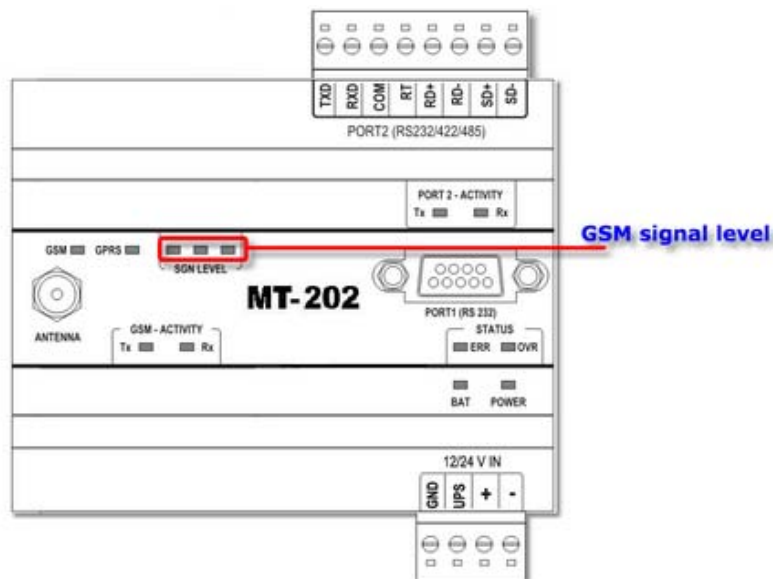


Too weak GSM signal may render log-in impossible.


## 7. Module's operating modes

Following chapters describe particular operating modes of **MT-202** module.
Full benefit of module's features depends on knowledge of different modes functionalities. Therefore we encourage to study this section with great attention.


The module operates in following modes:

- **Transparent mode** - enables efficient transmission of data frames sent by unknown serial protocols
- **Modbus RTU Maste Mode** - turns module into router of Modbus RTU packets
- **Modbus RTU Slave Mode** - enables local and remote access to module's internal and external resources and to internal resources of connected to PORT2 other Slave devices employing same transmission protocol.
- **Modem Mode** - enables (in justified cases) abandoning all functionality of MT-202 and using it as a plain, but automatically logging into selected network GSM/GPRS modem.
- **Modbus RTU Mirror Mode** - adds to Modbus RTU Slave mode a very sophisticated functionality enabling using the module as local Master of Modbus RTU protocol. The module is mapping registers of external Modbus devices into it's own internal registers. This functionality significantly reduces GPRS transmission necessary for efficient control.
- **GazModem Mode** - enables local control of devices working in GazModem protocol - not available in present version of module's firmware.
- **M-Bus LEC Mode** - enables local control of devices of LEC type (heat consumption meters) and other devices with identical mapping of data holding registers - not available in present version of module's firmware.
- **NMEA 0831 Mode** - prepared to read information subset transmitted to PORT2 by devices operating within NMEA 0831 standard.
- **FlexSerial Mode** - prepared to adopt other transmission protocols by employing user defined internal program.

## 7.1. Transparent mode

In this mode, the module receives data at serial PORT2, buffers it in memory and converts to packets to send to defined recipients.
This mode allows data transmission without recognition of the protocol the data is sent in.
Data received at PORT2 is transmitted to all IP addresses defined in Authorized IP numbers group when working in broadcast mode.

In order to reduce or eliminate transmitted data redundancy, two options are available:
- packet routing - useful for unambiguous definition of where in transmitted frames the recipient address is placed and how it is recorded without knowing the foreign protocol. In that case, routing table can be created to eliminate redundant transmissions.
- transmission channel reservation - diminishes transmitted data volume provided that multiple queries of one device are executed not knowing the foreign protocol. When channel reservation is employed only the first query is sent to all modules defined in Authorized IP numbers list. After reception of the response the module reserves, for a defined time, the direct communication channel with the module that responded to the last query thus creating a temporary point to point connection. The reservation time is counted since reception of last GPRS packet. Consecutive packets received from that module prolong channel reservation while sending packets does not prolong the reservation. When channel reservation time is set to 0 [sec], the module does not reserve channel and works in standard mode receiving and sending packets from all defined remote units.

## 7.2. Modbus RTU Master mode

In this mode, the module cooperates with the device acting as MODBUS RTU MASTER.
The master sends queries addressed to Slave devices to module's PORT2 .
Upon reception of the frame at PORT2, the module verifies recipient device address. Since module's own resources are available in Modbus RTU Master mode under Modbus ID assigned to module, the instruction is carried out and replay sent to PORT2 (to the master) when matched with address.
Frames with other Modbus ID addresses transmitted via GPRS are routed to other recipients on the basis of a table assigning IP addresses to Modbus IDs (routing table - redirection). In other words, MT-202 acts as a router of packets sent from Master to remote Slave units. This mode enables creation of data acquisition system, where standard queries generated by PLC controller or any other device working in Modbus RTU Master mode. This solution is not recommended if master role is played by a PC running Windows 2000 or newer and collected data is used by applications capable of using data collected by OPC or written in CSV or SQL format.

Following rules apply for Modbus RTU Master mode:

- Frames are sent only to recipients having Modbus ID matching address received via PORT2.
- All frames received at PORT2 are sent to recipients having Modbus ID=0 written in the table.
- Frames with address 0 (broadcast frames) are set to all recipients declared in the routing table.
- Repeating IP address in the table does not cause multiple transmission to the same recipient. This makes sending frames with different Modbus IDs to same IP recipient possible, connecting a large number of devices with various Modbus IDs to PORT2 of the remote module operating in MODBUS RTU Slave mode.
- The frame will be dispatched to recipient only when IP address is defined in Authorized numbers list of GPRS section and sending option is checked.

- After receiving frame via GPRS, the module checks if it is addressed to its internal resources (address matches declared Modbus ID). If that is the case, the instruction is carried out and confirmation sent to sender. This behaviour enables creation of Multi-master systems securing proper data transmission and access to all resources of the system.
- If Modbus ID of received frame does not match module's ID, the frame is forwarded to PORT2 and may be received by the Master device.
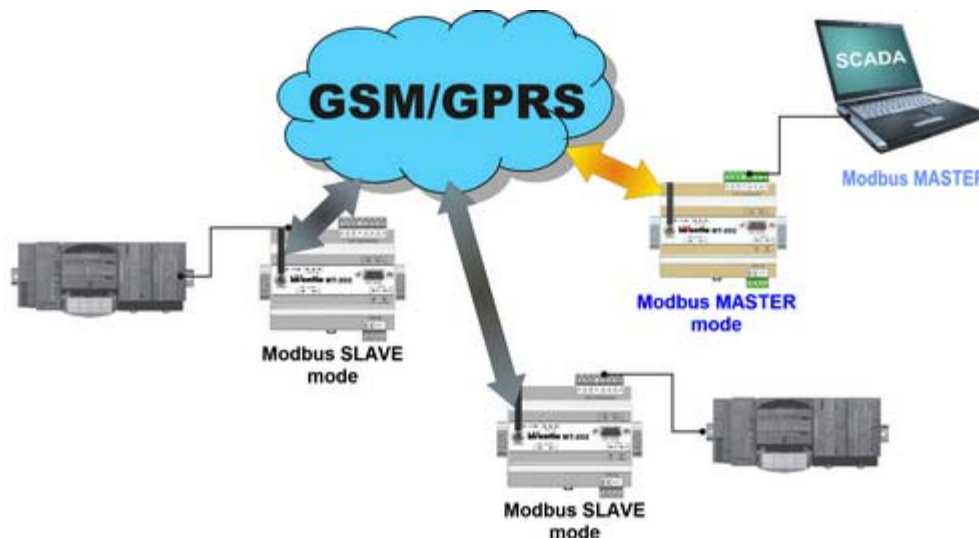


## 7.3. Modbus RTU Slave mode

In this mode, the module acts as a network node serving both own resources and all connected to PORT2 devices operating in Modbus RTU Slave mode, waiting for GPRS data frames.

Upon reception of the frame, it investigates recipient's address (matching declared Modbus ID with its own).

If the address matches, the instruction is carried out and confirmation set to sender. If it does not match the frame is forwarded to PORT2. The module waits for 0,5 sec for reply. When reply comes or the time elapses, the module whether new data frame arrived via GPRS.

Sending frames not addressed to module's ID to PORT2 is unconditional as module does not have the list of connected to PORT2 Slave devices.

Replies are sent to sender as standard. Sending of replies to additional Master units can be defined in routing table.

The IP address of additional Master is stated in the second column of the table, and the device Modbus ID from which replies have to be sent to additional master are stated in third column.

If third column states "0", replies from all slaves will be sent to corresponding IP address.
This means that it is possible to send replies to any number of devices acting as protocol Master.(Multimaster mode).
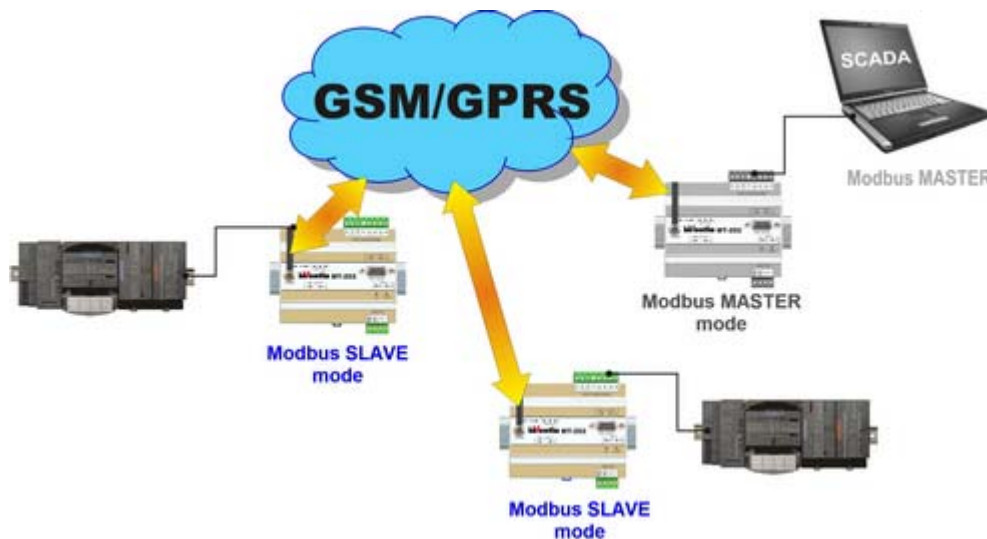
## 7.4. Modem mode

In this mode, the module establishes communication channel between PORT2 and internal GSM modem input. The channel is available after entering PIN code and logging on to the network, relieving the user of modem initiation.

In order to control the transmission, AT commands issuing is necessary. Full control of modem's operation is carried out by external device connected to PORT2 of the module.

This mode does not require configuration of any parameter beyond right PIN code but in turn does not give access to any MT-202 features except transmission control.

Being able to use transmission control, modem may be reset if no activity is detected in defined time. It prevents the phenomenon of an irrevocably "hanging modem" which in case of remote deployment is usually expensive.

## 7.5. Modbus RTU Mirror mode

**Modbus RTU Mirror** mode is an extension of MODBUS RTU Slave mode. Seen from GPRS side the module acts exactly as it does in Modbus RTU Slave mode securing remote access to own resources as well as resources of any slave connected to PORT2. The addition feature provides the possibility of resource mapping of devices connected to PORT2 into internal module Registers. With this option active, the module cyclically reads mapped areas of external devices and refreshes internal Registers.

Writing to external devices is possible as well. In Mirror mode, for every internal program cycle (100 ms) the module compares state of mapped areas and upon detection of change in any Register, a frame writing the change to external device is generated. Values in internal registers may be changed remotely by master system or locally by internal (user defined) program.

Highlights of Modbus RTU Mirror mode:
- The ability to remotely read resources of different areas and devices by one Modbus frame (reading internal Registers of the module), thus increasing efficiency of GPRS transmission.
- The ability to generate alarms on state change in remote devices. Local data reading in conjunction with processing of data by internal program and function of generating events significantly increases potential and dynamics of the whole system, simultaneously reducing costs of transmission compared to systems based on cyclic querying from master
- The ability of local control of the object including extension devices.

Bear in mind that:
- In communication with Slave devices, the module employs standard MODBUS RTU frames, with functions 1, 2, 3, 4 for reading, functions 5, 6 do for writing singular changes and functions 15 and 16 for block writing.
- Upon power up or start of communication with peripheral equipment, the module synchronizes Mirrored space with with Slave's contents, updating its registers with Slave contents. This assures that writing values to Mirrored space during interrupted communication with the slave will not overwrite slave's values upon reestablishment of communication. Bits informing about state of communication with slaves (SL1_ok .. SL16_ok)corresponding to defined during configuration areas, reside in transmission in binary inputs space. High corresponding bit signals good communication with particular slave.
- In Modbus RTU Mirror mode Slaves mirrored in module can not transmit events. When servicing events from slaves is required, standard Modbus RTU Slave mode is recommended.

## 7.6. GazModem mode

**GazModem** mode is an extension of Modbus RTU Slave mode with functionality similar to that available in Modbus Mirror mode with mapping of register space of peripheral Slave into MT-202 internal registers.

This mode was implemented due to requirement of reading from devices operating in Gaz-Modem and Gaz-Modem2 protocols created by Plum Sp. z o.o. for use in different volumetric gas counters and other gas systems. The user employing telemetry modules is capable of reading current data, binary inputs and alarm signals from gas devices.

## 7.7. M-Bus LEC mode

**M-Bus LEC** mode is an extension of Modbus RTU Slave mode with functionality similar to that available in Modbus Mirror mode with mapping of register space of periferal Slave into MT-202 internal registers.

This mode was implemented due to requirement of reading from devices operating in M-Bus protocol, applied mainly for measuring of energy consumption in heating systems. Since MT-202 module has no capability of direct communication with M-Bus units it is necessary to connect an external converter to PORT2. The converter RM-120 is one of our products.

## 7.8. NMEA 0183 mode

**NMEA 0183** mode is an extension of Modbus RTU Slave mode with the function of receiving and interpreting of data in NMEA 0183 format sent by peripheral device attached to PORT2.
Standard NMEA 0183 protocol frames are decoded and stored in module's internal registers where they are ready for further processing.

This mode was implemented to serve the AIRMAR weather station, but may be used to communicate with GPS receivers using NMEA 0183 protocol for data exchange.

## 7.9. FlexSerial mode

In **FlexSerial** mode, the transmission at PORT 2 is serviced exclusively by a user program resident in the module. Only standard access to module's resources is possible via GPRS.

No option of sending data frame to PORT 2 while data incoming to PORT2 arrives exclusively to internal program is available.

## Buffer structure

PORT 2 in the program is served by 2 buffers and helps registers placed in holding register's memory space. Transmission and reception are independent on each other.
The buffers have a length of 512, 16 bit registers, where only low order bytes are used for transmission. In other words, one 16 bit register stores 1 byte on low order byte. High order byte remains unused.

## Data reception

Configuration for FlexSerial mode uses 2 variables that define the way data stream received at PORT 2 is converted into data packets available for internal program.
Received data is stored in registers. Registers P2RCV_B1…P2RCV_B512, build the receiving buffer. Register P2RCV_NO controls data reception.
Data received at PORT 2 is internally buffered in 6kB buffer and put into receiving buffer only when register P2RCV_NO has "0" value. During data insertion to the buffer the length of inserted package is recorded into P2RCV_NO register. The user program should process data from receiving buffer after detecting a non zero value in P2RCV_NO buffer and when done should set P2RCV_NO to 0 value, thereby initiating loading of next package to buffer. Received data is always placed at the beginning of the buffer so the first byte of the package is placed in P2RCV_B1 register.
Furthermore, in binary output space there is P2RCV_err bit that is set high when reception error occurs. F.e.: receiving buffer overflow, parity, stop bit error. P2RCV_err bit is not automatically reset so in order to be useful for diagnostic purposes, it has to be reset by the user program. The state of this bit does not have influence on data reception algorithm and is only used for received data diagnostic.

## Resources

| Name | Space (address DEC) | Description |
| --- | --- | --- |
| P2RCV_NO | Internal registers (1022) | Register controlling reception |
| P2RCV_B1…513 | Internal registers. (1024…1535) | Receiving buffer |
| P2RCV_err | Output bits (60) | Reception error signaling |

## Data transmission

Registers P2SND_B1…P2SND_B512 constitute sending buffer, while register P2SND_NO controls the transmission. writing a non zero value to P2SND_NO triggers transmission of data packet with length equal to the value written into P2SND_NO from sending buffer. Data is transmitted from the beginning of the buffer, so the first Byte of the packet is taken from P2SND_B1 register.
After emptying the buffer, the register P2SND_NO is automatically zeroed thus informing that the next packet may be prepared for sending. Register P2SND_Bx may be modified only when P2SND_NO is zeroed.
If transmission error occurs (for example during transmission to P2SND_NO the value higher than 512 was written), P2SND_err bit in binary outputs space is set high.
Bit P2SND_err is not zeroed automatically so in order to use it in diagnostic purposes it must be zeroed by user's program.

## Resources

| Name | Space (address DEC) | Description |
| --- | --- | --- |
| P2SND_NO | int. Registers (1023) | Register controlling sending |
| P2SND_B1…513 | int. Registers (1536…2047) | Sending buffer |
| P2SND_err | Output bits (61) | Sending error signaling |

# 8. Configuration

## 8.1. General information

The configuration of MT-202 module, as is the case for other modules in the MT series, is carried out using the MTM (MT Manager) program portal, delivered free of charge to users of our telemetry solutions.

The portal is a specialized environment providing full control of the entire telemetry system regardless of the system's size. The possibility of dividing hardware resources into Projects and Folders facilitates efficient management of very complex telemetry systems.
After adding a new module to the environment and selecting it, all module parameters are available for editing. Detailed description of functions and their applications are to be found in MTM user manual.

> **NOTICE!**
> **Availability of different functions and parameters depends on module's firmware version and the settings of parameters they may be dependent on.**

## 8.2. Parameter groups

For clarity and ease of use, the operating parameters of MT-202 module are divided into logically or functionally connected groups in the following order:

**Header group** - contains unmodifiable parameters describing the module, its firmware and configuration.

**General group** - contains basic parameters defining module's operating mode

**GPRS group** - contains necessary parameters to log in to GPRS network and defines parameters vital for transmission reliability.

**Authorized numbers group** - contains lists of phone numbers and IP addresses of other terminals authorized for communication with configured module.

**Operating mode group** - contains parameters necessary for configuring currently selected operating mode.

**Resources group** - defines parameters for hardware and software resources related to reading and processing measurement data.

**Rules group** - contains lists of transmission tasks to be carried out upon occurrence of activating criteria

### 8.2.1. Header

Header of parameter structure describes **MT-202** telemetry module. It holds basic information unique to the module, the configuration contained by module and configuration file version. Information displayed is not user editable and solely used for verification and information purpose.

### 8.2.1.1. Module name

| | |
|---|---|
| **Function** | - displays name assigned to module during configuration |
| **Data type** | - text |
| **Range** | - n/a, read-only parameter |
| **Comments** | - n/a |

### 8.2.1.2. Module type

| | |
|---|---|
| **Function** | - displays the type of configured telemetry module |
| **Data type** | - text |
| **Range** | - n/a, read-only parameter |
| **Comments** | - n/a |

### 8.2.1.3. Module serial number

| | |
|---|---|
| **Function** | - displays serial number configured telemetry module |
| **Data type** | - text |
| **Range** | - n/a, Read-only parameter |
| **Comments** | - this field displays module serial number assigned during manufacturing. This number is static and unique identifier of the unit. |

### 8.2.1.4. IMEI number

| | |
|---|---|
| **Function** | - displays GSM modem's IMEI number |
| **Data type** | - text |
| **Range** | - n/a, read-only parameter |
| **Comments** | - n/a |

### 8.2.1.5. Internal program version

| | |
|---|---|
| **Function** | - displays the identification of actual version of internal telemetry module program (firmware) |
| **Data type** | - text |
| **Range** | - n/a, read-only parameter |
| **Comments** | - the value of this field changes automatically after download of firmware |

### 8.2.1.6. Configuration file version

| | |
|---|---|
| **Function** | - displays version identification of configuration file used for actual configuration |
| **Data type** | - text |
| **Range** | - n/a, Read-only parameter |
| **Comments** | - value depends on module's firmware version. Auxiliary extension character defines the sub-version |

### 8.2.1.7. Configuration identifier

| | | |
|---|---|---|
| **Function** | - | displays identification of current configuration |
| **Data type** | - | hexadecimal |
| **Range** | - | n/a, read-only parameter |
| **Comments** | - | the value of this parameter increases automatically by 1 after each successfully stored configuration. |

### 8.2.1.8. Last configuration date

| | | |
|---|---|---|
| **Function** | - | displays time and date of last successful configuration change |
| **Data type** | - | text |
| **Range** | - | n/a, read-only parameter |
| **Comments** | - | the value changes automatically with successful configuration change. Useful in tracing unauthorized configuration changes. |

### 8.2.1.9. Last read of device time

| | | |
|---|---|---|
| **Function** | - | displays internal module time recorded during last configuration reading or during last time setting |
| **Data type** | - | text |
| **Range** | - | compliant with Date and Time format |
| **Comments** | - | this field is useful in verifying last access time and checking internal module clock settings (RTC) |

## 8.2.2. General

General group encompasses parameters vital for whole module.
Contains data necessary for successful login to GSM network and parameters defining module operating mode.
Note: values set here have impact on module's behavior and in worst case, when chosen improperly may even lock the module.

---

**NOTICE!**
**Availability of parameters described in following chapters depends on firmware version and the superior parameters they depend on.**

---

### 8.2.2.1. Mode of operation

| | | |
|---|---|---|
| **Function** | - | defines module mode of operation in terms of internal functions and serviced protocols. The selection controls access to parameters connected with selected operating mode |
| **Data type** | - | selection list |
| **Range** | - | *Transparent* mode allowing sending GPRS data to PORT2. *Modbus RTU Master* in this mode, the MT-202 module cooperates with external unit acting as System Modbus Master connected to PORT2. The module routing table assigns Slaves' Modbus IDs to IP addresses of their respective GPRS serving nodes in order to optimize transmission. |

*Modbus RTU Slave*

    in this mode, the internal module resources are treated as resources of the slave with Modbus ID defined during configuration. Frames received by the module carrying its Modbus ID are serviced locally while frames carrying other Modbus IDs are sent to PORT2.

*Modem*

    in this mode, the module acts as GPRS modem controlled by AT commands. At power on, the module uses PIN code defined during configuration and logs on to GSM network.

*Modbus RTU Mirror*

    mode enables remote querying of module resources along with attached units working in Modbus RTU protocol and event-triggered data sending, as well as creating a mirror copy of connected units in module's internal registers.

*Transparent PLUS*

    identical with *Transparent*, extended with ability of querying internal resources and event processing.

*GazModem*

    mode enables cooperation with units connected to PORT2, using GazModem protocol. MT-202 acts as a local Master enabling local reading of connected units and alarm processing. This mode minimizes transmission required to supervise units. The mode is not supported by standard firmware and requires firmware marked as MT-202_x.xxgm.bin.

*M-Bus LEC*

    this mode is dedicated to cooperation with units employing M-Bus protocol. As in the GazModem mode, the MT module acts as local master for units connected to PORT2, enabling readings from central heating nodes. The mode is not supported by standard firmware and requires firmware marked as MT-101_x.xxgm.bin.

*NMEA 0831*

    this mode is dedicated to NMEA 0831 protocol. Received data is stored in module's internal registers compliant to established standards.

*FlexSerial*

    this mode enables integration of any kind of PLC controllers, I/O units, measurement equipment, operator panels equipped with serial communication port RS-232/422/485 on the base of programmed servicing of non standard protocols.

**Default value**   - *Transparent*

**Comments**   - choosing the appropriate working mode is essential for exploiting all available features. The choice influences availability of configuration parameters thus optimizing module operation.

### 8.2.2.2. SIM card's PIN code

**Function**   - defines PIN access code for SIM module delivered by GSM operator. For SIM modules not protected by PIN code, the value is insignificant.

**Data type**   - Text

**Range**   - Letters and numerals, max 8 characters

| **Default value** | - | 0000 |
|---|---|---|
| **Comments** | - | wrong pin can cause locking of SIM module |

<div style="border: 2px solid black; background-color: #ffff99; padding: 10px;">

**CAUTION!**

**Caution is vital when setting the PIN code value . Entering faulty PIN code may cause module start-up to be impossible and lock SIM card. In latest versions of the module, attempting to enter wrong PIN code twice renders a third attempt impossible.**

</div>

### 8.2.2.3. GSM band

| **Function** | - | sets the frequency of locally-employed GSM system. |
|---|---|---|
| **Data type** | - | selection list |
| **Range** | - | *EU-900/1800 MHz*<br>        for areas employing 900/1800 MHz standard<br>*US-850/1900 MHz*<br>        for areas employing 850/1900 MHz standard |
| **Default value** | - | *EU-900/1800 MHz* |
| **Comments** | - | Parameter active only in modules using quad-band Wavecom WISMO Quick PLUS modem module. Setting US-850/1900 MHz band in other modules does not have any effect. |

### 8.2.2.4. Access to configuration

| **Function** | - | defines configuration access restrictions. The user can decide whether access to configuration will be allowed for all or only selected IP addresses. |
|---|---|---|
| **Data type** | - | selection list |
| **Range** | - | *All*<br>        Unrestricted access for any IP address within the APN<br>*List*<br>        Access limited to addresses defined in the list of *Authorized numbers, IP* for addresses with *Configuration* option set to *Allowed*. |
| **Default value** | - | *All* |
| **Comments** | - | Note that access restriction configuration applies only to GPRS connection and when used improperly may block remote access for users that should have the right to configure the module. |

### 8.2.2.5. Configuration password

| **Function** | - | defines the password protecting access to configuration of the module. The password will be required for both local and remote access, thus protecting against unauthorized configuration alterations. The password does not protect against reading current configuration or the module status. |
|---|---|---|
| **Data type** | - | text string |
| **Range** | - | letters and numerals, max. 32 characters |
| **Default value** | - | n/a |
| **Comments** | - | since the only way of unlocking the module is resetting it to factory settings, it is vital that the password is stored in a safe way and available when needed. |

### 8.2.2.6. Configuration read disable

| | | |
|---|---|---|
| **Function** | - | blocks reading of module configuration even when using valid password |
| **Data type** | - | selection list |
| **Range** | - | *Yes* |
| | | Reading of configuration from the module is impossible. |
| | | *No* |
| | | Module is not protected against reading of configuration |
| **Default value** | - | *No* |
| **Comments** | - | This parameter does not influence writing a new full configuration but prevents writing changes if configuration identifiers in the module and in MTM application do not match |

### 8.2.2.7. Reset after inactivity

| | | |
|---|---|---|
| **Function** | - | Defines (in minutes) the interval between data reception by modem. Initiates modem restart when exceeded. |
| **Data type** | - | number |
| **Range** | - | 0....10080 [min] |
| **Default value** | - | 60 [min] |
| **Comments** | - | This parameter is valid only for Modem mode. Value 0 disables this function. The value of this parameter should not be lower than master system query interval. Setting of lower value results in frequent resets of the modem. |

### 8.2.2.8. Data overwriting protection

| | | |
|---|---|---|
| **Function** | - | blocks writing of data into internal module registers, protecting data significant to proper operation of module. |
| **Data type** | - | selection list |
| **Range** | - | *Yes* |
| | | Data writing is possible when password is provided |
| | | *No* |
| | | Data writing to internal registers not protected. |
| **Default value** | - | *No* |
| **Comments** | - | This parameter protects the module against accidental or intended intrusion into internal resources without preventing users authorized by password to make changes. |

### 8.2.2.9. Password for data writing

| | | |
|---|---|---|
| **Function** | - | defines password for Data overwriting protection parameter |
| **Data type** | - | text field |
| **Range** | - | letters and numerals, max. 32 characters |
| **Default value** | - | n/a |
| **Comments** | - | when Data overwriting protection parameter is active, writing to module is possible only after providing valid password. The procedure to be followed in this case is to be found in chapter Appendices - Unblocking writing to internal registers |

### 8.2.2.10. Error display time

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) error display time on Status GSM and SGN LEVEL and on ERR LED groups |
| **Data type** | - | number |
| **Range** | - | 1....60 [s] |
| **Default value** | - | 30 [s] |
| **Comments** | - | setting of too small value makes error code identification difficult while too long value extends the restart time after error code display |

### 8.2.2.11. Use GPRS

| | | |
|---|---|---|
| **Function** | - | defines means of communication for the module |
| **Data type** | - | selection list |
| **Range** | - | *Yes* |
| | | The Module works in GPRS mode. Upon power-up module tries to log in to selected APN. This mode requires SIM cards with enabled GPRS access. |
| | | *No* |
| | | The Module works in GSM mode. The only way of remote communication is SMS messaging. In this mode, pre-paid type SIM cards with no GPRS work without problems. |
| **Default value** | - | *Yes* |
| **Comments** | - | n/a |

### 8.2.2.12. Use SMS

| | | |
|---|---|---|
| **Function** | - | defines working sub-mode of module operating in GPRS |
| **Data type** | - | selection list |
| **Range** | - | *Yes* |
| | | Module operating in GPRS mode has SMS services active. As a result, the GPRS session is suspended every 12 min. and the module checks for incoming text messages. Event triggered SMS transmission is instantaneous. |
| | | *No* |
| | | Module operates in GPRS mode and SMS services are inactive. |
| **Default value** | - | *Yes* |
| **Comments** | - | This parameter is available only in GPRS mode. Setting this parameter to *No* may result in filling the SIM card with received messages that are not processed by internal logic. |

### 8.2.2.13. Monthly SMS limit

| | | |
|---|---|---|
| **Function** | - | Defines maximum amount of SMS messages module may send during 1 month in order to prevent uncontrolled number of sent messages thus saving on running expenses. Setting the parameter to *0* removes the limit. |
| **Data type** | - | number |
| **Range** | - | 0 ... 65.535 |

| **Default value** | - 0 |
| **Comments** | - This parameter is available in no GPRS and in GPRS mode with *Yes* option for use SMS chosen. |

<div style="border:2px solid black; background-color:#FFFFCC; text-align:center;">

**CAUTION!**
**Reaching the limit set by this parameter results in unconditional (without warning) stop of SMS sending. Note that no SMS will be sent until the month is up <u>even in alarm situations!</u>**

</div>

### 8.2.2.14. Roaming

| **Function** | - defines whether roaming in foreign GSM network is allowed |
| **Data type** | - selection list |
| **Range** | - *On* |
| | in case of absence of home network, the module will attempt to login to available operator network. |
| | *Off* |
| | login to foreign networks is not allowed |
| **Default value** | - *Off* |
| **Comments** | - This parameter decides whether the Module will try to login to available foreign network in the absence of home network. This is possible only when the SIM card in module has roaming service enabled. |

### 8.2.3. GPRS

GPRS Group encompasses parameters connected to login and data transmission in GPRS system. Parameters defined within this group are mandatory and optional as well as convenient for transmission optimization.

### 8.2.3.1. APN name

| **Function** | - defines APN name selected for GPRS transmission |
| **Data type** | - text |
| **Range** | - letters, numerals, special characters - max. 32 characters |
| **Default value** | - empty |
| **Comments** | - absence of APN name disables login to GPRS network |

### 8.2.3.2. APN user name

| **Function** | - defines APN user name |
| **Data type** | - text |
| **Range** | - letters, numerals, special characters - max. 32 characters |
| **Default value** | - empty |
| **Comments** | - Optional parameter used only if required by GSM network operator |

### 8.2.3.3. APN password

| | | |
|---|---|---|
| **Function** | - | defines password for APN user account |
| **Data type** | - | text |
| **Range** | - | letters, numerals, special characters - max. 32 characters |
| **Default value** | - | empty |
| **Comments** | - | Optional parameter used only if required by GSM network operator |

### 8.2.3.4. Module IP

| | | |
|---|---|---|
| **Function** | - | allows user to define IP number for newly created module definition and displays IP number read from the module configuration that was assigned to the module during last login to GPRS network |
| **Data type** | - | IP number field |
| **Range** | - | 0.0.0.0 - 255.255.255.255 |
| **Default value** | - | 0.0.0.0 |
| **Comments** | - | if the number is not read in nor written manually after local configuration, remote configuration of the module via GPRS will not be possible. |

### 8.2.3.5. IP assignment

| | | |
|---|---|---|
| **Function** | - | selects IP address assignment mode during login to GPRS network procedure |
| **Data type** | - | selection list |
| **Range** | - | *DHCP*<br>IP address is assigned by GSM network according to operator policy. It may be static or dynamic address.<br>*Manual*<br>IP address is assigned by GSM network to the value set in Set IP field by user. This mode is applicable only when operator policy allows forcing IP address by the user. |
| **Default value** | - | *DHCP* |
| **Comments** | - | *Manual* mode is allowed only in few GSM networks |

### 8.2.3.6. Set IP

| | | |
|---|---|---|
| **Function** | - | enables manual entering of IP when parameter IP assignment is set to *Manual* |
| **Data type** | - | IP number field |
| **Range** | - | 0.0.0.0 - 255.255.255.255 |
| **Default value** | - | 0.0.0.0 |
| **Comments** | - | forcing of IP address mode assigned by operator is serviced only in few GSM networks |

### 8.2.3.7. Virtual IP address

| | |
|---|---|
| **Function** | - defines IP address that will be placed in internal data header of frames sent by the module. |
| **Data type** | - IP number field |
| **Range** | - 0.0.0.0 - 255.255.255.255 |
| **Default value** | - 0.0.0.0 |
| **Comments** | - Parameter mandatory in case of operating MT-202 module in GSM networks where operator uses dynamic address translation of internal addresses to static external addresses visible to external users. The parameter enables placing the external IP address under which the internal network node is visible in the header for sent data frame. As a result, external recipients will experience match of sender's IP with IP address written in data header. It is necessary due to double authentication of received data employed by MT-Data Provider (MT-DP). |

### 8.2.3.8. Number of GPRS transmission retries

| | |
|---|---|
| **Function** | - defines number of retries of GPRS transmission in case of not receiving confirmation in time defined by Transmission timeout parameter |
| **Data type** | - number |
| **Range** | - 0....255<br>Setting this parameter to 0 results in sending data without waiting for confirmation of error-free reception. |
| **Default value** | - 3 |
| **Comments** | - In normal conditions, it is not recommended to set this value to higher than 3. This effectively secures against loss of transmitted data without hampering processing following rules. Note that consecutive data will be sent after successful conclusion of current transmission. |

### 8.2.3.9. Transmission timeout

| | |
|---|---|
| **Function** | - Defines waiting time (in seconds) for confirmation of reception of sent data frame. |
| **Data type** | - number |
| **Range** | - 0....655 [s] |
| **Default value** | - 12 [s] |
| **Comments** | - This value in connection with declared Number of GPRS transmission retries defines max. time of one data packet transmission, described by formula: |

$$MaxT = (number\ of\ GPRS\ transmission\ retries +1)*transmission\ Timeout$$

For default values: MaxT = (3 + 1) * 12 = 48s. Please notice that calculated value does not define the time of delivery but the time to elapse before the module considers that transmission to appointed IP address is not possible (the data will be lost due to unavailability of recipient) and moves to sending next data frame awaiting transmission.

### 8.2.3.10. Idle time

| | | |
|---|---|---|
| **Function** | - | Defines the interval (in seconds) for sending data frame (ping) controlling the ability to communicate with the network in case of transmission inactivity |
| **Data type** | - | number |
| **Range** | - | 0....86400 [s] (24h) |
| **Default value** | - | 240 [s] |
| **Comments** | - | in case of inactivity longer than the value defined in this parameter the module sends a control frame in order to check whether transmission is still possible. During network check, control data frame is sent to module's own IP address, respecting timeout and number of retries parameters. The length of the frame is 45B+length of the module's name. The frame is sent to module's own IP address or to the address defined in parameter GPRS testing IP address, if different than 0.0.0.0. In „Proxy" mode, the frame is sent to Proxy server IP address. |
| | | No reply to sent frame after exercising defined timeout and number of retries is considered as transmission failure and sets triggering input FS1_gprs 0--> 1, that can be used for Rules processing (SMS sending). As a consequence, after elapsing of time defined in Wait time after disconnection, the module performs RESET and commences GSM/GPRS login sequence. |
| | | Reduction of this parameter increases the frequency of testing GPRS network state. This shortens possible disruptions of control due to network failures but increases "unproductive" data transmission. |

### 8.2.3.11. GPRS testing IP address

| | | |
|---|---|---|
| **Function** | - | sets IP address where data frames testing GPRS network state are sent. |
| **Data type** | - | IP address field |
| **Range** | - | 0.0.0.0 - 255.255.255.255 |
| **Default value** | - | 0.0.0.0 |
| **Comments** | - | This parameter sets recipient's address for data frames testing GPRS transmission channel sent after defined Idle time elapses. Leaving recipient address at *0.0.0.0* sends data frames to module's own IP address. Any other valid address (within the APN) is accepted as the recipient. |

### 8.2.3.12. Number of login retries

| | | |
|---|---|---|
| **Function** | - | Defines max. number of login to GPRS network retries. Each unsuccessful attempt changes the state of triggering input FS1_gprs from 0 to 1 and increases the failure counter by 1. After reaching declared value the module displays error code and awaits user action. |
| | | Successful login resets failure counter. |
| **Data type** | - | number |
| **Range** | - | 0....255 |
| **Default value** | - | 0 |
| **Comments** | - | Setting the value to "**0**" results in endless retries |

### 8.2.3.13. Wait time after disconnection

| | | |
|---|---|---|
| **Function** | - | Defines interval (in seconds) before resuming after failed login attempt. |
| **Data type** | - | number |
| **Range** | - | 0....60 [s] |
| **Default value** | - | 5 [s] |
| **Comments** | - | Setting the value to *0* results in immediate retries. |

### 8.2.3.14. Data frame format

| | | |
|---|---|---|
| **Function** | - | This parameter selects data frame type used by module for GPRS communication, and indirectly the operating mode. |
| **Data type** | - | selection list |
| **Range** | - | *Standard* |
| | | Standard mode. Modules communicate using the protocol and transmission protection created by the manufacturer. |
| | | *Proxy* |
| | | mode allowing application in GPRS networks with dynamic IP assignment. This mode requires special communication software running on computer with static public address. Currently not supported. |
| | | *Open* |
| | | Configuration and operating modes as for Standard type frames. The only difference is lack of frame protection and opened UDP frame header format allowing creation of user's own access system. |
| | | *UDPStandard* |
| | | Data is send in form of ModbusRTU command encapsulated in standard UDP data frame. Data reception control is not available when using this data format. |
| **Default value** | - | *Standard* |
| **Comments** | - | More info about data formats employed in MT series can be found in chapter Appendices - Data formats |

### 8.2.3.15. Proxy server IP address

| | | |
|---|---|---|
| **Function** | - | inserts Proxy server IP for selected *Proxy* Data frame format. |
| **Data type** | - | IP address field |
| **Range** | - | 0.0.0.0 - 255.255.255.255 |
| **Default value** | - | 0.0.0.0 |
| **Comments** | - | inserted IP is public static address of communication server serving modules working in GSM/GPRS network with dynamic IP assignment. |

### 8.2.3.16. CRC compatibility

| | | |
|---|---|---|
| **Function** | - | This parameter sets CRC calculation for systems requiring full Modbus RTU compatibility. |

| | | |
|---|---|---|
| **Data type** | - | selection list |
| **Range** | - | *Yes* |
| | | For compatibility of Modbus RTU Slave mode, Modbus RTU Master mode and Modbus RTU Mirror mode with remaining modes, enabling the creation of systems consisting of modules working in transparent modes and Modbus RTU modes. This mode ensures compatibility with MT-DP communication software. |
| | | *No* |
| | | for maintaining compatibility when expanding existent systems operating in MODBUS modes or cooperating with old versions of OPC driver. |
| **Default value** | - | *Yes* |
| **Comments** | - | in new systems, it is recommended to leave the option at default value *Yes* |

## 8.2.4. Authorized numbers

This group holds lists of telephone numbers and IP addresses authorized to communicate with the Module.
Lists form the basis for assignment of privileges for configuring, receiving data and sending commands.
Numbers on lists are the only ones allowed to be used for Rules processing.

## 8.2.4.1. Number of phone numbers

| | | |
|---|---|---|
| **Function** | - | Defines length of phone number list that will receive SMS messages. Each phone number has defined privileges for SMS querying. |
| **Data type** | - | number |
| **Range** | - | 0....32 |
| **Default value** | - | 1 |
| **Comments** | - | The range value defines required volume of phone numbers used in SMS Rules processing. See more in Phone |
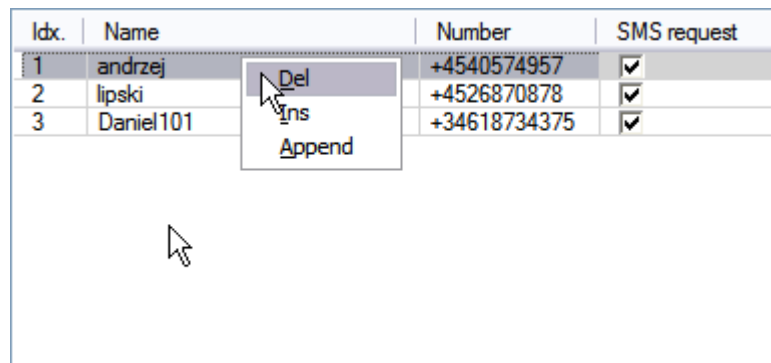
## 8.2.4.2. Number of IP numbers

| | | |
|---|---|---|
| **Function** | - | Defines length of IP numbers list authorized to communicate with the module via GPRS. Particular IP addresses have defined privileges for access to configuration and sending data queries. |
| **Data type** | - | number |
| **Range** | - | 0....128 |
| **Default value** | - | 1 |
| **Comments** | - | The range value defines required volume of IP addresses used in Rules for Data transmission. See more in IP |

### 8.2.4.3. Phone

**Idx**             - list index number

**Name**         - friendly name of the number facilitating identification in Rules processing. Max length - 16 characters.

**Number**      - phone number assigned to index and name. Max 23 characters. Phone number may be a stored in the internal registers which allows dynamic changes of SMS recipient number. Number should be stored in the form of ASCII characters string. Characters are stored only on the younger byte of register. String should be ended with NULL character (0x0000).

**SMS request**  - depending on check mark incoming SMS requests will be processed or ignored

The list may be edited using context menu activated by right mouse click. Available operations depend on the cursor placement. When cursor rests on an entry, all options are available, while only Append is available with cursor resting on active window's background.

| Idx. | Name | Number | SMS request |
|------|------|--------|-------------|
| 1 | andrzej | +4540574957 | ☑ |
| 2 | lipski | +4526870878 | ☑ |
| 3 | Daniel101 | +34618734375 | ☑ |

*(context menu: Del / Ins / Append)*

### 8.2.4.4. IP

**Idx**             - list index number

**Name**         - friendly name of the IP number facilitating identification in Rules processing. Max length - 16 characters.

**Number**      - number IP assigned to index and Name

**Configuration**  - grants or denies right to perform remote configuration by this IP number

**Receiving**    - depending on this setting, data incoming from this IP will be accepted or rejected

The list may be edited using context menu activated by right mouse click. Available operations depend on the cursor placement. When cursor rests on an entry, all options are available, while only Append is available with cursor resting on active window's background.

## 8.2.5. Mode of operation

Group **Mode of operation** is not present on the list of parameter groups available for MT-202 configuration.

It is only present in this manual in order to collect parameters accessible dependent on selected operating Mode of operation.

In practice, only the group of parameters relevant for the selected mode will appear on the list.

### 8.2.5.1. Transparent mode

MT-202 operating in Transparent mode sends all data received at PORT2 to IP addresses appointed during configuration.

This is the standard operating mode employed as autonomic gateway between non-intelligent device connected to serial port using not supported by MT-202 protocol.

#### 8.2.5.1.1. GPRS transmission

In transparent mode, the module receiving data at serial port buffers it in internal memory and converts it to data packet to be sent to defined recipients. During packets creation, criteria described by variables of **GPRS transmission** sub group are applied. Additionally, parameters allowing packet routing in order to reduce transmitted data volume are defined.

Packet routing is possible only when the place where the address identifying final recipient in a disclosed mode is known.

##### 8.2.5.1.1.1. Max. length of data packet

| | | |
|---|---|---|
| **Function** | - | Defines (in bytes) maximal data volume in a packet. When data volume in receiving buffer reaches declared size, packet is sent. |
| **Data type** | - | number |
| **Range** | - | 1....1408 |
| **Default value** | | 256 |
| **Comments** | - | n/a |

##### 8.2.5.1.1.2. Data packet delimiter

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) the time between received characters. Reaching max. value triggers transmission of received data. |
| **Data type** | - | number |

| | | |
|---|---|---|
| **Range** | - | 0,00....655,35 [s] |
| **Default value** | | 1 [s] |
| **Comments** | - | n/a |

### 8.2.5.1.1.3. Channel reservation time

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) the time of reservation of exclusive transmission channel to device from which the frame was received. |
| **Data type** | - | number |
| **Range** | - | 0,00....655,35 [s] |
| **Default value** | | 0 [s] |
| **Comments** | - | This parameter helps setting an exclusive transmission channel to one of remote devices for a period of time. This enables establishment of a system consisting of several devices with functionalities similar to point-to-point. |
| | | When a channel is established, module ignores packets received from other devices and all data received at PORT 2 are sent to the module with which the channel is established. Transmission costs are thus reduced significantly. Packets are sent to all recipients only at first transmission. |
| | | The function is activated by setting the value other to than zero thus defining the time channel is reserved. Time count starts from receiving first GPRS packet. |
| | | Consecutive packets from same source prolong reservation time while sent packets do not. |
| | | Setting this variable to 0 switches reservation off and module operates in standard way sending and receiving packets to all defined recipients. |

### 8.2.5.1.1.4. Routing

| | | |
|---|---|---|
| **Function** | - | enables choosing method of recording recipients address in transmitted data frame. |
| **Data type** | - | number |
| **Range** | - | *n/a*<br>Transparent mode without routing |
| | | *8 bit Address*<br>The Address is 8 bit long (one byte) |
| | | *16 bit HL Address*<br>The Address is 16 bit long (two bytes) in sequence „High" and „Low" byte |
| | | *16 bit LH Address*<br>The Address is 16 bit long (two bytes) in sequence „Low" and „High" byte |
| **Default value** | | *n/a* |
| **Comments** | - | setting the parameter to *n/a* switches routing off |

8.2.5.1.1.5. Address offset

| | | |
|---|---|---|
| **Function** | - | defines the offset of address field in transmitted data frame |
| **Data type** | - | number |
| **Range** | - | 0....1407 |
| **Default value** | | 0 |
| **Comments** | - | too short frames with lengths not encompassing address field which are ignored. Pay attention to proper setting of Max. length of data packet and Data packet delimiter so that all data frames will be received completely. |

8.2.5.1.1.6. Broadcast address

| | | |
|---|---|---|
| **Function** | - | Defines recipient address. If detected in address field of the frame received at PORT 2 will cause sending of this frame to all recipients defined in routing table. |
| **Data type** | - | number |
| **Range** | - | 0....65535 |
| **Default value** | | 0 |
| **Comments** | - | Placing this address in routing table will result in sending all transmitted frames to corresponding IP address. |

8.2.5.1.1.7. Routing table size

| | | |
|---|---|---|
| **Function** | - | Defines size of routing table |
| **Data type** | - | number |
| **Range** | - | 1....256 |
| **Default value** | - | 1 |
| **Comments** | - | the length of routing table should not be shorter than number of network nodes the module communicates with. |

8.2.5.1.2. Routing table

| | | |
|---|---|---|
| **Idx** | - | index number of the list |
| **number IP** | - | IP address of GPRS network node |
| **number ID** | - | ID number of device connected to GPRS network node with given IP address |
| **Comments** | - | n/a |

### 8.2.5.2. Modbus RTU Master mode

MT-202 Module operating in **Modbus RTU Master** mode receives data sent by peripheral device operating in Modbus RTU Master mode at PORT2. Data are sent via GPRS to remote Slave devices according to routing table.

Module MT-202 operating in this mode acts as intelligent GPRS communication gateway allowing physical separation of Modbus RTU network nodes. Module's internal resources have assigned Modbus ID which allows reading them from PORT2.

#### 8.2.5.2.1. Routing table size

| | | |
|---|---|---|
| **Function** | - | Defines size of routing table |
| **Data type** | - | number |
| **Range** | - | 1....255 |
| **Default value** | - | 1 |
| **Comments** | - | the length of routing table should not be shorter than number of network nodes the module communicates with. |

#### 8.2.5.2.2. Routing table

| | | |
|---|---|---|
| **Idx** | - | index number of the list |
| **number IP** | - | IP address of GPRS network node |
| **number ID** | - | Modbus ID number (0-255) of slave device connected to GPRS network node with given IP address |
| **Comments** | - | entering Modbus ID of one of the Slave devices connected to PORT2 (including internal module resources) results in sending replies from that ID to specified IP address regardless of the address of inquiring unit. Setting this value to *0 (zero)*, makes the module with specified IP address receive all frames sent by Master unit regardless of Modbus addressing. |

### 8.2.5.3. Modbus RTU Slave mode

MT-202 Module operating in **Modbus RTU Slave** mode sends all received via GPRS queries to PORT2, except those that have Modbus IDs identical with module's own Modbus ID of internal resources in address field - since they are handled locally.
replies received at PORT 2 from peripheral Slave devices along with replies from internal module resources are sent to the address from which query was issued or according to Routing Table content.

### 8.2.5.3.1. Routing table size

| | | |
|---|---|---|
| **Function** | - | Defines size of routing table |
| **Data type** | - | number |
| **Range** | - | 1....255 |
| **Default value** | - | 1 |
| **Comments** | - | the length of routing table should not be shorter than number of network nodes the module communicates with. |

### 8.2.5.3.2. Routing table

| | | |
|---|---|---|
| **Idx** | - | index number of the list |
| **number IP** | - | IP address of GPRS network node |
| **number ID** | - | Modbus ID number (0-255) of slave device connected to GPRS network node with given IP address |
| **Comments** | - | In basic configuration where replies are sent only to inquirer the table may remain empty. |
| | | The table needs to be filled out when replies have to be sent to other Masters. Second column holds IP address of auxiliary master, while third column - Modbus ID of device which answer has to be sent additionally to auxiliary master. If third column has value *0 (zero)*, replies from all connected to module slaves and module's internal resources will be sent to appointed IP address. Sending replies to any number of Master devices is possible (Multimaster mode ). |

## 8.2.5.4. Modbus RTU Mirror mode

**Modbus RTU Mirror** is an extension of Modbus RTU Slave mode. The behavior of the module is identical to Modbus RTU Slave mode seen from the GPRS point of view. Remote access to internal module resources and to Slave devices connected to PORT 2 is secured. An additional feature is the opportunity to map resources of slave devices connected to PORT 2 into internal module registers. Upon activation of this option, the module cyclically reads mapped areas and refreshes internal registers.

### 8.2.5.4.1. Number of data blocks

| | | |
|---|---|---|
| **Function** | - | Defines number of registers read from peripheral Slave devices via PORT2 |
| **Data type** | - | number |
| **Range** | - | 1....16 |
| **Default value** | - | 1 |
| **Comments** | - | n/a |

8.2.5.4.2. Delay after error in communication with SLAVE

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) delay before reestablishing reading of registers in peripheral Slave devices after transmission errors. Time is counted independently for each defined area (Slave device) and reduces only the frequency of querying for areas with communication errors. |
| **Data type** | - | number |
| **Range** | - | 1....65535 [s] |
| **Default value** | | 15 [s] |
| **Comments** | - | Increasing this value reduces influence of malfunctioning devices on communication with other devices but decreases statistic time to reestablish communication after removal of error cause. |

8.2.5.4.3. Data block 1...16

The table defining sequential data blocks read from attached to PORT2 peripheral Slave devices allows defining of 1 to 16 independent blocks in one or more devices. Consecutive blocks may encompass registers from different spaces and be refreshed at different intervals.

8.2.5.4.3.1. Modbus ID of Slave device

| | | |
|---|---|---|
| **Function** | - | Defines Modbus ID of Slave device from which configured data block is to be read |
| **Data type** | - | number |
| **Range** | - | 0....255 |
| **Default value** | - | 0 |
| **Comments** | - | setting Modbus ID to *0* (*zero*) switches the mapping off |

8.2.5.4.3.2. Space

| | | |
|---|---|---|
| **Function** | - | selects Modbus space mapped from peripheral Slave device attached to PORT2 |
| **Data type** | - | selection list |
| **Range** | - | *Bi In*<br>    Binary inputs (1xxx), read-only<br>*Bi Out*<br>    Binary outputs (0xxx), read and write<br>*Inp. Reg.*<br>    Input Registers (3xxx), read-only<br>*Hold. Reg.*<br>    Internal Registers (4xxx), read and write |
| **Default value** | - | *Bi In* |
| **Comments** | - | writing into registers mapping read-only spaces does not cause an error. Values will be replaced by next error-free reading from peripheral device. |

### 8.2.5.4.3.3. Address of mapped space in module

| | | |
|---|---|---|
| **Function** | - | defines start address of internal register in the module, used for mapping space from the Slave. For mapping space Internal registers are always used. |
| **Data type** | - | number |
| **Range** | - | 0....999 |
| **Default value** | - | 1 |
| **Comments** | - | n/a |

### 8.2.5.4.3.4. Address of mapped space in SLAVE

| | | |
|---|---|---|
| **Function** | - | Defines address of first register mapped from peripheral Slave device. |
| **Data type** | - | number |
| **Range** | - | 0....65535 |
| **Default value** | - | 0 |
| **Comments** | - | n/a |

### 8.2.5.4.3.5. Mapped space size

| | | |
|---|---|---|
| **Function** | - | Defines the size of register space necessary for reading mapped space from peripheral Slave device. |
| **Data type** | - | number |
| **Range** | - | 1....2040 |
| **Default value** | - | 1 |
| **Comments** | - | for register space, value of this variable defines size of mapped space in Registers while for bitmapped in bits. Bits from mapped space are placed on consecutive bits in registers (starting from least significant). So, in module in one registers 16 bits are stored. Typical layout of mapped bit space:<br>**Address of mapped space module**: *64*<br>**Address of mapped space SLAVE**: *3*<br>**Size of mapped space**: *20* |

### 8.2.5.4.3.6. Mapped space read interval

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) read interval of peripheral Slave device for update. |
| **Data type** | - | number |
| **Range** | - | 0....65535 [s] |
| **Default value** | - | 1 [s] |
| **Comments** | - | value *0* (*zero*) forces max. possible frequency of updating. It depends on speed of PORT2 along with size and number of defined mapped spaces. |

8.2.5.4.3.7. Force write the whole block

| | | |
|---|---|---|
| **Function** | - | Parameter allows forcing whole data block writes |
| **Data type** | - | selection list |
| **Range** | - | *Yes*<br>Block writing is forced - any change of register or bit on side of module forces write of whole data block, not single register or bit. It is advised to turn this option on when using variables longer than 16 bits (long, float)<br>*No*<br>Method of writes depends on changes of resources that require writing to external device. It can be hole data blocks or single bits/register. |
| **Default value** | - | *No* |
| **Comments** | - | This parameter is available only if mirrored resources (external device resources) are stored in Bi Out or Holding Registers address space (defined by parameter Space). |

## 8.2.5.5. GazModem mode

<table>
<tr><td style="text-align:center"><strong>CAUTION!</strong><br><strong>Option discontinued from v 1.43 of firmware for module MT-202.</strong><br><strong>The description is solely for legacy support.</strong></td></tr>
</table>

MT-101 Module operating in **GazModem mode** enables communication with gas counters operating on GazModem transmission protocol. Data received at PORT2 are placed in respective registers of MT-202 module's memory map.

8.2.5.5.1. Read interval

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) an interval between readings from peripheral device operating on GazModem protocol in order to update measurement data. |
| **Data type** | - | number |
| **Range** | - | 0....65535 [s] |
| **Default value** | - | 180 [s] |
| **Comments** | - | value *0* (*zero*) forces max. possible frequency of updating. It depends on speed of PORT2 along with size and number of defined mapped spaces. |

8.2.5.5.2. Number of retries

| | | |
|---|---|---|
| **Function** | - | defines number of attempts to transmit data via PORT2 in case of unconfirmed (within time defined by transmission Timeout parameter) reception. |
| **Data type** | - | number |
| **Range** | - | 0....7 |
| **Default value** | - | 2 |
| **Comments** | - | value *0* (*zero*) forces max. possible frequency of updating. It depends on speed of PORT2 along with size and number of defined mapped spaces. |

### 8.2.5.5.3. Transmission timeout

| | |
|---|---|
| **Function** | - Defines (in seconds) waiting time for confirmation of data frame reception. |
| **Data type** | - number |
| **Range** | - 0....10 [s] |
| **Default value** | - 2 [s] |
| **Comments** | - This value along with declared Number of retries influences max. time of single data packet transmission. |

### 8.2.5.5.4. Threshold hysteresis

| | |
|---|---|
| **Function** | - Defines hysteresis for alarm thresholds values |
| **Data type** | - number |
| **Range** | - 0,01...100,00 [%] |
| **Default value** | - 2,00 [%] |
| **Comments** | - setting of value of hysteresis adequate to signal variations prevents untimely activations of alarm flags. |

### 8.2.5.5.5. Alarm station IP address

| | |
|---|---|
| **Function** | - defines alarm recipient's IP address. |
| **Data type** | - selection list |
| **Range** | - friendly names assigned to IP addresses of devices declared in Authorized numbers list |
| **Default value** | - IP1 - first number on authorized numbers list |
| **Comments** | - since recipient's IP address is selected from selection list, it is practical to define unique unambiguous names facilitating identification on the list. |

### 8.2.5.5.6. Number of devices

| | |
|---|---|
| **Function** | - Defines number of declared register spaces (devices) read on PORT2 from peripheral devices operating on GazModem protocol |
| **Data type** | - number |
| **Range** | - 1....16 |
| **Default value** | - 1 |
| **Comments** | - n/a |

### 8.2.5.5.7. MC 1....16

The table defining consecutive data blocks read from peripheral devices attached to PORT2 operating in GazModem protocol. The table can hold from 1 to 16 independent blocks in one or more devices. Consecutive blocks may encompass Registers from different spaces.

### 8.2.5.5.7.1. Address

| | | |
|---|---|---|
| **Function** | - | Defines address of gas counter from which the configured data block should be read |
| **Data type** | - | number |
| **Range** | - | 0....65535 |
| **Default value** | - | 0 |
| **Comments** | - | n/a |

### 8.2.5.5.7.2. Alarm reading

| | | |
|---|---|---|
| **Function** | - | defines status of alarm reading from gas meter |
| **Data type** | - | selection list |
| **Range** | - | *Yes*<br>　　Reading active<br>*No*<br>　　Reading inactive |
| **Default value** | - | *No* |
| **Comments** | - | n/a |

### 8.2.5.5.7.3. Signal reading

| | | |
|---|---|---|
| **Function** | - | defines status of signal reading from gas meter |
| **Data type** | - | selection list |
| **Range** | - | *No*<br>　　Signal reading inactive<br>*1 byte*<br>　　Read 1 signal byte in Gaz-Modem protocol<br>*2 bytes*<br>　　Read 2 signal bytes in Gaz-Modem protocol<br>*3 bytes*<br>　　Read 3 signal bytes in Gaz-Modem protocol<br>*4 bytes*<br>　　Read 4 signal bytes in Gaz-Modem protocol<br>*Gazmodem 2*<br>　　Read signals in Gaz-Modem2 protocol |
| **Default value** | - | *No* |
| **Comments** | - | n/a |

### 8.2.5.5.7.4. Current data reading

| | | |
|---|---|---|
| **Function** | - | sets status of current data reading from gas meter |
| **Data type** | - | selection list |
| **Range** | - | *Yes*<br>　　Current data reading active<br>*No*<br>　　Current data reading inactive |
| **Default value** | - | *No* |
| **Comments** | - | n/a |

8.2.5.5.7.5. Current data block index

| | | |
|---|---|---|
| **Function** | - | Defines address of first current data register from peripheral GazModem device |
| **Data type** | - | number |
| **Range** | - | 0....64 |
| **Default value** | - | 0 |
| **Comments** | - | n/a |

8.2.5.5.7.6. Current data block length

| | | |
|---|---|---|
| **Function** | - | Defines size of current data block read from peripheral GazModem device |
| **Data type** | - | number |
| **Range** | - | 1....8 |
| **Default value** | - | 1 |
| **Comments** | - | n/a |

## 8.2.5.6. M-Bus LEC mode

<div style="border:1px solid black; background:yellow; text-align:center; color:red; font-weight:bold;">

**CAUTION!**
**This option is not supported since firmware 1.43 in MT-101 modules.**
**The following description is published for legacy support.**

</div>

MT-202 module operating in **M-Bus LEC mode** along with converter RM-102 plays the role of data converter for data received from heat meters that employ M-Bus protocol. Data received at PORT2 is placed in dedicated registers of the memory map of MT-202. Additionally, in this mode it is possible to attach a gas meter to PORT1 of the MT-202 module. After selecting this mode, local configuration is possible during the first 30s after power-up - PWR LED is flashing.

8.2.5.6.1. Read interval

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) an interval between readings from peripheral device operating on GazModem protocol in order to update measurement data. |
| **Data type** | - | number |
| **Range** | - | 0....255 [min] |
| **Default value** | - | 5 [min] |
| **Comments** | - | value *0* (*zero*) forces max. possible frequency of updating. It depends on speed of PORT2 along with size and number of defined mapped spaces. |

8.2.5.6.2. Number of retries

| | | |
|---|---|---|
| **Function** | - | defines number of attempts to transmit data via PORT2 in case of unconfirmed (within time defined by transmission Timeout parameter) reception. |

| **Data type** | - number |
|---|---|
| **Range** | - 0....7 |
| **Default value** | - 1 |
| **Comments** | - value 0 results in sending without confirming faultless reception. |

### 8.2.5.6.3. Transmission timeout

| **Function** | - Defines (in seconds) waiting time for confirmation of data frame reception. |
|---|---|
| **Data type** | - number |
| **Range** | - 0....10 [s] |
| **Default value** | - 2 [s] |
| **Comments** | - This value along with declared Number of retries influences max. time of single data packet transmission. |

### 8.2.5.6.4. Threshold hysteresis

| **Function** | - Defines hysteresis for alarm thresholds values |
|---|---|
| **Data type** | - number |
| **Range** | - 0,01...100,00 [%] |
| **Default value** | - 5,00 [%] |
| **Comments** | - setting of value of hysteresis adequate to signal variations prevents untimely activations of alarm flags. |

### 8.2.5.6.5. Gas meter address

| **Function** | - Defines address of gas meter connected to module's PORT1 |
|---|---|
| **Data type** | - number |
| **Range** | - 0....65535 |
| **Default value** | - 0 |
| **Comments** | - n/a |

### 8.2.5.6.6. Gas meter reading interval

| **Function** | - Defines (seconds) interval of readings from peripheral device using GazModem protocol for update measurement data. The connection is via PORT1 |
|---|---|
| **Data type** | - number |
| **Range** | - 0....65535 [s] |
| **Default value** | - 180 [s] |
| **Comments** | - value *0* (*zero*) forces max. possible frequency of updating. It depends on speed of PORT2 along with size and number of defined mapped spaces. |

### 8.2.5.6.7. Number of transmission retries to gas meter

| | |
|---|---|
| **Function** | - defines number of attempts to transmit data via PORT1 in case of unconfirmed (within time defined by Transmission timeout to gas meter parameter) reception. |
| **Data type** | - number |
| **Range** | - 0....7 |
| **Default value** | - 2 |
| **Comments** | - value 0 results in sending without confirming faultless reception. |

### 8.2.5.6.8. Transmission timeout for gas meter

| | |
|---|---|
| **Function** | - Defines (in seconds) waiting time for confirmation of data frame reception at PORT1. |
| **Data type** | - number |
| **Range** | - 0,01....10,00 [s] |
| **Default value** | - 2,00 [s] |
| **Comments** | - This value along with declared Number of transmission retries to gas meter influences max. time of single data packet transmission. |

### 8.2.5.6.9. Number of devices

| | |
|---|---|
| **Function** | - Defines number of declared register spaces (devices) read on PORT2 from peripheral devices operating on M-Bus protocol. |
| **Data type** | - number |
| **Range** | - 1....16 |
| **Default value** | - 1 |
| **Comments** | - n/a |

### 8.2.5.6.10. MC

The table defining consecutive data blocks read from peripheral devices attached to PORT2 operating in M-Bus protocol. The table can hold from 1 to 16 independent blocks in one or more devices.

#### 8.2.5.6.10.1. Address 1....16

| | |
|---|---|
| **Function** | - Defines address of heat meter whose internal parameters shall be read. |
| **Data type** | - number |
| **Range** | - 0....255 |
| **Default value** | - 0 |
| **Comments** | - address 255 means querying stopped |

### 8.2.5.6.10.2. Geographical coordinates format

| | | |
|---|---|---|
| **Function** | - | Defines format of geographical coordinates received in NMEA 0183 protocol when writing into registers. |
| **Data type** | - | selection list |
| **Range** | - | *Degrees.minutes (DDMM.mmmmm)*<br>    Position registered in format<br>*Degrees (DD.dddddd)*<br>    Position registered in format |
| **Default value** | | *Degrees.minutes (SSMM.mmmmm)* |
| **Comments** | - | n/a |

### 8.2.5.6.10.3. Identifier 1....16

| | | |
|---|---|---|
| **Function** | - | Defines additional number of heat meter whose internal parameters shall be read. |
| **Data type** | - | number |
| **Range** | - | 0....999999999 |
| **Default value** | - | 0 |
| **Comments** | - | Parameter enables addition of auxiliary identifier for heat meter in telemetry system. The parameter is in no way connected to real address of meter. The value of parameter is stored in MT module and can be read from its memory map. |

## 8.2.5.7. NMEA 0183 mode

Module MT-202 operating in **NMEA 0183** mode receives data from peripheral devices using communication protocol compliant with NMEA 0183 at PORT2. The module recognizes data received from AIRMAR weather station and places it in Internal registers. Read more about NMEA 0183 mode...

### 8.2.5.7.1. Data validity time

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) validity time of parameters read from NMEA frames received at PORT2. When exceeded due to incomplete reception, relevant bits signal that values in registers are out of date. In other words, older than the time declared in configuration. |
| **Data type** | - | number |
| **Range** | - | 1....300 |
| **Default value** | | 2 |
| **Comments** | - | n/a |

## 8.2.5.8. FlexSerial

In **FlexSerial mode** MT-202 module can service units connected to PORT2 that employ non standard protocols.

### 8.2.5.8.1. Max. length of data packet

| | | |
|---|---|---|
| **Function** | - | defines (in bytes) max length of data packet in receiving buffer. |
| **Data type** | - | number |
| **Range** | - | 1....512 |
| **Default value** | | 256 |
| **Comments** | - | none |

### 8.2.5.8.2. Data packet delimiter

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) interval between incoming bytes. Exceeding this interval results in storing incomplete data packet in receiving buffer |
| **Data type** | - | number |
| **Range** | - | 0,00....655,35 [s] |
| **Default value** | | 1 [s] |
| **Comments** | - | none |

## 8.2.6. Resources

Group **Resources** encompasses a list of hardware and software resources available to users. Sub-groups hold configurable parameters for asynchronous and synchronous Timers, MT2MT Buffer and Constant parameters.

### 8.2.6.1. Modbus ID number of module's internal resources

| | | |
|---|---|---|
| **Function** | - | Defines Modbus ID number for internal resources of the module operating in Modbus Slave mode. |
| **Data type** | - | number |
| **Range** | - | 0....255 |
| **Default value** | - | 1 |
| **Comments** | - | setting Modbus ID to *0* (*zero*) makes access to internal module resources impossible |

### 8.2.6.2. Serial port

Sub-group serial port holds configuration parameters for PORT2.

PORT1 parameters are unchangeable  (Speed: 9600, 8 bits, no parity, 1 stop bit , flow control: hardware (RTS/CTS).

#### 8.2.6.2.1. Interface type

| | | |
|---|---|---|
| **Function** | - | defines electrical standard for serial port |
| **Data type** | - | selection list |
| **Range** | - | *RS232* <br> voltage interface ±12V, full duplex, three wire, <br> *RS422* <br> differential interface, full duplex, double pair, |

*RS485*
differential interface, half duplex, single pair.

| **Default value** | - | *RS232* |
|---|---|---|
| **Comments** | - | n/a |

### 8.2.6.2.2. Transmission speed

| **Function** | - | defines transmission speed ( bits/s) for serial port |
|---|---|---|
| **Data type** | - | selection list |
| **Range** | - | *1200, 2400, 4800, 9600,19200, 38400 [b/s]*<br>List of supported speeds |
| **Default value** | - | *9600 [b/sec]* |
| **Comments** | - | n/a |

### 8.2.6.2.3. Stop bits

| **Function** | - | defines number of stop bits |
|---|---|---|
| **Data type** | - | selection list |
| **Range** | - | *1, 2*<br>List of options |
| **Default value** | - | *1* |
| **Comments** | - | Function has no influence on transmission parameters for Modbus modes. Number of stop bits depends on selected parity control mode. |

### 8.2.6.2.4. Parity

| **Function** | - | defines control of transmitted byte |
|---|---|---|
| **Data type** | - | selection list |
| **Range** | - | *n/a*      *(None)*<br>*Even*<br>*Odd*<br>List of available options |
| **Default value** | - | *n/a* |
| **Comments** | - | For operating modes for ports with MODBUS protocol, this setting has following influence on stop bits:<br>1 stop bit<br>    for *Even* and *Odd*,<br>2 stop bits<br>    for *n/a*. |

### 8.2.6.3. Asynchronous clocks

Two **Asynchronous clocks** can cyclically count time for up to days (8640000 s). Counting starts immediately after module starts up and goes on until switched off. Asynchronous clocks have two Triggering outputs T1, T2, that can be used for rules processing.

### 8.2.6.3.1. Clocks TMR1, TMR2

#### 8.2.6.3.1.1. Period

| | | |
|---|---|---|
| **Function** | - | Defines (in seconds) asynchronous timer counting period |
| **Data type** | - | number |
| **Range** | - | 0....8640000 [s] |
| **Default value** | - | 0 [s] |
| **Comments** | - | *0* (*zero*) value switches the clock off |

## 8.2.6.4. Synchronous clocks

**Synchronous clocks** group contains parameters set for two clocks capable of cooperating with module real time (RTC) clock thus enabling triggering of events synchronized with defined time.

### 8.2.6.4.1. Clock TMR3, TMR4

#### 8.2.6.4.1.1. Start

| | | |
|---|---|---|
| **Function** | - | synchronizes timer's clock setting start point and counting period. |
| **Data type** | - | time [HH:mm] |
| **Range** | - | 0:00 - 23:59 |
| **Default value** | - | 0:00 |
| **Comments** | - | n/a |

#### 8.2.6.4.1.2. Period

| | | |
|---|---|---|
| **Function** | - | defines **synchronous clock** counting period in minutes. |
| **Data type** | - | number |
| **Range** | - | *0…1440 [min]* |
| **Default value** | - | *0 [min]* |
| **Comments** | - | *0* (*zero*) value switches the clock off |

## 8.2.6.5. MT2MT Buffer

**MT2MT Buffer** enables creation of system where modules may exchange information (internal registers) with each other. Using buffer requires activation and defining register space where exchange is going to take place. More in chapter Internal Resources/MT2MT Buffer.

### 8.2.6.5.1. Active

| | | |
|---|---|---|
| **Function** | - | defines state of employing MT2MT Buffer |
| **Data type** | - | selection list |
| **Range** | - | *Yes* |
| | | MT2MT Buffer active |
| | | *No* |
| | | MT2MT Buffer inactive |

| **Default value** | - *No* |
| **Comments** | - n/a |

### 8.2.6.5.2. Sending to PORT2

| **Function** | - Defines whether received event data buffer has to be sent to PORT2 |
| **Data type** | - selection list |
| **Range** | - *Yes* |
| | Received data will be sent |
| | *No* |
| | Received data will not be sent |
| **Default value** | - *No* |
| **Comments** | - This parameter has no influence on events with status that are always sent to PORT2 |

### 8.2.6.5.3. Buffer address

| **Function** | - Defines start address of internal register space used for events reception. |
| **Data type** | - number |
| **Range** | - 0....999 |
| **Default value** | - 64 |
| **Comments** | - received events registers laying outside defined space are not copied. |

### 8.2.6.5.4. Buffer size

| **Function** | - Defines the size of internal registers space used for events reception. |
| **Data type** | - number |
| **Range** | - 1....512 |
| **Default value** | - 16 |
| **Comments** | - received events registers laying outside defined space are not copied. |

## 8.2.6.6. Constant parameters

An option of defining Constant parameters under configuration was added for the user's convenience. Parameters are loaded to module memory during initialization of the module. More in chapter Internal Resources/Parameters.

### 8.2.6.6.1. Number of parameters

| **Function** | - Defines number of available constant parameters |
| **Data type** | - number |

| Range | - 0....128 |
|---|---|
| **Default value** | - 0 |
| **Comments** | - n/a |

### 8.2.6.6.2. Parameter 1...128

Consecutive parameters are defined as numbers ranging from 0....65535.

## 8.2.7. Rules

Rules group contains lists of transmission tasks performed by internal program when criteria defined in rules are met. Tasks are divided into two groups:

- rules concerning transmission of SMS messages
- rules concerning transmission of data

In both cases, the criteria are defined by using same resources and conditions of application of the rule.

## 8.2.7.1. SMS sending

List of SMS sending rules can hold max. 32 entries defining short text message transmission conditions. Adding a new position is done in the context menu by right-clicking mouse while one of positions on the list is highlighted.
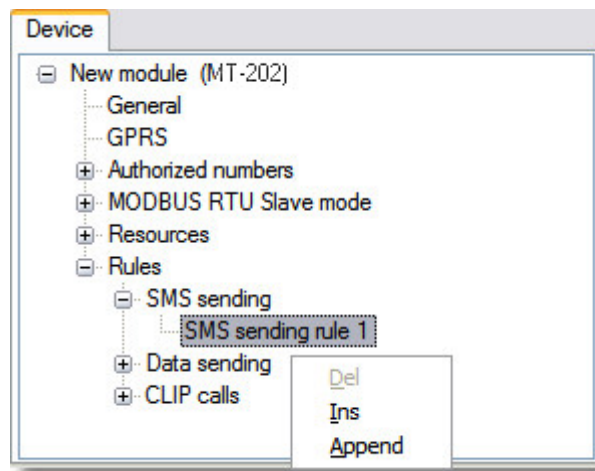


Adding more rules is done by setting the parameter number of SMS sending rules to desired value.

### 8.2.7.1.1. Number of SMS sending rules

| **Function** | - declares number of SMS sending rules |
|---|---|
| **Data type** | - number |
| **Range** | - 1....32 |
| **Default value** | - 1 |
| **Comments** | - diminishing the number of rules does not delete settings until the configuration is written to the module. |

8.2.7.1.2. SMS sending rule

Each of the rules residing on the list is defined by following parameters:

- Trigger input
- Trigger flag
- SMS text
- Recipient number

8.2.7.1.2.1. Trigger input

| | | |
|---|---|---|
| **Function** | - | defines resource to observe |
| **Data type** | - | selection list |
| **Range** | - | *n/a* |
| | | the rule is inactive |
| | | *FS1_ups, FS1_gprs* |
| | | system trigger inputs |
| | | *P1…P32* |
| | | user program inputs |
| | | *TMR1, TMR2, TMR3, TMR4* |
| | | synchronous and asynchronous clocks trigger inputs |
| **Default value** | - | *n/a* |
| **Comments** | - | more about trigger inputs and flags in chapter Appendices |

8.2.7.1.2.2. Trigger flag

| | | |
|---|---|---|
| **Function** | - | defines event-triggering flag associated with selected trigger input |
| **Data type** | - | selection list |
| **Range** | - | *n/a* |
| | | rule inactive |
| | | *Bi In 0->1, Bi In 1->0 Bi In Chg* |
| | | binary input state change |
| **Default value** | - | *n/a* |
| **Comments** | - | more about trigger inputs and flags in chapter Appendices |

8.2.7.1.2.3. SMS text

| | | |
|---|---|---|
| **Function** | - | allows entering text sent in message triggered by defined rule |
| **Data type** | - | text |
| **Range** | - | letters, numerals, special characters - max. length: 160 characters |
| **Default value** | - | . |
| **Comments** | - | if the message will include the status of the module, total length of SMS text and the status may not exceed 160 characters. If the length is greater, the text will be truncated so that full status is sent. |

### 8.2.7.1.2.4. Recipient number

| | | |
|---|---|---|
| **Function** | - | selects message recipient number |
| **Data type** | - | selection list |
| **Range** | - | friendly names of recipients associated with phone numbers in Authorized numbers list |
| **Default value** | - | NUM 1 - first number on the list |
| **Comments** | - | since recipient's phone number is selected from the list of friendly names, it is important to enter unique, unambiguous names facilitating identification. |

### 8.2.7.1.2.5. Attach timestamp

| | | |
|---|---|---|
| **Function** | - | selects whether module timestamp is to be attached to the message. |
| **Data type** | - | List |
| **Range** | - | *Yes*<br>    Timestamp attached to SMS<br>*No*<br>    Only SMS text is send |
| **Default value** | - | *Yes* |
| **Comments** | - | Total length of SMS text and additional information may not exceed 160 characters. If the length is greater, the text defined by user will be truncated so that additional information will be sent. |

## 8.2.7.2. Data sending

List of SMS sending rules can hold max. 32 entries defining data transmission conditions. A defined data block will be sent to appointed IP address. Adding a new position is done in the context menu by right-clicking mouse while one of positions on the list is highlighted.
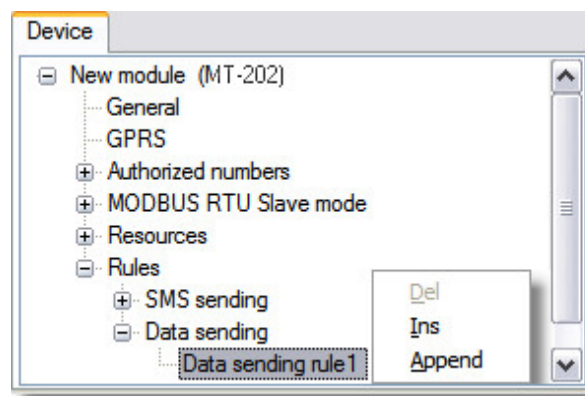


Adding more rules is done by setting the parameter number of data sending rules to desired value.

### 8.2.7.2.1. Number of data sending rules

| | | |
|---|---|---|
| **Function** | - | declares number of SMS sending rules |
| **Data type** | - | number |
| **Range** | - | 1....32 |
| **Default value** | - | 1 |
| **Comments** | - | diminishing the number of rules does not delete settings until the configuration is written to the module. |

### 8.2.7.2.1.1. Data sending rule

Each of rules residing on the list is defined by following parameters:

- Trigger input
- Trigger flag
- IP Address
- Send
- Buffer address
- Buffer size

### 8.2.7.2.1.1.1. Trigger input

| | | |
|---|---|---|
| **Function** | - | defines resource to observe |
| **Data type** | - | selection list |
| **Range** | - | *n/a*<br>    the rule is inactive<br>*FS1_ups, FS1_gprs*<br>    system trigger inputs<br>*P1…P32*<br>    user program inputs<br>*TMR1, TMR2, TMR3, TMR4*<br>    synchronous and asynchronous clocks trigger inputs |
| **Default value** | - | *n/a* |
| **Comments** | - | more about trigger inputs and flags in chapter Appendices |

### 8.2.7.2.1.1.2. Trigger flag

| | | |
|---|---|---|
| **Function** | - | defines event triggering flag associated with selected trigger input |
| **Data type** | - | selection list |
| **Range** | - | *n/a*<br>    rule inactive<br>*Bi In 0->1, Bi In 1->0 Bi In Chg*<br>    binary input state change |
| **Default value** | - | *n/a* |
| **Comments** | - | more about trigger inputs and flags in chapter Appendices |

8.2.7.2.1.1.3. IP address

**Function** - selects recipient's IP address

**Data type** - selection list

**Range** - friendly names of recipients associated with IP addresses in Authorized numbers list

**Default value** - IP1 - first number on the list

**Comments** - since recipient's IP address is selected from the list of friendly names, it is important to enter unique, unambiguous names facilitating identification.


8.2.7.2.1.1.4. Send

**Function** - defines data type sent in transmission triggered by defined rule

**Data type** - List

**Range** - *Buffer Hold. Reg.*
Registers from modules' internal registers' space will be sent. Defining the transmitted space is required.
*Buffer Inp. Reg.*
Registers from module input registers' space will be sent. Defining the transmitted space is required.

**Default value** - *Buffer Hold. Reg.*

**Comments** - n/a


8.2.7.2.1.1.5. Buffer address

**Function** - Defines start address of internal registers space sent in transmission triggered by defined rule

**Data type** - number

**Range** - 0....8191

**Default value** - 64

**Comments** - n/a


8.2.7.2.1.1.6. Buffer size

**Function** - Defines size of internal registers space sent in transmission triggered by defined rule

**Data type** - number

**Range** - 1....700

**Default value** - 16

**Comments** - n/a

### 8.3. Configuration writing

After required modifications and parameter settings, the configuration is stored on the configuring PC's hard disk only. In order to write it to the module memory, it has to be transmitted to the module .

The method of transmission depends on whether we configure it locally or remotely via GPRS. For local configuration, it is enough to secure a connection via RS-232 cable.
Detailed description of local configuration is to be found in the MTManager user manual.

For remote configuration, it is vital that the computer running the configuration application has access to the APN where the configured module resides. Detailed description of remote configuration is to be found in the MTM user manual.

### 8.4. Verification of configuration

Despite high reliability of both local and remote module configuration, verify of it is important. It is relevant if the module's behavior does not comply in accordance with the performed configuration. For verification, please read the configuration from the module and check parameters settings. Reading of module configuration is described in details in MTM users manual.

## 9. Programming

### 9.1. General information

Modules from the MT-10x and MT-202 series and EX-101 expansion allow downloading user-defined internal programs, thereby expanding module functionality with non-standard algorithms of data processing and module control. Programming is accomplished by using the **MTProg** application delivered free of charge to our customers, giving them the possibility of programming in integrated environments.

Basic information regarding user programs functionality:
- The program is executed cyclically every 100ms.
- If the particular program cycle does not complete execution within 100ms, the next cycle will not start immediately but at the next 100ms round. The omission of the program cycle is signaled by flashing of OVR LED. In such cases, the user program should use RTC register values or clock blocks instead of measuring the time by incrementing a register value for each cycle.
- The user program can consist of max. 1024 instructions.
- Max. number of instructions carried out in one cycle is limited to 2000. Upon reaching this value, the program is automatically disrupted and restarted at the next 100ms round.
- The program is capable of carrying approx. 750 instructions in 100ms.
- The function of copying the buffers copies approx. 1500 registers in 100ms.
- The function of fast copying of blocks copies approx. 7000 registers in 100ms.

### 9.2. Starting to work

It is recommended to set up the working environment during the first run of the program.

For this purpose, select menu item Help/Settings or activate  icon from the Toolbar and the following dialog window will appear. Fill in the relevant data for parameters.

**Modem init file**
Finds and selects the file holding initialization parameters for the GPRS modem used to communicate with remote module.

**APN name**
States name of the APN where programmed module resides.

**PPP user**
Parameter has to be defined only if network operator requires it.

**PPP password**
Parameter has to be defined only if network operator requires it.

**Connect to GPRS**
Parameter has to be defined only if network operator requires it.

**APN server**
IP address of the computer routing data packets sent via internet.

**Internet connection**
When selected in conjunction with "RS-232 port", it sets up the communication via dial up GPRS connection or via routed Ethernet connection.
This is the optimal way of communication between MTProg and remote modules.
When unselected, it leaves the connection to GPRS modem and MTProg takes care of initializing modem and establishing connection.

**Ping**
IP address pinged by application in order to maintain internet connection session. This address must belong to the same APN as programmed module. (Can be the programmed module's IP). If used, leave it at default „0.0.0.0".
**PIN code**
Contains PIN code of the SIM card placed in the modem employed to communicate with APN.

**Modem port**
Contains number of the com port the GPRS modem is connected to.

**RS-232 Port**
Contains number of the com port the programmed module is connected to via serial cable.

Select the chosen module and connection type. In order to do so activate parameter „Select" from menu „Module" or click the icon  on the toolbar. The selection window will open and present all available options.
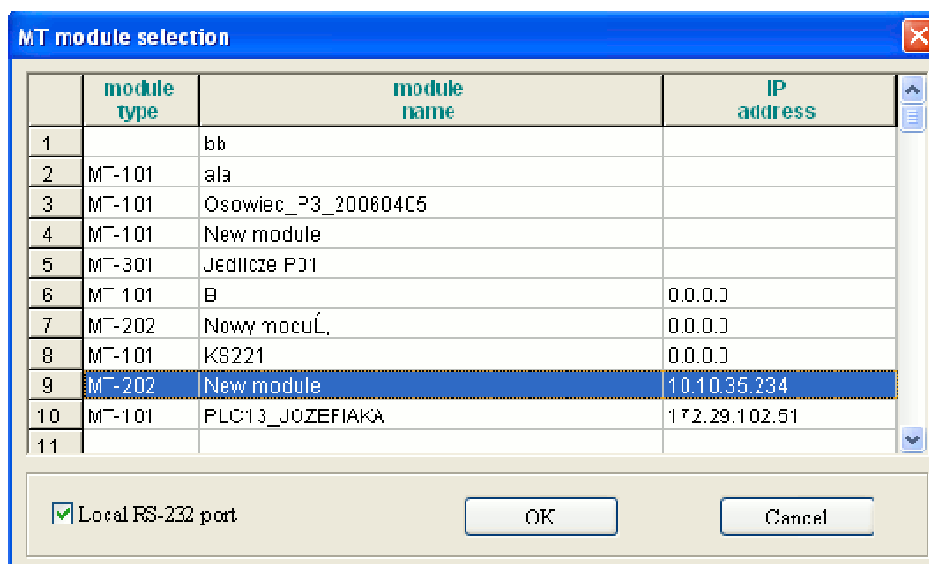
Notice that only modules defined and configured by MTManager application in active Project are selectable.

When Local port RS-232 is checked in, serial transmission via RS-232 takes place. When Local port RS 232 is unchecked, GPRS transmission is employed and module IP address is used.

„OK" button stores the choice for connecting MTprog.exe application with selected module and opens program editing table.

## 9.3. Main window layout

Main program window displays a table containing the program to be executed by module's command interpreter. Right side of the window contains a field with button groups dedicated to defining operations and constants. On top, there is a system menu and a toolbar with icons for frequently used functions. At the bottom, a status bar displays from the left: status of command interpreter, selected module's name and selected communication channel.
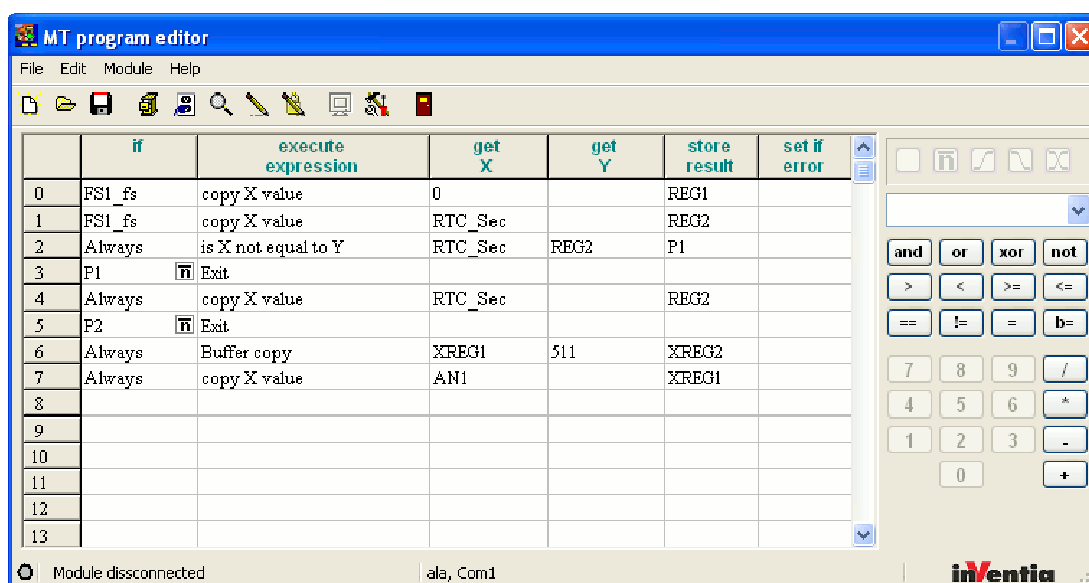
## 9.3.1. Menu items

## 9.3.1.1. File
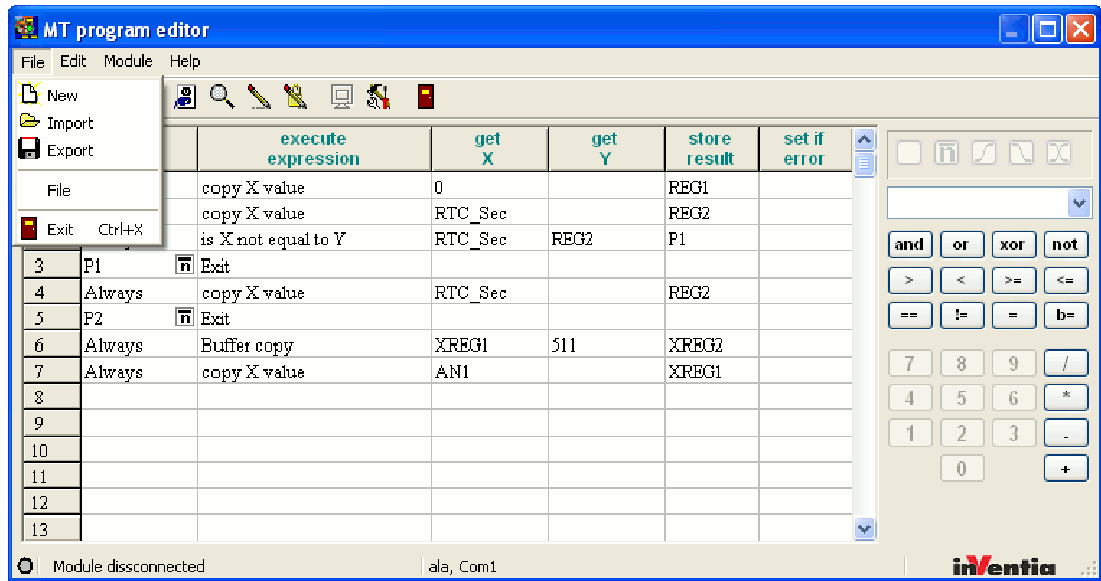


**Function "New"**
Erases the program visible in the table and the table is ready for editing of a new program.

The icon ![icon] on the toolbar performs same function.

**Function "Import"**
Writes a program previously stored on the hard disc into the table. Programs have a default extension ".MTp".

The icon ![icon] on the toolbar performs same function.

**Function "Export"**
Stores the program from the table on the hard disc with default extension ".MTp".

The icon ![icon] on the toolbar performs same function.

**Function "File"**
Stores the program from the table on the hard disc in clear text, creating good documentation.

**Function "Exit"**
Closes the application after confirmation. The keyboard shortcut for this function is "Ctrl-X".

The icon ![icon] on the toolbar performs same function.

## 9.3.1.2. Edit
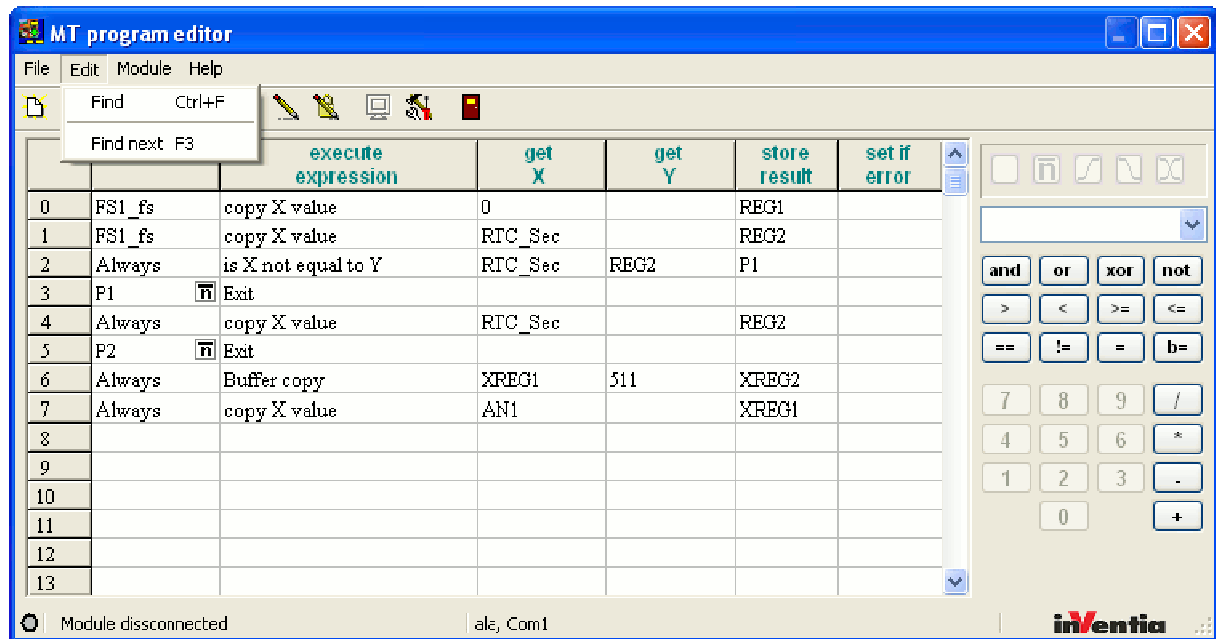


**Function "Find"**
Allows searching the program for defined text (e.g. register name). Text that you want to look for is entered in the window opened jus after clicking this menu position. It is possible to choose if the search phrase is case sensitive and if search phrase is a complete "word" or only part of it (whole contents only). After selecting OK cursor in main window will move to first cell with found search phrase.
Search area begins from first cell after selected and ends where program ends.
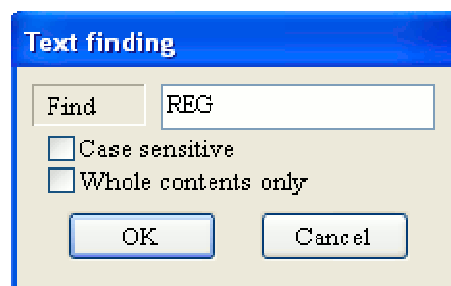


Function can also be started by using keyboard shortcut **Ctrl+F**.

**Function "Find next"**
Moves cursor to next cell with searched phrase.
Function can also be started by using keyboard shortcut **F3**.

## 9.3.1.3. Module

**Module Menu** consists of functions governing the state of programmed module. Functions of this menu item change dynamically depending on the state of connection with the module and the state of the module's internal program.
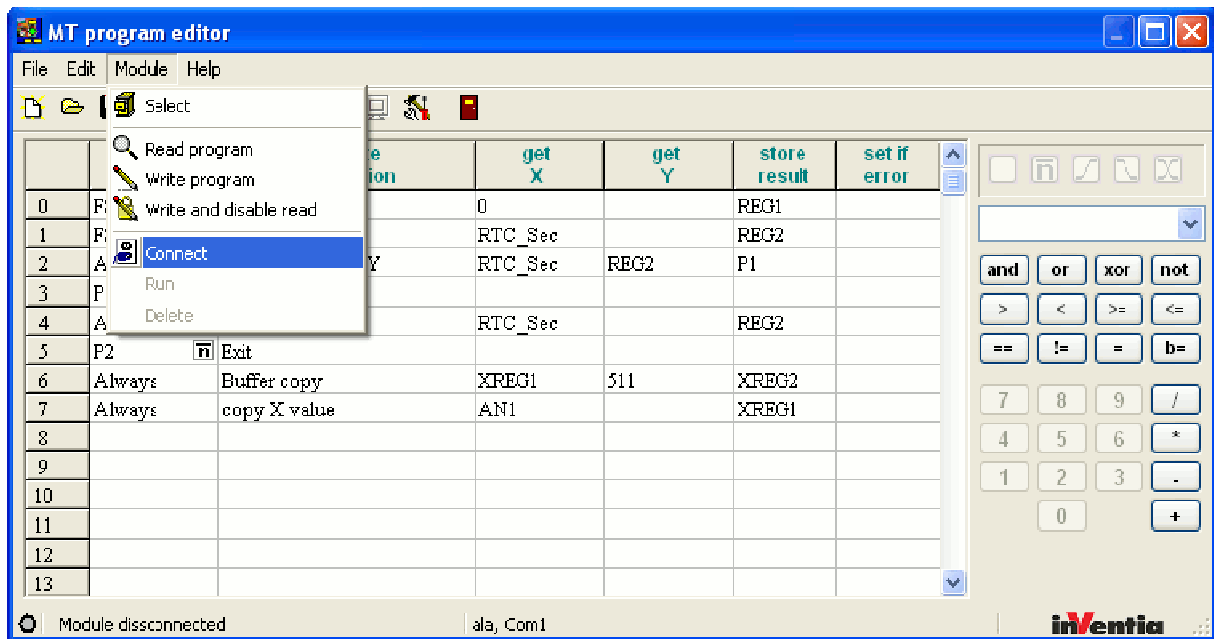
and after connecting to the module with internal program running. Notice the green dot in lower left corner of the status bar.
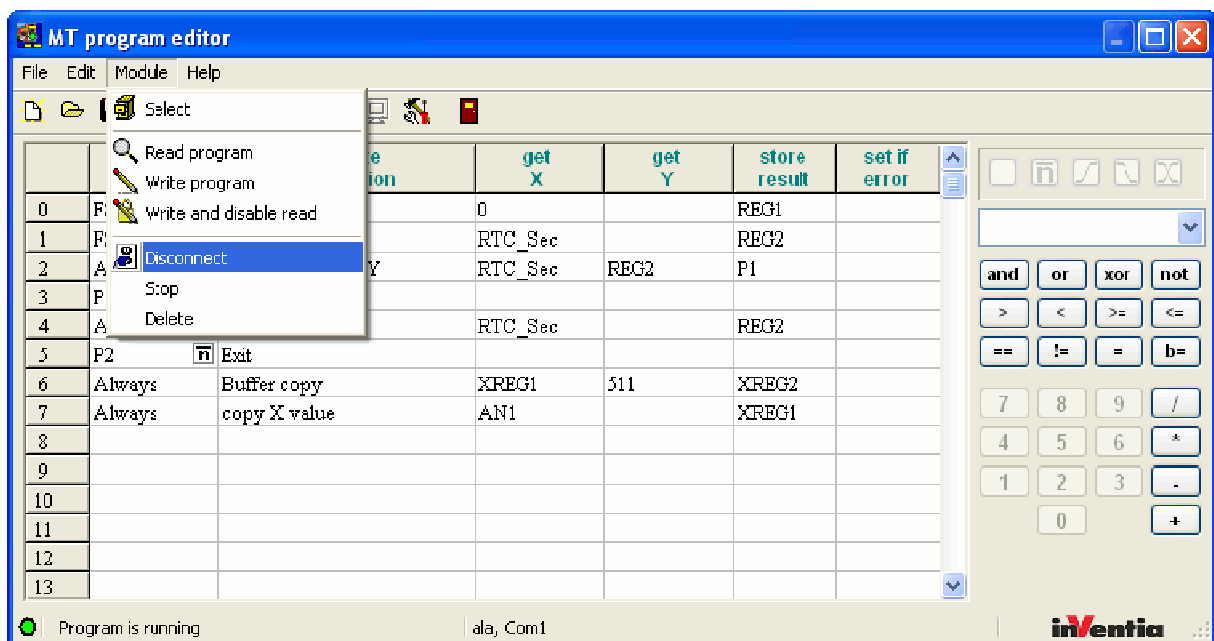


**Function "Select"**
Presents the list of defined modules for selection of the desired module. The list has the option of selecting transmission mode via either RS-232 cable or wireless (GPRS) connection using the module's IP address.
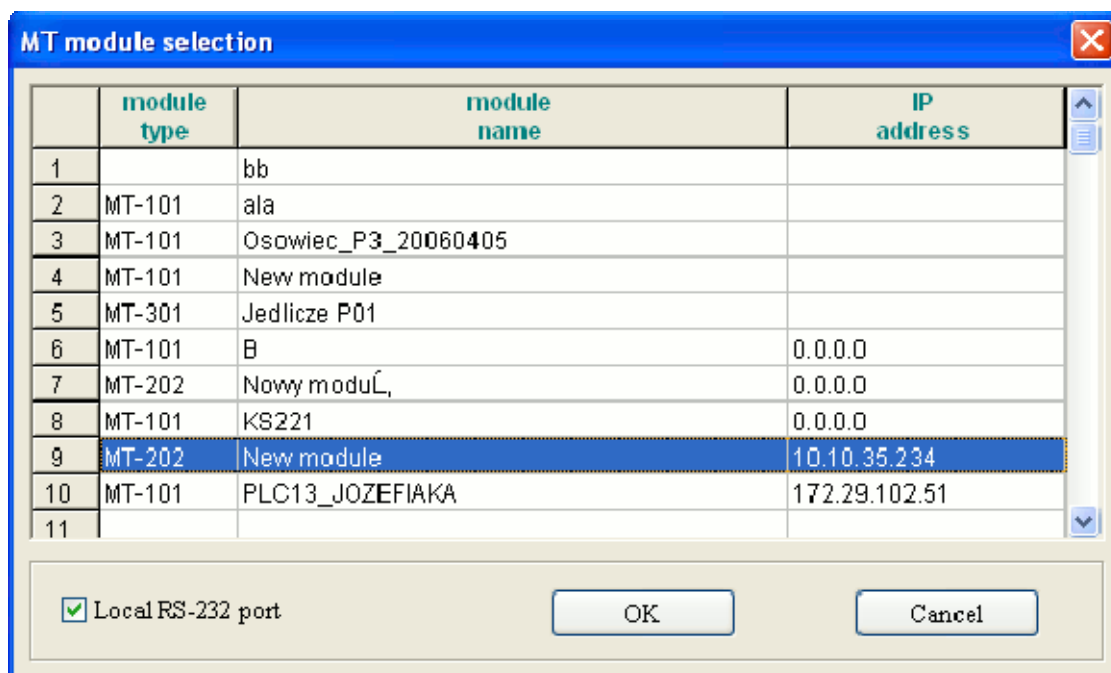
The table shows data written in system registers by MTManager application. MTProg application can only access modules previously defined and configured in active Project by MTManager.

The icon  on the toolbar performs same function.

**Function "Read program"**
Reads the content of module's program interpreter into the table.

If the application is connected to the module, the control in the left side of the status bar is lit in red or green depending on the state of the program interpreter. The text displayed close to the control reflects the actual state and the function "Read program" is active.



If the connection is not established or broken the control is gray and reading from the module is impossible.
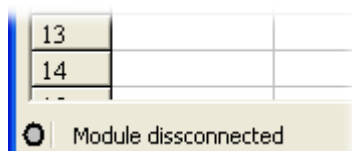


In case of serial cable connection the queries about sequential program lines are sent. Lines are continuously read and displayed in the table. In GPRS mode there is a couple of seconds delay between the query and response arrival. In order to speed the function up queries are sent without waiting for response. Upon arrival of response the value is displayed in the table.

The icon  on the toolbar performs same function.

**Function "Writing to module"**
This function writes the program from the table to the module. If the table is empty, the effect of using this function will be erasing a program existing in the module.

66

All introductory remarks concerning the connection and password protection of Read function are valid in this case as well. The process of writing program into module's interpreter is similar. The only difference appears in GPRS mode. After verification of privileges all lines of program are sent and the application waits for confirmations.

The icon ![icon] on the toolbar performs same function.

**Function "Write and disable read"**
This function writes the program from the table into the module disabling the reading function. The only way of modifying the program in the module is to write it again or import previously stored programs and modify it.

*The icon* ![icon] on the toolbar performs same function.

**Function "Connect"**
This function toggles between ON-Line and OFF-Line mode.

In first case, MTprog.exe application sends cyclically queries about module's interpreter state and on the base of received response displays status information in lower left corner of main window. The menu item Help/Transmission opens transmission window displaying sent commands and replies.

ON-Line/OFF-Line differentiation is important during connection via modem in GPRS mode. Leaving the application connected with the module for longer time results in higher costs of transmission since transfer may be roughly estimated to 2 kB per minute.

In ON-Line mode, the serial port connecting the computer with the module or GPRS modem is occupied by MTprog.exe application and cannot be used by other applications. In OFF-Line mode, serial port is released and may be used by any other application.

The icon ![icon] on the toolbar performs same function.

**Function "Start"**
This function starts the command interpreter of the module.
To start the interpreter the module has to be in ON-Line mode. This function does not have a corresponding icon on the toolbar.

**Function "Delete"**
This function deletes data from the module's data memory space.
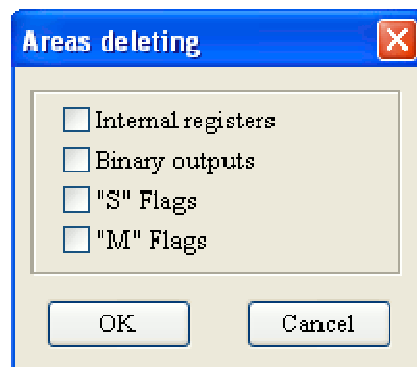


The selected for deletion areas should be checked in. Pressing the OK button resets selected areas.

## 9.3.1.4. Help
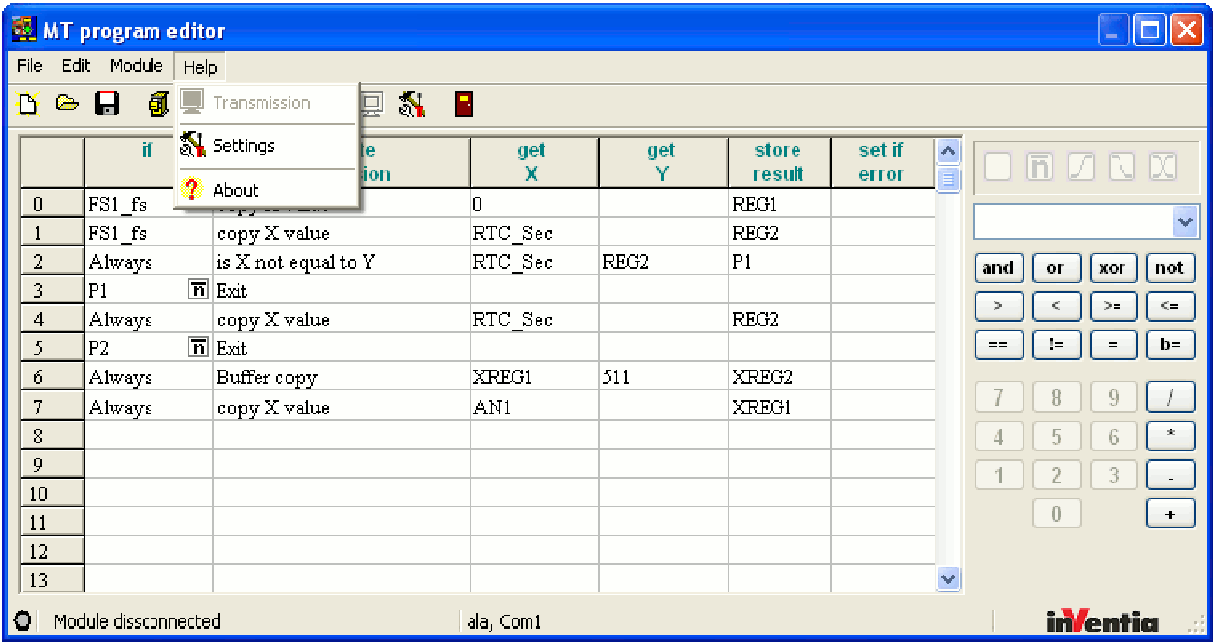


**Function "Transmission"**
toggles transmission review window
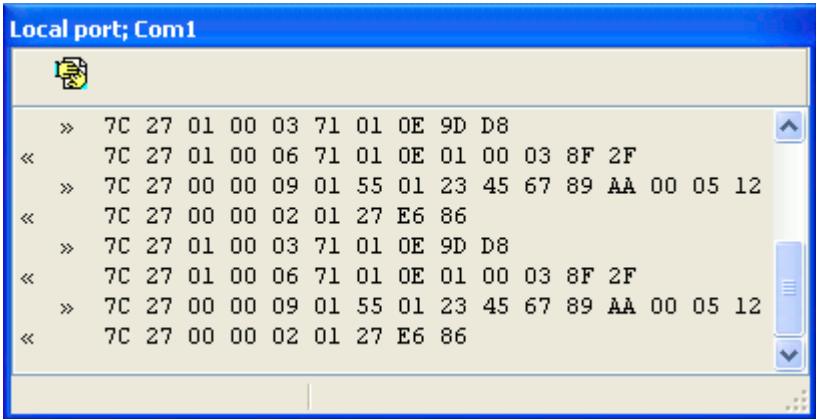


The title bar displays transmission type and recipient address while status bar displays connection status. The tool bar displays the icon closing the window.

The main window's tool bar icon ⬛ toggles transmission window display.

**Function "Settings"**
opens environment configuration window described in "Starting to work" section.

The icon 🔧 on the toolbar performs same function

**Function "About"**
opens window displaying version number and Manufacturer's address data.

## 9.3.1.5. Toolbar

The main window's toolbar holds icons corresponding to following menu functions:

| | menu item "File" | Function "New" |
| --- | --- | --- |
| | menu item "File" | Function "Red" |
| | menu item "File" | Function "Write" |
| | menu item "Module" | Function "Select" |
| | menu item "Module" | Function "Connect/Disconnect" |
| | menu item "Module" | Function "Read program" |
| | menu item "Module" | Function "Write program" |
| | menu item "Module" | Function "Write and block reading" |
| | menu item "Help" | Function "Transmission" |
| | menu item "Help" | Function "Settings" |
| | menu item "File" | Function "Exit" |

## 9.4. Program editor table

The table has six columns. Each column has specific role assigned to be performed by the command interpreter:

| | if | execute expression | get X | get Y | store result | set if error |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | | | | | | |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |

"If" – defines the condition for the table row to be executed. If not met, the line will be skipped.
"Execute expression" – defines the function to be performed,
"Get X" - holds the first argument of interpreted function,
"Get Y" - holds the second argument of interpreted function,
"Store Result" - points to where to store the result of the function,
"Set if error" - holds the optional flag to be set if the function fails due to e.g. division by zero or overflow.

## 9.5. Standard functions

Most commonly used functions are grouped with numeric keyboard on the right side of the main window.
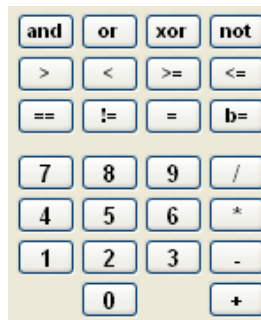


Top button row groups 4 logical operations of true/false type.



Next group represents 6 functions comparing arithmetical values.
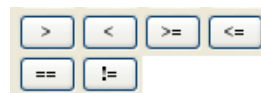


Next 2 buttons represent assignment of arithmetical and logical values.



Right column of numerical keyboard represent 4 standard arithmetical operations.



## 9.6. Numeric keyboard

Numerical keyboard has all keys active only when selected function allows arithmetic input.]
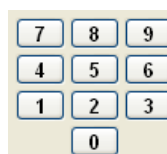


If selected function allows only logical input only keys "0" and "1" meaning respectively False/Never and True/Always.
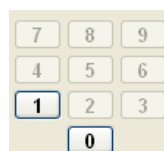
When selected function does not allow numerical input the keyboard is grayed out (inactive).



## 9.7. Auxiliary functions

Some modules types and versions may implement auxiliary functions. Access to these functions is possible via the drop-down menu just above numerical keyboard:
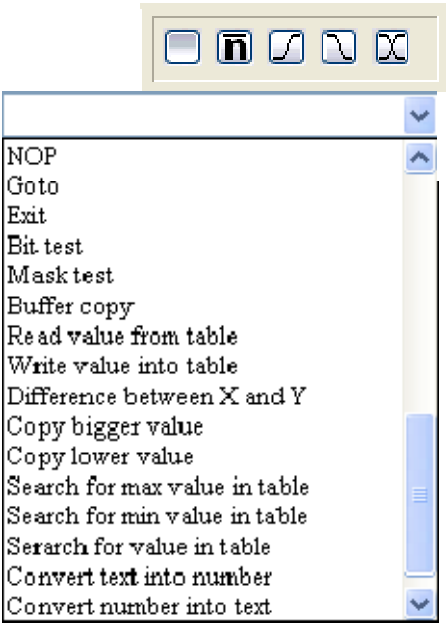


Detailed description of standard and auxiliary functions is located at chapter Description of program functions

## 9.8. Description of Program functions

**Boolean AND X ,Y**

| get X (bit) | get Y (bit) | store result (bit) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Boolean OR X , Y**

| get X (bit) | get Y (bit) | store result (bit) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Boolean XOR X , Y**

| get X (bit) | get Y (bit) | store result (bit) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Boolean NOT X**

| get X (bit) | get Y | store result (bit) |
|---|---|---|
| 0 | n.a | 1 |
| 1 | n.a | 0 |

**Is X greater than Y**

| get X (register) | get Y (register) | store result (bit) |
|---|---|---|
| get X > get Y | | 1 |
| get X <= get Y | | 0 |

**Is X lower than Y**

| get X (register) | get Y (register) | store result (bit) |
|---|---|---|
| get X < get Y | | 1 |
| get X >= get Y | | 0 |

**Is X greater or equal Y**

| get X (register) | get Y (register) | store result (bit) |
|---|---|---|
| get X >= get Y | | 1 |
| get X < get Y | | 0 |

**Is X lower or equal Y**

| get X (register) | get Y (register) | store result (bit) |
|---|---|---|
| get X <= get Y | | 1 |
| get X > get Y | | 0 |

**Is X equal Y**

| get X (register) | get Y (register) | store result (bit) |
|---|---|---|
| get X equal get Y | | 1 |
| get X not equal get Y | | 0 |

**Is X not equal Y**

| get X (register) | get Y (register) | store result (bit) |
|---|---|---|
| get X not equal get Y | | 1 |
| get X equal get Y | | 0 |

**Copy X value**
store result (register) = get X (register)

**Copy Boolean X**
store result (bit) = get X (bit)

**NOTICE!!!**
When copying register DREG to a 16 bit register the function returns an error, since the copied value is not in the range of 0-65535.

**Divide X by Y**
store result (register) = get X (register) / get Y (register)

**NOTICE !!!**
The function operates on integers. The result is truncated to integer.
Crossing the valid range sets an error flag.
Examples:
    10 / 3 = 3
   -15 / 4 = -3

**Multiply X by Y**
store result (register) = get X (register) * get Y (register)
Crossing the valid range sets an error flag.

**Subtract Y from X**
store result (register) = get X (register) - get Y (register)
Crossing the valid range sets an error flag.

**Add X to Y**
store result (register) = get X (register) + get Y (register)
Crossing the valid range sets an error flag.

**Modulo**
The function assigns the modulus of the division of X by Y. For proper functioning, it is required that get X >= 0 and get Y > 0. When any of preconditions are not met, the function returns an error which sets an error flag.
Examples:
    10 / 3 = 1
   -15 / 4 = error
    15 / 5 = 0

**Negation**
store result (register) = -get X (register)

**CAUTION!!!**
Note that all registers store unsigned values. Only DREG registers can hold 32 bit signed values. Using this function on DREG registers in will in most cases return an error.

**Bitwise OR**
The function calculates logical sum of corresponding bits in 2 registers.

Example:

|  | decimal | hexadecimal | binary |
|---|---|---|---|
| get X (register) | = 1234 | = 04D2 | = 0000 0100 1101 0010 |
| get Y (register) | = 4991 | = 137F | = 0001 0011 0111 1111 |
| store result (register) | = 6143 | = 17FF | = 0001 0111 1111 1111 |

## Bitwise AND
The function calculates the product of corresponding bits in 2 registers.
Example:

|  | decimal | hexadecimal | binary |
|---|---|---|---|
| get X (register) | = 1234 | = 04D2 | = 0000 0100 1101 0010 |
| get Y (register) | = 4991 | = 137F | = 0001 0011 0111 1111 |
| store result (register) | = 82 | = 0052 | = 0000 0000 0101 0010 |

## Bitwise XOR
The function calculates symmetrical difference of corresponding bits in 2 registers.
Example:

|  | decimal | hexadecimal | binary |
|---|---|---|---|
| get X (register) | = 1234 | = 04D2 | = 0000 0100 1101 0010 |
| get Y (register) | = 4991 | = 137F | = 0001 0011 0111 1111 |
| store result (register) | = 6061 | = 17AD | = 0001 0111 1010 1101 |

## Bitwise NOT
The function negates bits in the register.

Example:

|  | decimal | hexadecimal | binary |
|---|---|---|---|
| get X (register) | = 1234 | = 04D2 | = 0000 0100 1101 0010 |
| store result (register) | = 64301 | = FB2D | = 1111 1011 0010 1101 |

## Bit copy
The function copies selected bits from a register (get X) to register (store result). Only bits from positions where there are values of 1 in register (get Y). Other bits remain intact.
Example:

|  | decimal | hexadecimal | binary |
|---|---|---|---|
| get X (register) | = 1039 | = 040F | = 0000 0100 0000 1111 |
| get Y (register) | = 4915 | = 1333 | = 0001 0011 0011 0011 |
| store result (register before operation) | = 3925 | = 0F55 | = 0000 1111 0101 0101 |
| store result (register after operation) | = 3143 | = 0C47 | = 0000 1100 0100 0111 |

This function is very handy when copying values between register space and bit addressed memory space. Virtual registers from bit addressed memory spaces (VREG_BIx - binary inputs, and VREG_BO - binary outputs) enable access from functions operating on registers to bit variables. Virtual register mapping is sequential: first register holds first 16 bits, the next following 16 and so on...

## NOP
No operation performed.

## Goto
The function jumps to specified in parameter (get X) location (program line).

## CAUTION!!!
The number of instructions carried out in one program cycle is limited to 2000. Upon reaching limit, the execution stops. Next cycle starts from line number 0. Reaching the limit is signaled by ERR LED for diagnostic purposes.

## Exit
The function ends the program execution for a particular cycle.

## Bit test

This function verifies whether at least one bit in get X register is set according to mask defined by get Y register. If confirmed, the function returns value 1 in store result column, else 0.
Example:

|  | hexadecimal | binary |
|---|---|---|
| get X (register) | = 04D2 | = 0000 0100 1101 0010 (value) |
| get Y (register) | = 1820 | = 0001 1000 0010 0000 (mask) |
| store result (bit) | = 0 | = 0 |
|  |  |  |
| get X (register) | = 04F2 | = 0000 0100 1111 0010 (value) |
| get Y (register) | = 1820 | = 0001 1000 0010 0000 (mask) |
| store result (bit) | = 1 | = 1 |

## Mask test

This function verifies whether all bits in get X register defined in mask get Y are set. If confirmed, the function returns value 1 in store result column, else 0.

Example:

|  | Hexadecimal | binary |
|---|---|---|
| get X (register) | = 04D2 | = 0000 0100 1101 0010 (value) |
| get Y (register) | = 1820 | = 0001 1000 1110 0010 (mask) |
| store result (bit) | = 0 | = 0 |
|  |  |  |
| get X (register) | = 04F2 | = 0001 1100 1111 0010 (value) |
| get Y (register) | = 1820 | = 0001 1000 1110 0010 (mask) |
| store result (bit) | = 1 | = 1 |

## Buffer copy

This function copies value of source buffer starting at register (get X) to destination buffer defined in store result. The length of copied buffer depends on the value in (get Y) register. The function stops when:
- buffer length value is negative,
- writing attempted beyond the buffer space,
- copied value exceeds allowed value of destination buffer. (For instance, an attempt to copy -1 from 32 bit register to 16 bit register).
Example:

```
get X (register)        = XREG10
get Y (register)        = REG1
store result (register)  = DREG2
```

If REG1 = 4, to function copies 4 values:
DREG5 = XREG13,
DREG4 = XREG12,
DREG3 = XREG11,
DREG2 = XREG10

## CAUTION!!!

The function starts copying from the last buffer register and ends at the first register. It creates an easy way to create a logger keeping the history of selected resources in internal registers. In section "Examples of programs" a „Logger program" illustrates using this function.

## Copy block quickly

The function copies data among blocks in internal registers space. Following parameters are used:
register (get X)        – source block address

register (get Y)          – block size
register (store result)     - destination block address
Destination and source block may overlap.
When block size exceeds the size of internal register space the data is not copied and the function rises an error flag.

Example:
REG1 = 0x60, REG2 = 0x100, REG3 = 0x600

    get X (register)        = REG1
    get Y (register)        = REG2
    store result (register)   = REG3

The function copies data from source block 0x060…0x15F (XREG1…XREG256) to destination block 0x600…0x6FF (P2SND_B1…P2SND_B256)

**Read value from table**
This function takes the numerical value from table starting with (get X) register. The value from table cell defined by index register (get Y) is copied to store result register. Table cells are indexed from 0.
store result = get X [get Y].
The function rises error flag when copied value exceeds the range of destination register or cell address lays beyond the space.
Example:

    get X (register)        = XREG10
    get Y (register)        = REG1
    store result (register)   = DREG2

If REG1 = 4, than function performs the operation:
DREG2 = XREG14

**Write value to table**
The function writes numeric value to table starting with register (store result). The value from register (get X) is copied to table cell defined by index register (get Y). Table cells are indexed from 0.
store result [get Y] = get X.
The function rises error flag when copied value exceeds the range of destination register or cell address lays beyond the space.
Example:
    get X (register)        = REG2
    get Y (register)        = DREG1
    store result (register)   = XREG100

If DREG1 = -5, than function performs the operation:
XREG95 = REG2

**Difference between X and Y**
This function calculates the integer difference between X and Y, no matter which value is higher.
Exceeding the valid range rises the error flag.

store result (register) = integer value of (get X (register) - get Y (register))

**Copy bigger value**
The function verifies which value, X or Y is higher and copies the higher one.

If   get X > get Y        than     store result = get X
If   get X <= get Y        than     store result = get Y

**Copy lower value**
The function verifies which value, X or Y is lower and copies the lower one.

If   get X < get Y          than     store result = get X
If   get X >= get Y         than     store result = get Y

**Search for max value in table**
The function scans the table beginning with register (get X) for length defined by register (get Y). The highest value found in search is written to (store result) register.
Both 16 and 32 bit registers can be searched. If the value found exceeds the range of (store result) register, the error flag is set.
Example 1:
Table from XREG1 = 1, 5, 0, 100, 23, 340, 1, 25, 340, 5, 560, 23

    get X (register)       = XREG1
    get Y (register)       = 10
    store result (register)    = REG1

The result is REG1 = 340
Example 2:
Table from DREG1 = 1, 5, 0, -12000, 23, 340, 1, 25, 340, 5, 65000, 23, 100000, -65000

    get X (register)       = DREG1
    get Y (register)       = 12
    store result (register)    = REG1

The result is REG1 = 65000.
If register had the value of 13 for (get Y), the function would have returned REG1 value equal to 100000.

**Search for min. value in table**
The function scans the table beginning with register (get X) for length defined by register (get Y). The lowest value found in the search is written to (store result) register.
Both 16 and 32 bit registers can be searched. If the value found exceeds the range of (store result) register, the error flag is set.

Example 1:
Table from XREG1 = 1, 5, 6, 100, 23, 340, 1, 25, 340, 5, 560, 0
    get X (register)       = XREG1
    get Y (register)       = 10
    store result (register)    = REG1

The result is REG1 = 1
Example 2:
Table from DREG1 = 1, 5, 0, -12000, 23, 340
    get X (register)       = DREG1
    get Y (register)       = 4
    store result (register)    = REG1

The result is since REG1 cannot hold the value of -12000.

**Search for value in table**
The function scans the table (buffer) starting with register (get X) searching for value defined in (get Y) register. The table is searched for first occurrence of searched value or to the end of the memory space. If the value is found, the function returns an index to the register. Indices are calculated from 0. If not found, the function sets an error flag.

**NOTICE!!!**

The function can search tables of 32 bit registers though one has to be aware that index returned is a 32 table index and not equal to the offset in the address space, as is the case for 16 bit tables.

Example 1:
Table from XREG1 = 1, 5, 0, 100, 23, 340, 1, 100, 340, 5, 560, 23
    get X (register)     = XREG1
    get Y (register)     = 100
    store result (register)   = REG1

The result is REG1 = 3

Example 2:
Table from DREG1 = 1, 5, 0, -12000, 23, 340, 1, -100

    get X (register)     = DREG2
    get Y (register)     = DREG1
    store result (register)   = REG1

The result is REG1 = 5

**Convert text to number**

The function converts decimal value from text to binary. Get X defines start of the text buffer to convert. Text characters are written to low order bytes of 16 bit registers. Get Y defines max number of characters (registers) to convert. The function after successful conversion writes the result in (store result) register. The result is always an integer value. If converted value had a decimal fraction, the decimal separator is omitted and function writes the number of digits after comma to AUX_RET2 register. This allows handling of scaled floating point values. Recognized delimiters are '.' and ','.

Furthermore, the function writes the length of converted value. The converted value may be a negative number provided that result storing register is a 32 bit register.

Examples:

| get X | get | store result | | AUX_RET1 | AUX_RET2 |
|---|---|---|---|---|---|
| | | REG1 | DREG1 | | |
| 0abc | 4 | 0 | 0 | 1 | 0 |
| 0.000abc | 8 | 0 | 0 | 5 | 3 |
| 1234.56789 | 3 | 123 | 123 | 3 | 0 |
| -1234.56789 | 7 | Error | -12345 | 7 | 1 |
| +0.1234abc | 10 | 1234 | 1234 | 7 | 4 |
| abc | 3 | Error – number format | | | |
| 1234.5678900 | 12 | Error – value too big | | | |
| 1234.56 | 7 | Error | 123456 | 7 | 2 |
| 0,00000012 | 10 | 12 | 12 | 10 | 8 |
| .123 | 4 | Error – number format | | | |
| 123.456.789.000 | 15 | Error | 123456 | 7 | 3 |
| +000111.2 | 9 | 1112 | 1112 | 9 | 1 |
| 12.0000 | 7 | Error | 120000 | 7 | 4 |
| 12.0000 | 6 | 12000 | 12000 | 6 | 3 |

## Convert number to text

This function converts a binary value from register (get X) to text. The result is written into buffer starting with (store result) register. Separate text characters are stored in low order bytes of 16 bit registers. Get Y defines converted number's format.

get Y:
- unit number defines the number of digits after comma in resultant format.
- tens number defines number of digits before comma , 0 – automatically
+ 100 – sets delimiter to ',' instead of '.'
+ 200 – forces adding a sign before positive value.

Furthermore, the function writes the resultant number's length in characters to AUX_RET1 register.
If the converted number is larger then allowed range, the function does not perform the conversion and raises an error flag.

Examples:

| get X | get Y | store result | AUX_RET1 |
|-------|-------|--------------|----------|
| 0 | 0 | 0 | 1 |
| 123 | 0 | 123 | 3 |
| -1234 | 0 | -1234 | 5 |
| 12345 | 2 | 123.45 | 6 |
| 123456 | 103 | 123,456 | 7 |
| 123456 | 23 | Error | --- |
| 0 | 323 | +00,000 | 7 |
| -15 | 323 | -00,015 | 7 |
| -15 | 50 | -00015 | 6 |

## Logical shift left

Function shifts left bits in argument X. Number positions to shifts is defined by parameter Y. The bits that are shifted out are discarded, and zeros are shifted in. Get Y should be $\geq 0$, otherwise operation will not be completed and error bit will be set. Error bit will be set also when bit set high (1) will be shifted out.

Examples:

| get X | get Y | store result | set if error |
|-------|-------|--------------|--------------|
| 0x0000 | 1 | 0x0000 | 0 |
| 0x1234 | -2 | undefined | 1 |
| 0x0001 | 15 | 0x8000 | 0 |
| 0x0003 | 15 | REG=0x8000 | 1 |
| 0x0003 | 15 | DREG=0x00018000 | 0 |
| 0x0003 | 31 | REG=0x0000 | 1 |
| 0x0003 | 31 | DREG=0x80000000 | 1 |
| 0x0001 | $\geq$32 | 0x0000 | 1 |
| 0x0000 | $\geq$32 | 0x0000 | 0 |
| 0x00010000 | 0 | DREG=0x00010000 | 0 |
| 0x00010000 | 0 | REG=0x0000 | 1 |

**Logical shift right**

Function shifts right bits in argument X. Number positions to shifts is defined by parameter Y. The bits that are shifted out are discarded, and zeros are shifted in. Get Y should be $\geq 0$, otherwise operation will not be completed and error bit will be set. Error bit will be set also when bit set high (1) will be shifted out.

Examples:

| get X | get Y | store result | set if error |
|-------|-------|--------------|--------------|
| 0x0000 | 1 | 0x0000 | 0 |
| 0x1234 | -2 | undefined | 1 |
| 0x1112 | 1 | 0x0889 | 0 |
| 0x1111 | 1 | 0x0888 | 1 |
| 0x01118000 | 15 | 0x0223 | 0 |
| 0x81118000 | 15 | REG=0x0223 | 1 |
| 0x81118000 | 15 | DREG=0x00010223 | 0 |
| 0xC0000000 | 31 | 0x0001 | 1 |
| 0x80000000 | $\geq$32 | 0x0000 | 1 |
| 0x0000 | $\geq$32 | 0x0000 | 0 |
| 0x00010000 | 0 | DREG=0x00010000 | 0 |
| 0x00010000 | 0 | REG=0x0000 | 1 |

## 9.9. Description of internal function blocks

### 9.9.1. Timers T1...T8
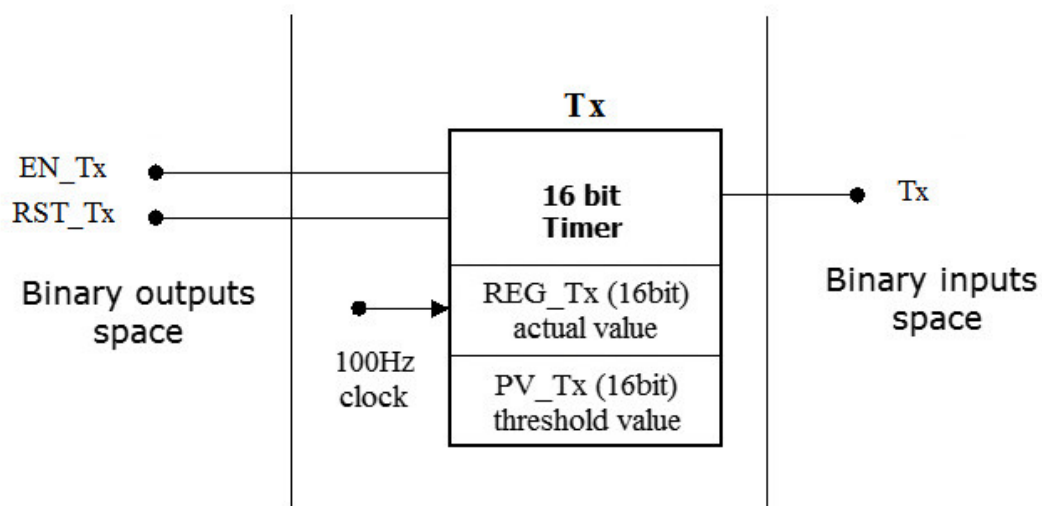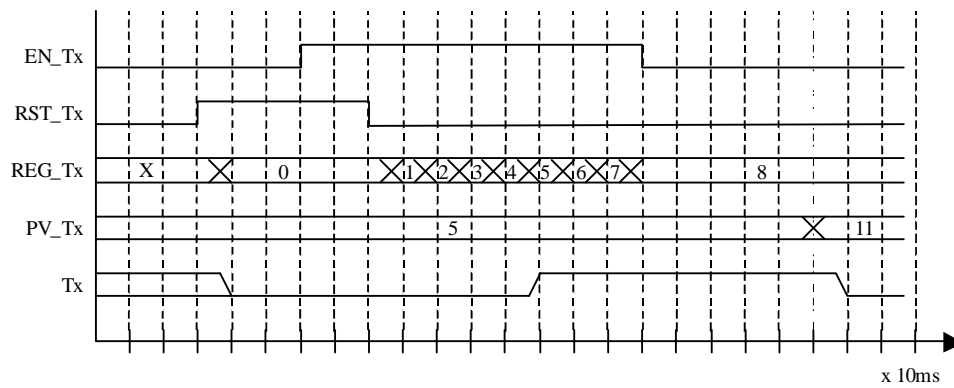
### 9.9.2. Counters C1...C8
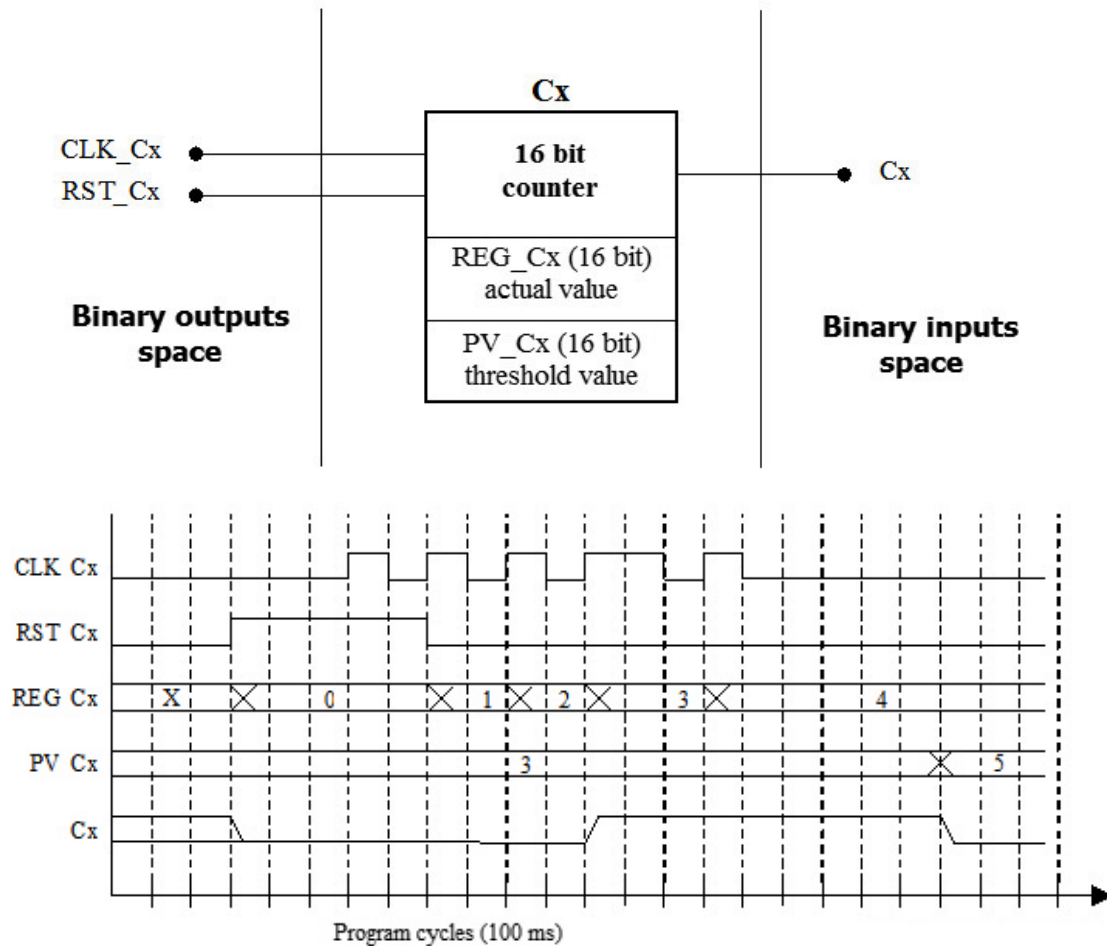
## 9.10. Signal levels or edges

All logical input values may be additionally affected by a level or edge condition.



The default value is a positive signal level. However, the user can change each of the input signals (Condition, Parameter X or Parameter Y) so that the program reacts to inverted value, rising edge, falling edge or change of state.

## 9.11. Filling and modifying program table

The program table has to be filled out with subsequent lines starting with the left column and moving towards the right column. Double-clicking on any given field in the table unfolds a list of variables or functions available to the given column.
In the "Condition" column, double-clicking unfolds a list of logical variables whose state can be checked. The name of the variable can also be entered directly by keyboard or by clicking on the 0/1 values on the numerical keypad. After selecting the name of the variable, it is also possible to define the level or edge condition the execution of the function.

Double-clicking with left mouse button causes a list of available functions to drop down. However, the function can also be selected from standard functions grouped around the numerical keypad, from the list placed above the numerical keypad or by entering the name of the function manually. In case of manual entering, a warning message will appear if the name entered manually is not identical to one of the available functions.

For the columns "Take X" or "Take Y", double-clicking with the left mouse button unfolds a box with a list of variables which may be used as arguments of the function selected earlier. These will be either arithmetic or logical variables. The same discrimination will be visible on the numerical keypad in the right side of the window. In case of logical variables, only the "0" and "1" buttons will be active. Of course, the names of variables or values can be entered from the computer keyboard. If the function can only accept one argument, it will be impossible to select and enter a second one.

The "Store result" column is filled out in a similar way. A list of modifiable variables drops down after double-clicking on the column with left mouse button. Logical and analog inputs will not be displayed. The type of variables is correlated with the previously selected function.

The optional selection in the "If error" column is limited to logical variables whose modification is possible.

Manipulation of program rows is possible after activating context menu by clicking with right mouse button.
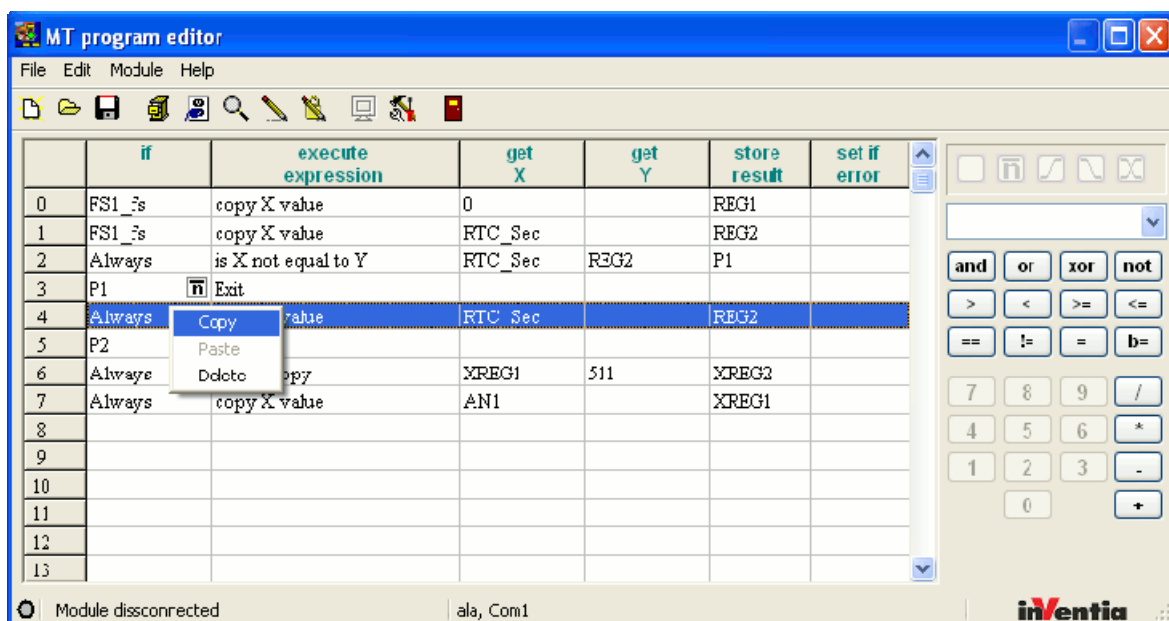
Functions of Context Menu apply for the selected table row. Copied rows are pasted above the highlighted row.

## 9.12. Downloading the program

After completed editing of program, data can be sent to the module.

The method of transmission depends on selected means of communication. When programming locally, it is necessary to establish RS232 cable connection.
For remote programming, it is necessary that the computer on which MTProg is running has a network connection to the APN where the module resides.
Next step is selecting "Connect" and "Write Program" or "Write and block reading". The command description can be found in section Menu Module.

## 9.13. Verifying the program

Despite the high reliability of both local and remote programming, it is recommended to verify the program written into the module. Especially in cases where the module does not seem to be acting according to the controlling algorithm.

The device status LED which displays possible problems resulting from errors in processing of downloaded internal programs has to be verified.

## 9.14. Examples of programs

This chapter aims at acquainting the user with common methods of defining algorithms. Programs included in this chapter are built on simple premises and do not take the fact that they are all made for purely educational purposes into consideration. The authors renounce any liability for faults resulting from using programs without prior analysis of circumstances.

### 9.14.1. The timer

This program illustrates the use of the typical timer (1 of 8 available in MTProg), activated by S1 going high (third program line). The present value PV_T1 that the timer counts up to is 5s (clock frequency is 100Hz) - the line is executed only by first program cycle. When preset value is reached, T1 flag is raised and flag P1 is set to high. The timer can be stopped before 5 seconds lapse by setting S1 low.
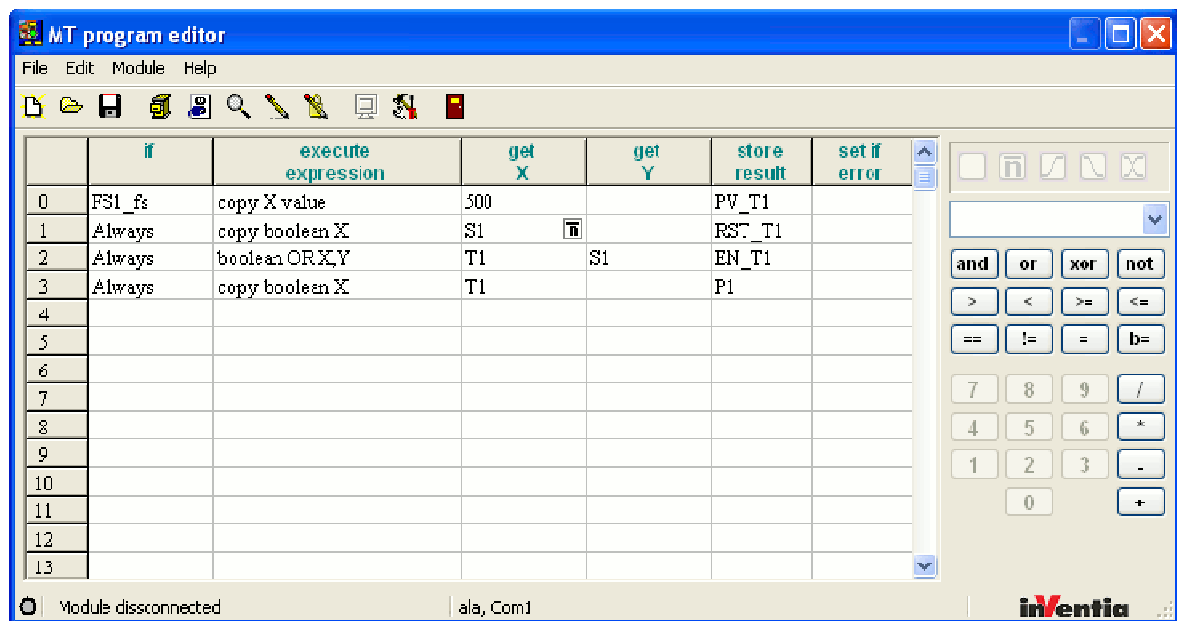
## 9.14.2. The counter

MTProg has 8 counters of this type. The example illustrates a counter counting 10 activations of S1. Upon reaching the count of PV_C1 the flag C1 rises. Line 2 copies the state of C1 to flag P1. Change of P1 may be used in the rule defining the data or SMS transmission or in further programming. The counter is reset (line 3) upon reaching the pre-set value of 10.
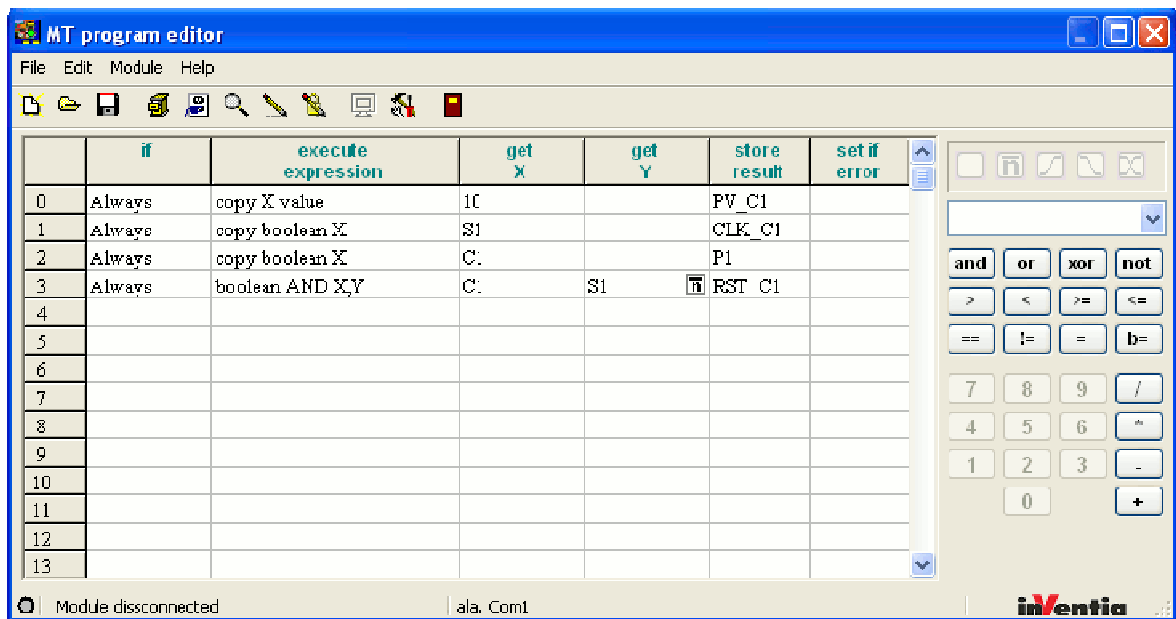
## 9.14.3. Pulse generator

Simple example of pulsing flag P1. The interval of switching on/off is 1 second.

| | if | execute expression | get X | get Y | store result | set if error |
|---|---|---|---|---|---|---|
| 0 | Always | Modulo | RTC_Sec | 2 | REG1 | |
| 1 | Always | is X equal to Y | REG1 | 0 | P1 | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |

Module dissconnected — ala, Com1
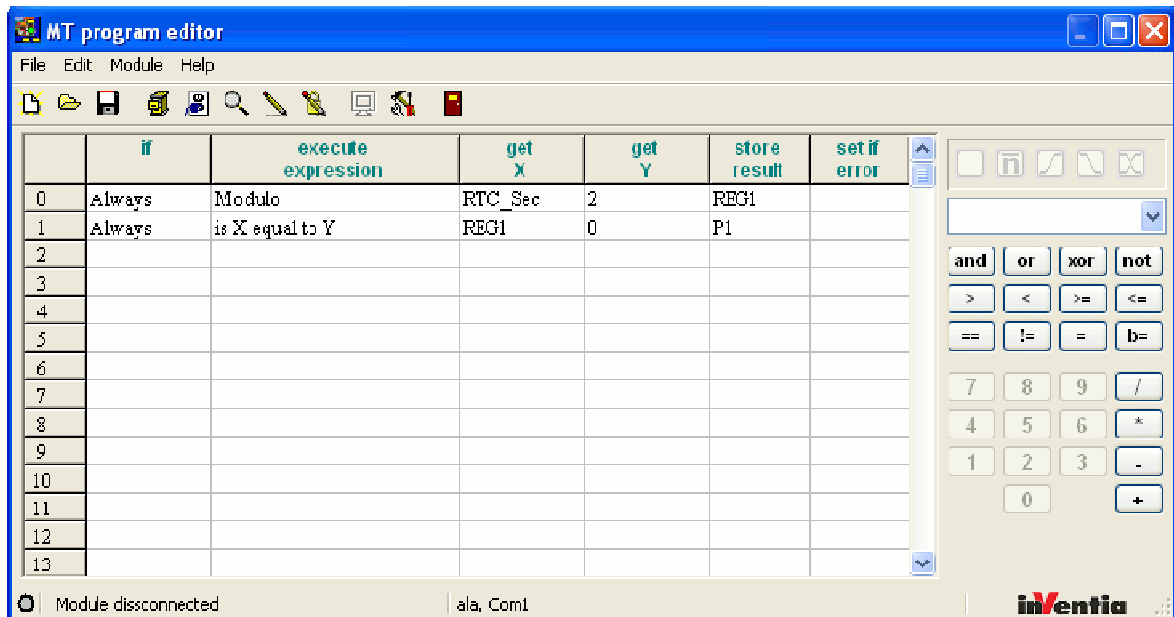
## 9.14.4. Alarm with confirmation

The program generates cyclic transmission of SMS messages until S5, declared as alarm confirmation, is activated or the alarm condition ceases (S1=0)

When binary input S1 is activated, flag P1 is set high. (In MTManager - the rule triggered by P1 sending SMS is defined). Simultaneous to S1 going high, the timer 1 is activated and counts to PV_T1 amounting 600 (6 seconds) starts. Upon reaching PV_T1, the flag T1 is set and subsequently P1 is set. As a result of the rule, an SMS message is dispatched. Until S5 is activated or S1 deactivated, SMS messages will be regularly dispatched.

| | if | execute expression | get X | get Y | store result | set if error |
|---|---|---|---|---|---|---|
| 0 | FS1_fs | copy X value | 600 | | PV_T1 | |
| 1 | S5 | copy boolean X | False | | EN_T1 | |
| 2 | SL1_ok | Exit | | | | |
| 3 | Always | copy boolean X | S1 | | EN_T1 | |
| 4 | Always | copy boolean X | T. | | RST_T1 | |
| 5 | Always | copy boolean X | T. | | P1 | |
| 6 | S1 | copy boolean X | True | | P1 | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |

Module dissconnected — ala, Com1
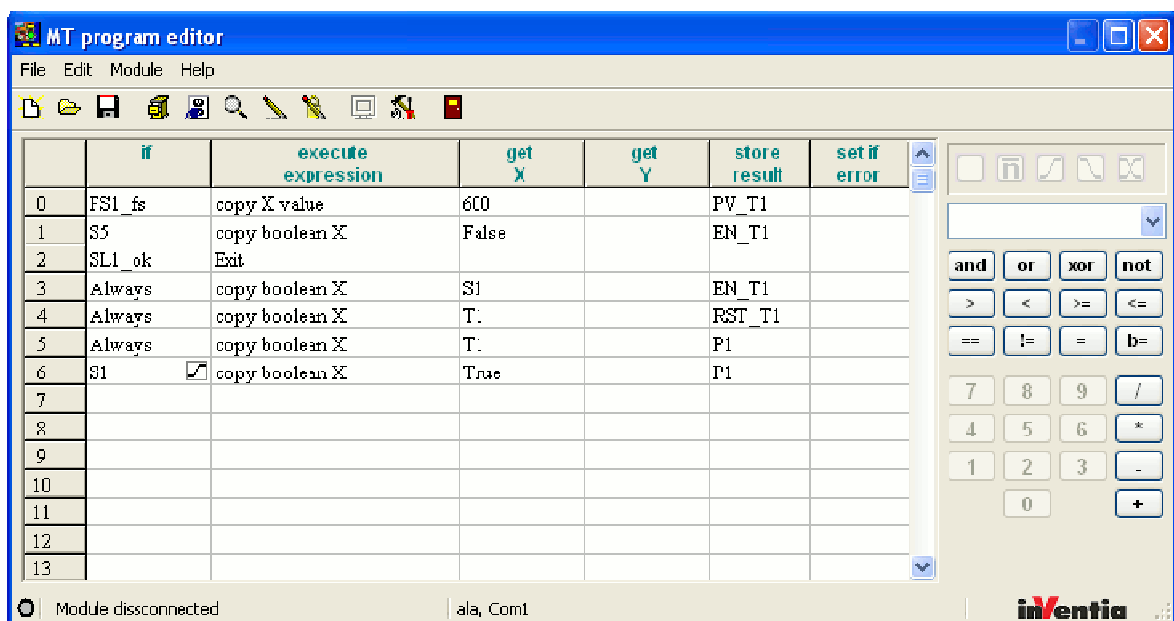
## 9.14.5. Motion detector

The motion detector signal is mapped to S1 and light source to P1. When S1 is activated, the module turns the light on for 10 seconds. Repeated activation of S1 resets the time count back to 10 seconds.
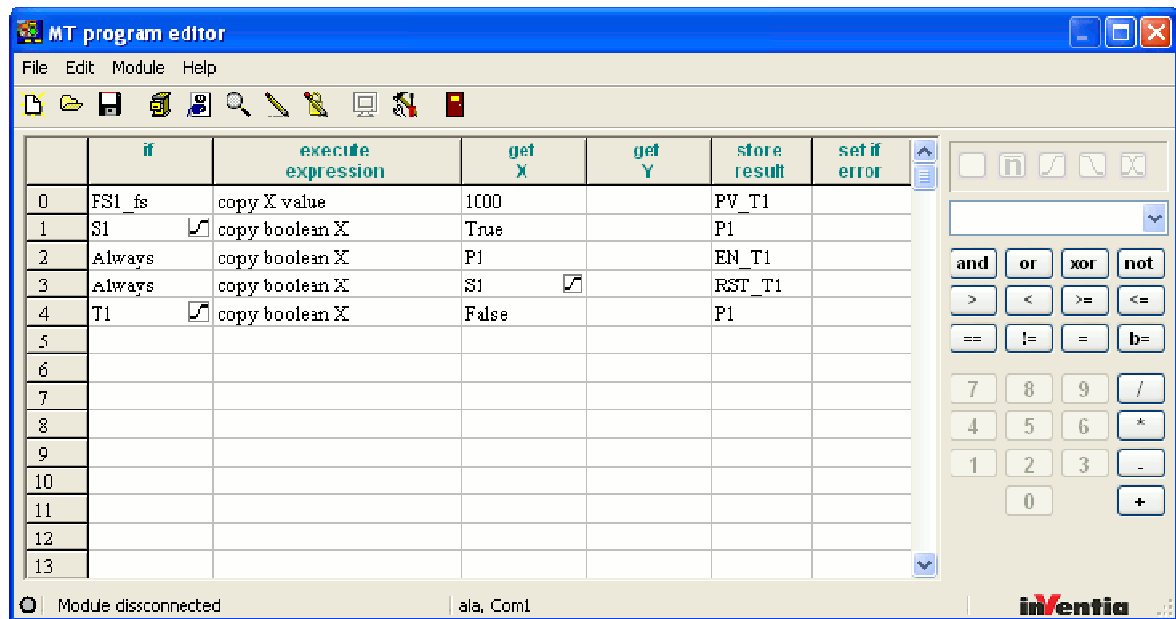


## 9.14.6. Logger program

An example of creating 512 elements logger for REG1 with 1 sec interval activated when P1 = 1.
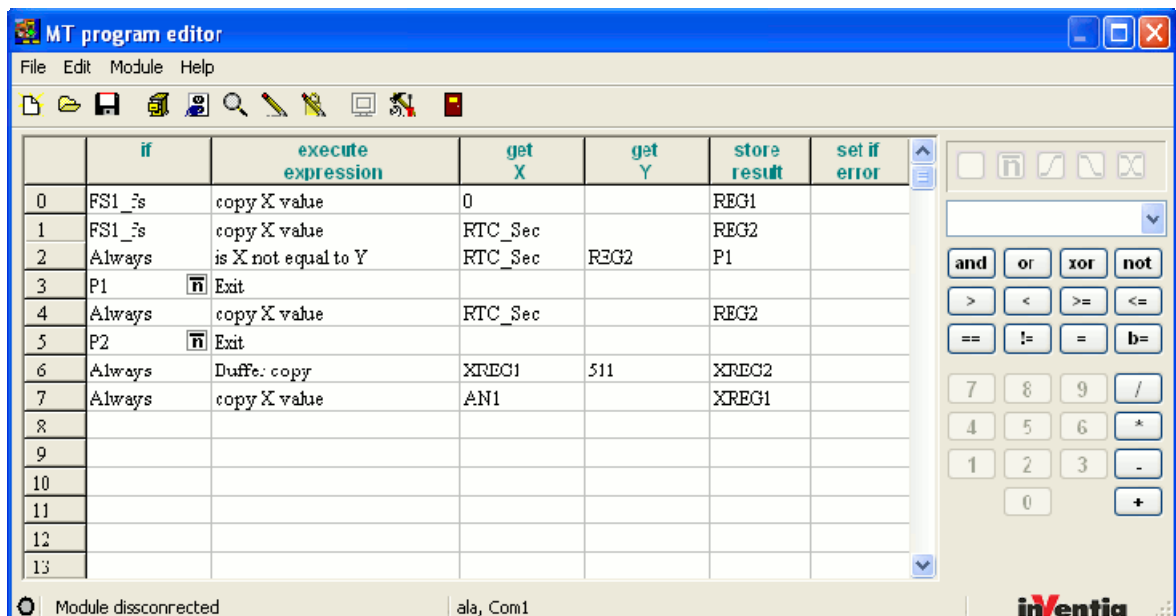
## 10. Problem solving

### 10.1. LED signaling

**LED** indicators placed on front panel of the **MT-202** module are very convenient during start-up procedure. In order to understand their message please get acquainted with error codes.
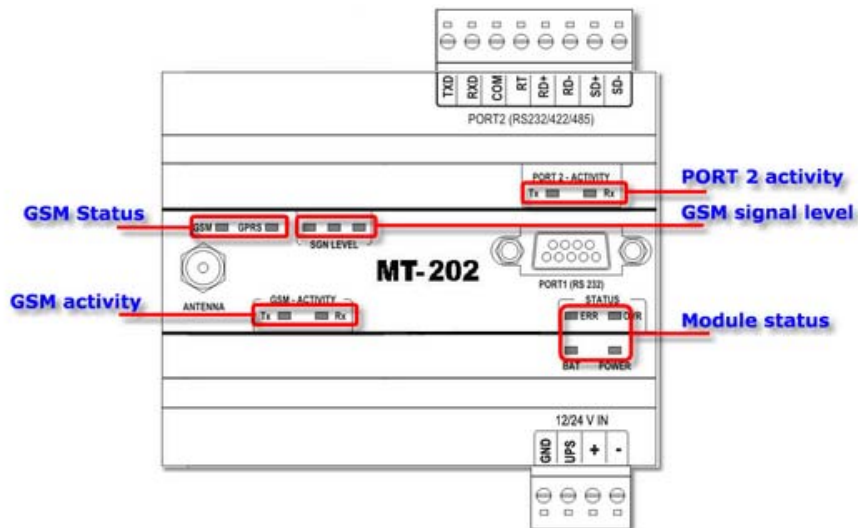


### 10.1.1. GSM status
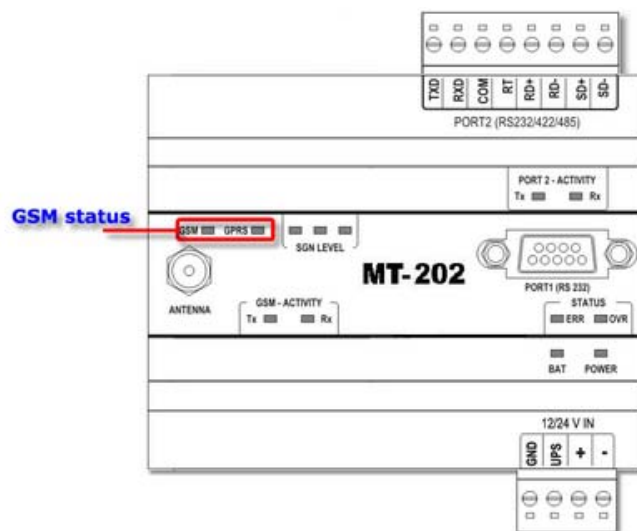


*GSM Status* LEDs indicate:

**GSM** LED- reflects current login to GSM network state.
- lit - the module not logged in
- flashing with app. 2 Hz frequency - the module is logged in

**GPRS** LED - when lit, signifies proper login to APN.

LED indicators for *GSM Status* group reflect module's readiness to perform its duties. Modules not logged in GSM/GPRS network are not able to transmit data and performs cyclic resets and retries to log in.

## 10.1.2. GSM activity
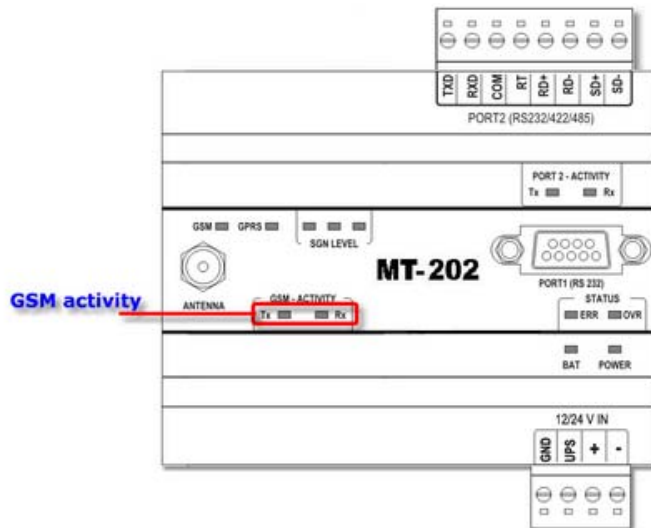


LED indicators **Tx** and **Rx** of *GSM activity* signal, respectively, transmitting and receiving of data via GPRS. Since SMS sending is a form of data transmission, both data frame transmission and SMS transmission cause short flashes of **Tx** LED. Short flashes of **Rx** LED indicate either SMS or data frame reception. This is an easy way of detecting transmission activities.

## 10.1.3. GSM signal level
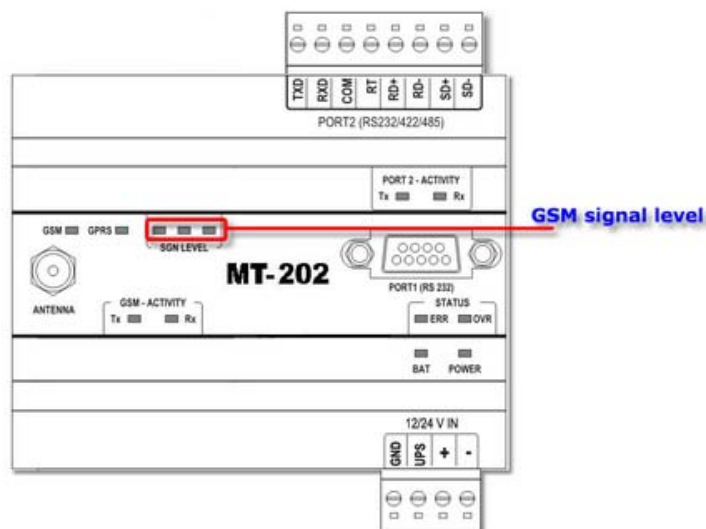


LED indicators of *GSM Signal level* display information received from GSM network on signal level in the place where the antenna is mounted.
It is presumed that for reliable operation at least one LED is lit. Lower level signal does not guarantee reliable operation and means that antenna placement should be changed or the antenna should be replaced with one securing higher signal level.

Reading of signal level happens in the moment of login to network and every 12 minutes, if the module operates in SMS mode. Selection of GPRS mode only means that signal level is investigated only during login procedure, typically at power up. That does not pose any danger since no disturbances in module's operation mean that the signal strength is sufficient.

### 10.1.4. PORT2 activity
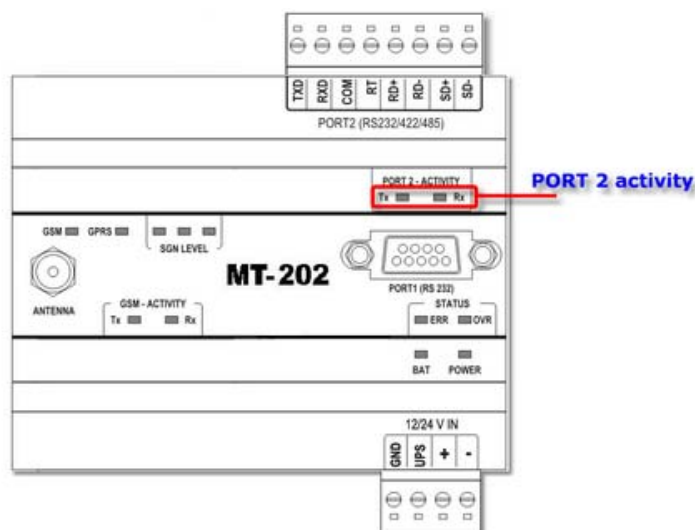


LED indicators **Tx** and **Rx** of *PORT 2 Activity* display, respectively, sending and receiving data via PORT2.
The activity may be observed when:
- The Module operates in *Modbus RTU Master* mode and communicates with attached Master device, passing GPRS received data.
- The Module operates in *Modbus RTU Slave* mode and has sent received query to PORT 2 for Modbus ID different than its own
- The Module operates in *Modbus RTU Mirror* and sends queries for attached SLAVES to PORT 2.
- The Module operates in *Transparent* mode and reroutes data received via GPRS to PORT 2
- The Module operates in *GazModem* or *M-Bus LEC* mode and communicates with attached devices.
- The Module operates in *NMEA0183* mode and receives data from attached device. Activity is observed only on **Rx** LED.
- The Module operates in *Modem* mode and communicates with superior device.
- The Module operates in *FlexSerial* mode running built in port service for communication with device working on non standard transmission protocol.

### 10.1.5. Module's status
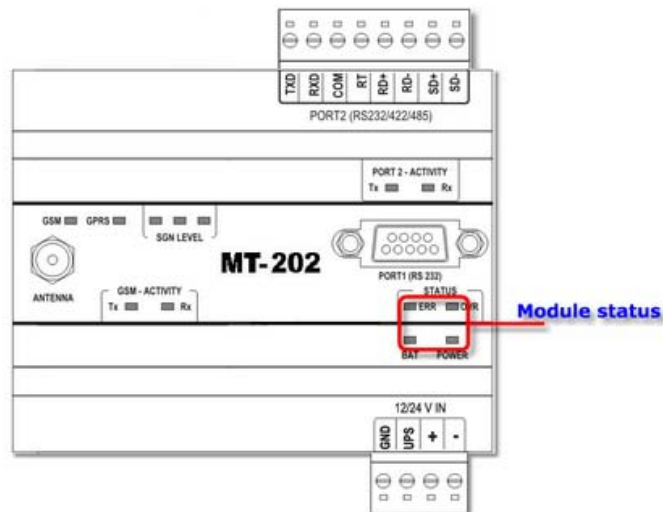


*Module status* group encompasses four LED indicators displaying the state of the circuit controlling the module's operation and with execution of user-defined program.
Significance of LED indicators:

- **Err** LED - when lit, the **Err** LED indicates an error forcing automatic reboot. The reason may be lack of GPRS communication disabling transmission of awaiting data.
  Triple flash of Err LED indicates that current firmware does not support the function used in the program. In this situation, one solution is updating the firmware.
- **Ovr** LED - when lit, the **Ovr** LED indicates that execution of the program cycle took over 100ms and the next cycle was delayed as the result and may render improper execution of the program.
  The LED is lit in following cases:
  - the program loaded is stopped,
  - a new configuration or firmware is loaded and internal program was automatically stopped in order avoid interference. In that case make sure not to disrupt the power supply until the module restarts automatically. It may take couple of minutes.
- **Bat** LED - is lit when the potential on UPS input falls below 13,8V. Since this input is used to signal main supply failure, the system flag FS1_ups is raised simultaneously. The FS1_ups flag may be used in rules processing.
- **Power** LED - is lit all the time the module is supplied with power.

Indicators of *Module status* group are the main sources of visual information about correct operation of the module.


## 10.2. Unblocking of SIM card

Three failed attempts of entering PIN code locks the SIM card and requires entering the PUK code. In order to prevent this, the module controls the number of failed attempts written into SIM card by allowing only two attempts making a third attempt impossible even if the third attempt was correct.
A double unsuccessful attempt is perceived as a fault requiring user intervention. An attempt to unlock the module may be performed only when the right PIN code is known.

Necessary procedure:

- turn the power supply off
- remove SIM card from the module
- insert SIM to ordinary mobile phone accepting cards from the operator that issued actual SIM

- start the phone and enter proper PIN code
- if not accomplished before...
  - start the module
  - insert appropriate PIN into configuration
  - power the module off
- remove the SIM from the phone and place it in the module
- start the module

The described procedure reset SIM card's fault counter and allow using the card in **MT-2O2** module.

In older versions of GSM modems without implemented protection procedures the SIM card may get blocked after 3 failed attempts and the only method of unblocking it is to supply the right PUK code. Unfortunately this cannot be performed in the MT-202 module.

The PUK code may by inserted only after taking the SIM card out of MT-202 module and placing it in a standard GSM mobile phone. The phone will demand entering of PUK code at power-up.
Entering correct PUK-code unblocks the card and resets PIN fault counter making the card operational.

## 10.3. Error signaling

Despite the efforts of module designers and users errors in function do occur. It is often imperative to diagnose and remove the cause of error. Error signaling is a tool for solving problems.

LED indicators on module's front panel interpret errors:

- **ERR**
- **GPRS**
- **L1**, **L2**, **L3** of **SGN LEVEL** group
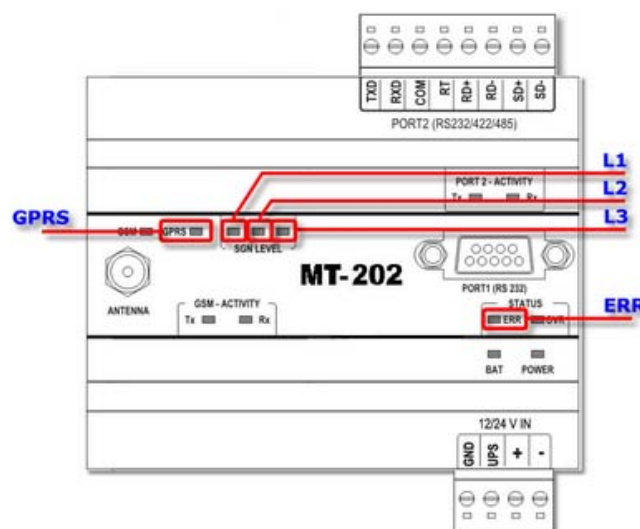


Depending on type, errors are classified as standard or critical. Standard errors are a result of faulty configuration or reasons independent of the module. Critical errors are usually connected to physical damage or faults in internal program.

## 10.3.1. Standard errors

A sign of **Standard error** occurrence is a lit **ERR** LED. Error code numbers are displayed on signal level and **GPRS** LEDs.

| ERR | State |
|---|---|
| ○ | lit |

| GPRS | L1 | L2 | L3 | Error no | Description |
|---|---|---|---|---|---|
| ● | ○ | ● | ● | 1 | Modem error |
| ● | ● | ○ | ● | 2 | GSM network error – check antenna connection and SIM card activation in GSM network. |
| ● | ○ | ○ | ● | 3 | GPRS network error – check SIM card activation in GPRS network |
| ● | ● | ● | ○ | 4 | Wrong user name or password for GPRS network |
| ● | ○ | ● | ○ | 5 | Log in to GPRS error |
| ● | ● | ○ | ○ | 6 | Connection interrupted |
| ● | ○ | ○ | ○ | 7 | Other error |
| ○ | ● | ● | ● | 8 |  |
| ○ | ○ | ● | ● | 9 | SIM card error (locked or missing) |
| ○ | ● | ○ | ● | 10 |  |
| ○ | ○ | ○ | ● | 11 |  |
| ○ | ● | ● | ○ | 12 |  |
| ○ | ○ | ● | ○ | 13 | Number of log into GPRS trials exceeded |
| ○ | ● | ○ | ○ | 14 | Module blocked |
| ○ | ○ | ○ | ○ | 15 | Wrong PIN for SIM card |

| | |
|---|---|
| ○ | LED flashing (0,5Hz) |
| ● | LED off |

When GPRS LED is off the module is automatically trying reinitiate transmission.
When GPRS LED is flashing user intervention is required. Remove the reason for error and recycle the power.

## 10.3.2. Critical errors

A sign of **Critical error** is the flashing of **ERR LED**. Error code numbers are displayed on signal level and **GPRS** LEDs.

| ERR | State |
|---|---|
| ○● | flashes (0,5Hz) |

| GPRS | L1 | L2 | L3 | Error nr | Description |
|---|---|---|---|---|---|
| ● | ○ | ● | ● | 1 | BOOT module error |
| ● | ● | ○ | ● | 2 | Main program error |
| ● | ○ | ○ | ● | 3 | Update error |
| ● | ● | ● | ○ | 4 | RAM_N error in CPU |
| ● | ○ | ● | ○ | 5 | External RAM error |
| ● | ● | ○ | ○ | 6 | |
| ● | ○ | ○ | ○ | 7 | Stack overflow |
| ○ | ● | ● | ● | 8 | |
| ○ | ○ | ● | ● | 9 | |
| ○ | ● | ○ | ● | 10 | |
| ○ | ○ | ○ | ● | 11 | |
| ○ | ● | ● | ○ | 12 | |
| ○ | ○ | ● | ○ | 13 | |
| ○ | ● | ○ | ○ | 14 | |
| ○ | ○ | ○ | ○ | 15 | Undefined error |
| All LED indicators are flashing | | | | | RAM_T memory error in CPU |

| | |
|---|---|
| ○ | flashing LED (0,5Hz) |
| ● | LED off |

Occurrence of any errors described above indicates either a fault in program or a module defect. If a critical error occurs, please note the error code and contact the manufacturer.

## 11. Technical data

### 11.1. General

| | |
|---|---|
| Dimensions (height x width x length) | 105x86x60 mm |
| Weight | 300 g |
| Mounting method | DIN rail 35mm |
| Operating temperature | $-20^0 ... +55^0C$ |
| Protection class | IP40 |
| Max. potential on any connector referenced to GND | 60Vrms max. |
| Humidity | 5 … 95% non condensing |

## 11.2. GSM/GPRS Modem

Data for WISMO Quick 2406B

| Modem type | WISMO Quick 2406B |
|---|---|
| GSM | Dual Band GSM/GPRS module EGSM900/1800 |
| Frequency range (EGSM 900 MHz) | Transmitter: from 880 MHz to 915 MHz<br>Receiver: from 925 MHz to 960 MHz |
| Peak transmitting power (EGSM 900 MHz) | 33 dBm (2W) – class 4 station |
| Frequency range (EGSM 1800 MHz) | Transmitter: from 1710 MHz to 1785 MHz<br>Receiver: from 1805 MHz to 1880 MHz |
| Peak transmitting power (EGSM 1800 MHz) | 30 dBm (1W) – class 1 station |
| Modulation | 0,3 GMSK |
| Channel spacing | 200 kHz |
| Antenna | 50Ω |

Data for WISMO Quick Q24PLUS

| Modem type | WISMO Quick Q24Plus |
|---|---|
| GSM | Multiband GSM module (900/1800 or 850/1900)MHz |
| GPRS | Class 10 |
| Frequency range (EGSM 900 MHz) | Transmitter: from 880 MHz to 915 MHz<br>Receiver: from 925 MHz to 960 MHz |
| Frequency range (GSM 850 MHz) | Transmitter: from 824 MHz to 849 MHz<br>Receiver: from 869 MHz to 894 MHz |
| Peak transmitting power (EGSM 900 MHz & GSM 850 MHz) | 33 dBm (2W) – class 4 station |
| Frequency range (EGSM 1800 MHz) | Transmitter: from 1710 MHz to 1785 MHz<br>Receiver: from 1805 MHz to 1880 MHz |
| Frequency range (PCS 1900 MHz) | Transmitter: from 1850 MHz to 1910 MHz<br>Receiver: from 1930 MHz to 1990 MHz |
| Peak transmitting power (EGSM 1800 MHz & PCS 1900 MHz) | 30 dBm (1W) – class 1station |
| Modulation | 0,3 GMSK |
| Channel spacing | 200 kHz |
| Antenna | 50Ω |

## 11.3. Power supply

| Direct Current DC (12V, 24V) | 10,8 ... 36V |
|---|---|
| Alternating current AC (24V) | 18...26,4Vrms |
| Input current (A) (for 12V DC) | Idle 0,10<br>Active 0,60<br>Max 1,90 |
| Input current (A) (for 24V DC) | Idle 0,06<br>Active 0,25<br>Max 1,00 |

**CAUTION!**
**Due to high momentary current consumption the power supply must be capable of delivering >= 2A of current.**

**Inappropriate power supply may result in faulty operation or cause damage to MT-101 module!**

## 11.4. Drawings and dimensions

<div style="text-align:center">
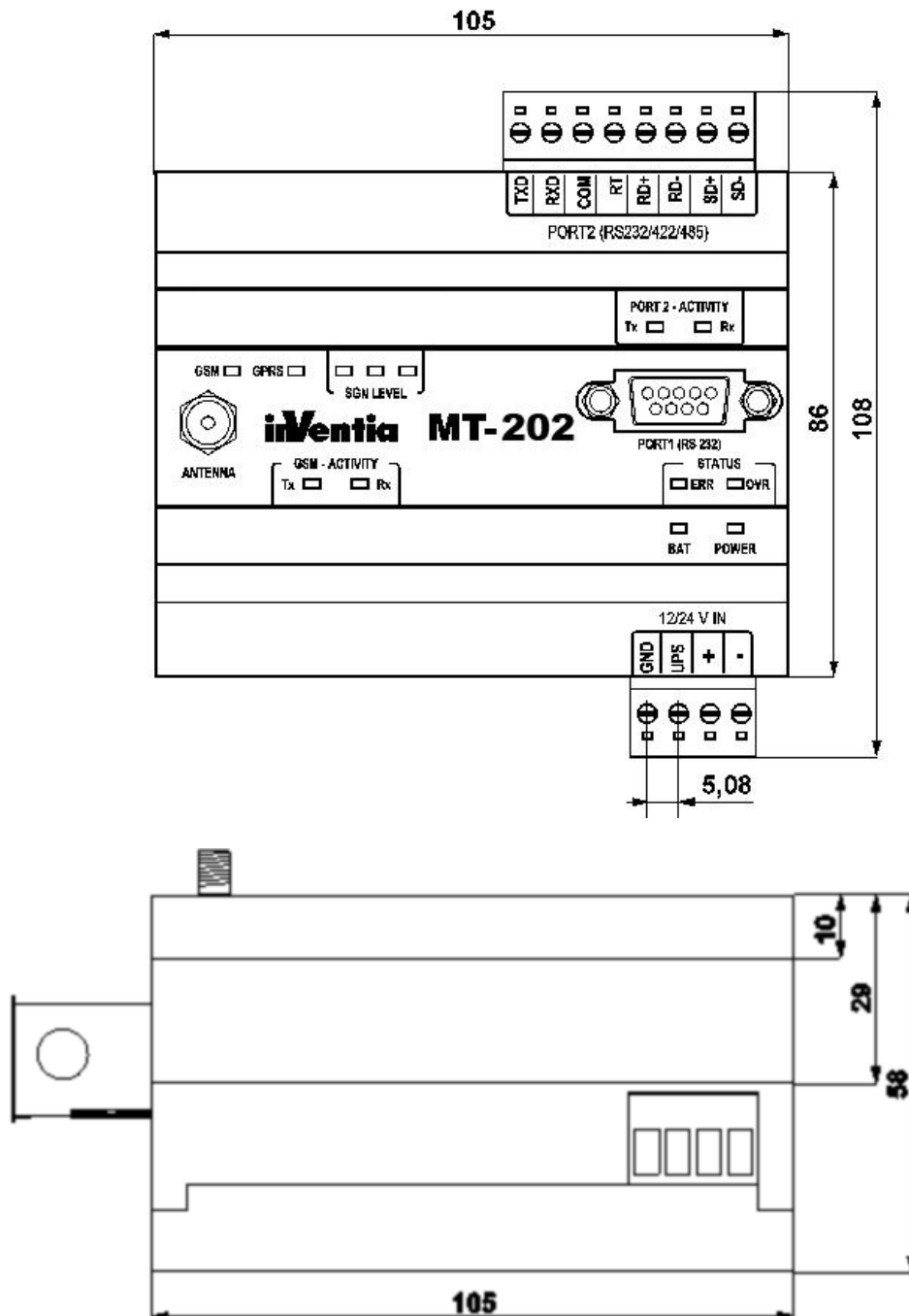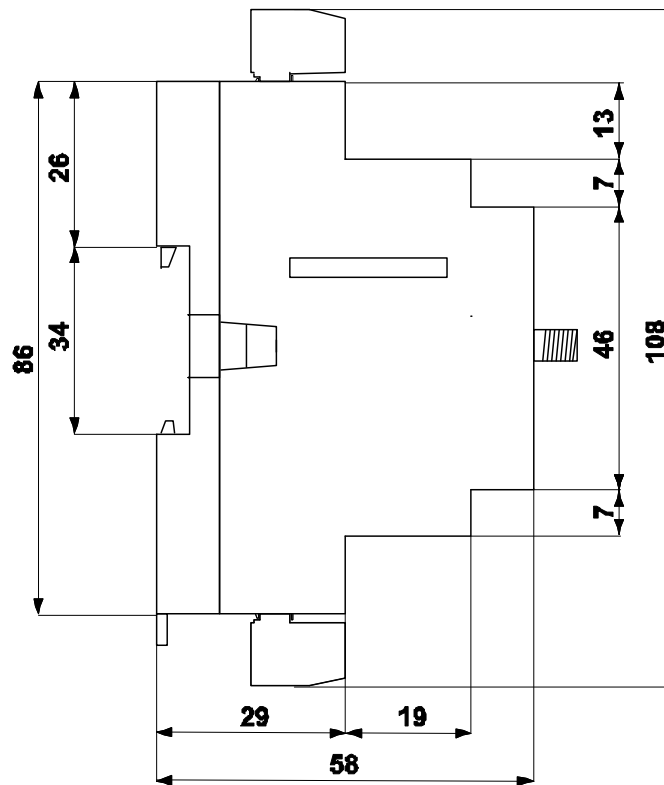
**NOTE!**
**All dimensions are in millimeters!**

</div>

## 12. Safety information

### 12.1. Working environment

When deploying telemetry modules one has to observe and comply to local legislation and regulations. Using the telemetry module in places where it can cause radio noise or other disturbances is strictly prohibited.

### 12.2. Electronic equipment

Thou most of modern electrical equipment is well RF (Radio Frequency) shielded there is no certainty that radio waves emitted by the telemetry module's antenna may have negative influence on its function.

#### 12.2.1. Heart pacemakers

It is recommended that the distance between the antenna of telemetry module and the Heart Pacemaker is greater than 20 cm. This distance is recommended by manufacturers of Pacemakers and in full harmony with results of studies conducted independently by Wireless Technology Research.

### 12.2.2. Hearing aids

In rare cases the signal emitted by the telemetry module's antenna may disturb hearing aids functions. Should that occur, one has to study detailed operating instructions and recommendations for that particular product.

### 12.2.3. Other medical equipment

Any radio device including the telemetry module may disturb the work of electronic medical equipment.
When there is a need of installing telemetry module in vicinity of medical equipment one has to contact the manufacturer of this equipment in order to make sure that the equipment is adequately protected against interference of radio frequency waves (RF).

### 12.2.4. RF Marked equipment

The restriction against installing telemetry modules in areas marked as radio frequency (RF) prohibition zones must be unconditionally observed.

## 12.3. Explosive environment

Installation of telemetry modules in the environment where explosion hazard is present is not permitted. Usually, but not always, these places are marked with warning signs. Where there is no marking do not install telemetry modules at liquid or gas fuels stores, inflammable materials stores, nor places contaminated with metal or wheat dust.

## 13. Appendices

## 13.1. Data transmission in GSM systems

At the moment, a subscriber to GSM services has 3 technologies of sending data at his disposal besides voice communication. These are: SMS, CSD and GPRS. Since they offer different functionalities a short description is necessary.

### 13.1.1. SMS

„**Short Message Service**" - the technology of sending text messages. The length of the message is 160 characters. Permitted characters are letters and numbers. Despite of its popularity it is not an optimal service for telemetry. The cost of an SMS is constant regardless of information length (within 160 characters limit).

In telemetric applications, using the text mode would require dedicated formatting and special programming for reception. The conclusion is simple. This service was not created for telemetry applications but may be an attractive supplement.
Suppose that along with monitoring current values, it is possible to receive perfectly readable textual information containing a warning about alarm situation.

It is quite a useful supplement.

### 13.1.2. CSD (HSCSD)

„**Circuit Switched Data**" - a technology for sending data via commuted communication channel set up on subscriber demand. Despite using a digital telephony technology the data transmission is analogical to traditional telephone modem.
The user establishes a connection with a defined subscriber number and carries out a transmission of data stream sent to the serial port connected to the GSM modem. It is a

point-to-point transmission where only 2 data sources are connected. After transmission, the connection is broken and the subscriber charged for the time occupying the channel. This type of data transmission justifiable when large amounts of data are to be transmitted but would be extremely expensive in real time monitoring since it occupies the channel all of the time.

### 13.1.3. GPRS

„**General Packet Radio Services**" – a technology of transmitting data as addressed digital packets. Seen from user's point of view it is identical to Internet technology. The technology uses packet data protocols, particularly UDP/IP and TCP/IP.

This technology is radically different from technologies employed by standard modems using GSM/CSD commuted mode.
The main difference is the inability to transmit the stream of data directly in traditional serial protocols.
For transmission by GSM/GPRS modem it is necessary to "package" data in frames compatible with employed protocol.
All procedures necessary for login to GPRS have to be completed, so connecting GSM/GPRS modem to the data source operating in serial protocol ( MODBUS, PPI, SNP, M-Bus....) incompatible with packet transmission specification is impossible - even if it has a serial interface.
After completing login sequence we have a connection equal to virtual "wired connection" available all of the time.

### 13.1.3.1. Advantages of GPRS technology

The absolutely greatest advantage of GPRS technology is the possibility for maintaining a permanent connection with the network while paying only for the volume of transmitted data with no charge for maintaining the connection. This makes establishing of "on line" services for minimal expenses possible. An additional advantage is potentially high transmission speed (up to ~170kb/s), facilitating high data volume transmission.
GPRS standard supports four channel encoding schemes named respectively CS1 to CS4 with throughput of
9,05 kb/s, 13,4 kb/s, 15,6 kb/s and 21,4 kb/s.
Reached this way, maximal transmission rates though rigidly defined are different depending on the number of connected channels and limited usually to max throughput of 115,2 kb/s (typically 8x13,4kb/s = 107,2 kb/s), and in particular situations even up to 171,2 kb/s (8x21,4=171,2).

### 13.1.3.2. GPRS in telemetry applications

GSM/GPRS technology is possibly the ideal solution for telemetry and control of dispersed objects.

The undisputable advantages are:

- Use of an existing advanced structure of GSM transmission structure.
- The gigantic range of the network – works everywhere!
- Low cost of establishing and utilizing the system
- No need for specialized antenna systems
- The possibility of building networked systems
- No necessity for retransmission
- Full access protection on operator and hardware level
- Cost of transmission system maintenance rests with network operator
- Easy rescaling and reconfiguration of the system

- Great availability of various receiving terminals
- Easy setup of temporary systems
- Only transmitted data volume is charged

For proper operation of terminals - GSM/GPRS network nodes, one needs a SIM card with GPRS service enabled, permission to log in to existing APN and a static IP assignment.
A static IP address is the base for addressing terminals in packet transmission GPRS networks.
Note that when using GPRS for real time monitoring, packet transmission networks add a delay dependent of the route the addressed packet has to go between the sender and recipient terminal. Usually this delay does not exceed a few seconds and is insignificant from a monitoring perspective. In turn, the possibility of creating networks independent of terrain topography and territorial size of the system is gained.

### 13.1.4. EDGE

**EDGE** (Enhanced Data rates for GSM Evolution) is a technology for data transmission employed in GSM networks.

This is an extension of GPRS technology (EDGE is also called for EGPRS - Enhanced GPRS), with enhanced radio interface allowing triple throughput (in most of current systems up to 236.8 kbit/s) and dynamic adjustment of packet transmission speed depending on transmission conditions.

### 13.1.5. UMTS

Universal Mobile Telecommunications System (**UMTS**) is a system of third generation cellular phone networks following 2G systems like GSM. The new radio interface significantly improved data transfer between the subscriber and the network boosting the quality of service (384 kbit/s throughput).

### 13.1.6. HSDPA

**HSDPA** (High Speed Downlink Packet Access) is a technology based on shared transmission channel. The main feature is dynamic adaptation to changes in radio environment and quick retransmission of faulty data. The HSDPA technology allows transmission from the network to device with 14,4 Mb/s speed.

## 13.2. Application examples

This chapter shows basic configurations of systems employing MT-202 modules.

### 13.2.1. Communication with single module

This is the simplest possible monitoring system based on transmission of SMS messages by MT modules in case of event defined during module configuration. In this mode, the module sends text SMS with static or dynamic content holding information on input/output states or content of module's internal registers. This mode may be employed in systems where continuous monitoring is not imperative but alert when defined event occurs or an answer for enquiry are wanted. SMS syntax is described in appendices.

In this system, the MT-202 module set to Modbus RTU Mirror mode can be used. The module generates events based on data read from slave device connected to PORT 2 and mirrored into modules internal registers. This method significantly extends the number of inputs/outputs responsible for generation of events
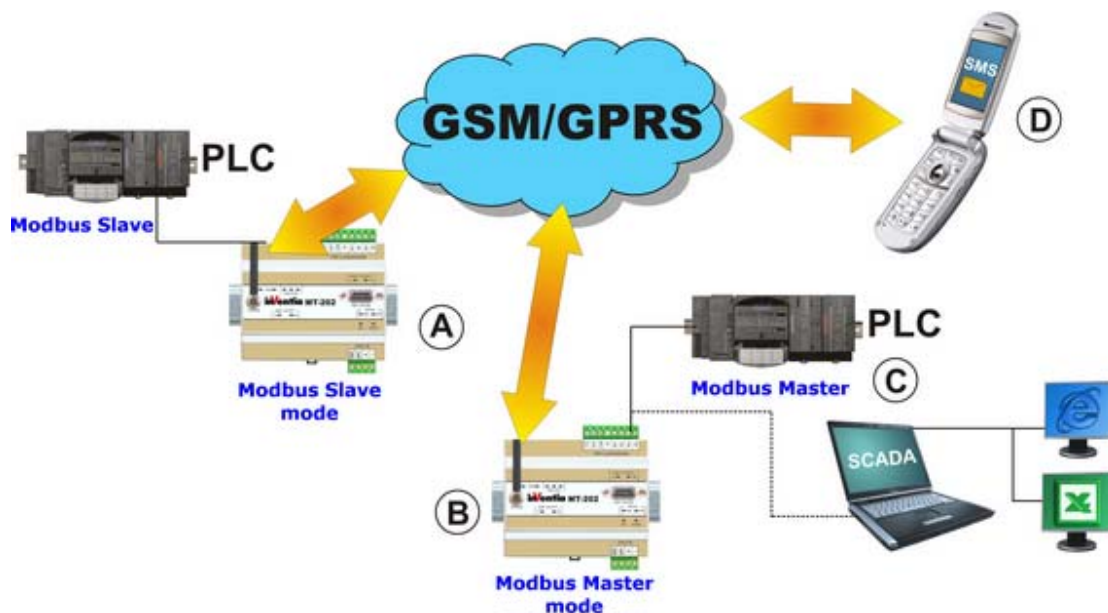


### 13.2.2. Point to point communication

This is a basic minimal configuration enabling data transmission between two devices. In this configuration, one can access internal module resources, discrete resources (inputs/outputs) or access resources of a device connected to module's serial port or mixed access to all above mentioned.

### 13.2.2.1. Using internal resources

This configuration requires two MT-202 modules, where one acts as a gateway to a transmission system (PLC, SCADA) set in Modbus Master mode. In this case, the device connected to MT-202 plays the role as master, sending inquiries about appropriate resources of remote MT-202 module. The module connected to a master gives access to its internal resources thus widening the central system.
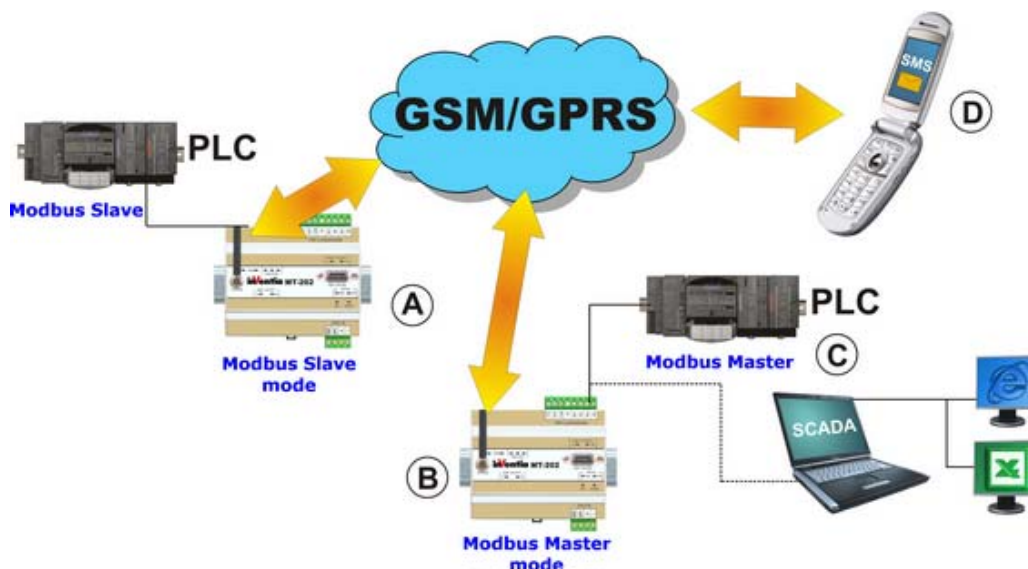
Internal configuration of modules is as follows:

**Module (A)**
Set Module operating mode to Modbus RTU Slave or MT Slave
Set Modbus ID number of the module to a value different than zero (f.e.1)
Set Authorized IP to 1 and add the IP assigned to module MT-202 operating in Modbus RTU Master mode (options allowing data transmission and reception have to be checked in).

**Module (B)**
Set Module operating mode to Modbus RTU Master
Set Modbus ID number of the module to a value different than zero (f.e.2)
Set Authorized IP to 1 and add the IP assigned to module MT-202 operating in Modbus RTU Master mode (options allowing data transmission and reception have to be checked in).
Fill in the Routing table setting IP address and Modbus ID assigned to MT-202 operating in Modbus RTU Slave (or MT Slave) mode.

The device playing role of master (C) should be connected to PORT2 of module (B) and query resources of address assigned to module Slave (A). The master can query module (B) provided that Modbus address of module (B) is sent along.

A system configured this way secures permanent access to all internal resources of the Slave including SMS services in situations defined during configuration. If the master unit is a PC running on Windows, the MT Data Provider (OPC Server) delivered with the module is the best solution for communication. MT Data Provider serves both querying and unsolicited messaging on the basis of Data sending rules defined during modules configuration.

Access to internal resources of modules in a telemetry network is similar to that described above. The only difference is the number of network nodes and assignment of unique Modbus IDs necessary for precise addressing of queries associated with it. Remember to put all slaves IP addresses into Master module's IP list along with corresponding Modbus ID and make sure that reading and writing is enabled.

### 13.2.2.2. Data transmission from external devices

#### 13.2.2.2.1. Configuration for transparent mode

Transparent mode does not require any configuration of internal resources. In this mode, all data arriving at PORT2 will be transmitted in accordance with recipients defined in IP number list as allowing transmission. One has to notice that queries are not routed to a particular telemetry module with attached slave but sent to all devices in the system without decoding of protocol. This allows for transmission of data in unknown protocol but increases costs of transmission since data is sent unselectively.

Reduction of transmission expenses is possible only by employing the function of transmission channel reservation.

#### 13.2.2.2.2. Configuration for GazModem mode

The system consisting of MT-202 modules in GazModem mode integrates dispersed objects in gas metering systems equipped with electronic counters into centralized measuring system.

After setting configuration parameters and connecting of device to PORT 2 of the module, the user obtains access to current data and alarms stored continuously in module memory. In this setup, MT-202 module may act as a concentrator for 16 devices.
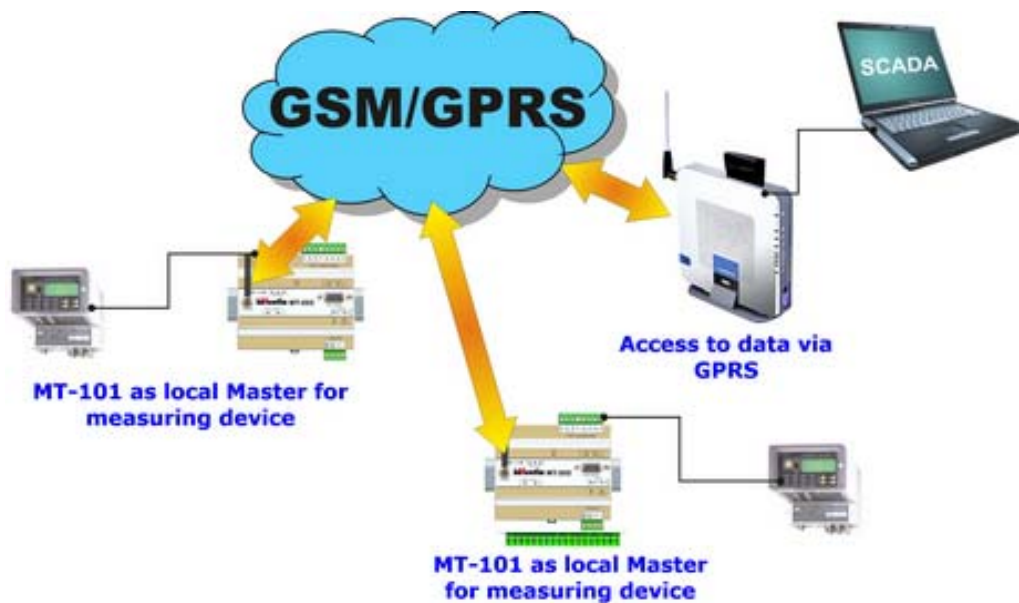


#### 13.2.2.2.3. Configuration for M-Bus Lec mode

The system consisting of MT-202 modules in Mbus LEC mode integrates dispersed objects in heat consumption metering systems equipped with electronic counters into a centralized measuring system. After setting configuration parameters and connecting of device to PORT 2 of the module, the user obtains access to current data and possibility of defining alarm thresholds continuously stored in module memory. In this setup, MT-202 module may act as a concentrator for 16 devices.

Additionally, in this mode connection of PORT1 to an auxiliary counter serving in Gaz-Modem protocol is possible.

13.2.2.2.4. Configuration for NMEA 0183 mode

In NMEA 0183 mode, MT-202 module monitors information delivered by weather stations like AIRMAR to PORT2, which is stored in register memory space. Beyond localization parameters from integrated GPS Receiver, AIRMAR station delivers parameters like: temperature, pressure, humidity, speed and direction of wind. Full range of variables delivered by this device is to be found in chapter additional resources for NMEA 0183 mode.
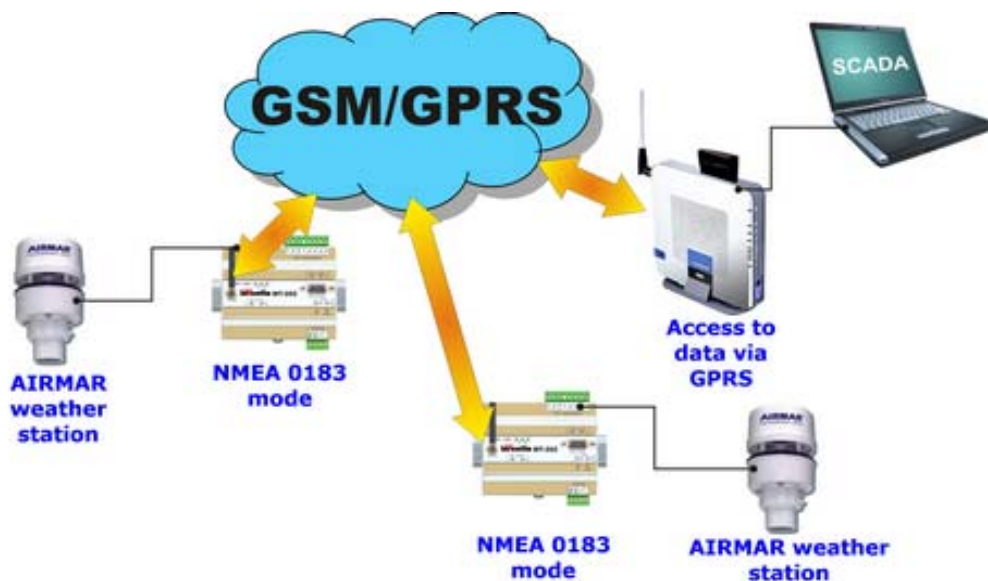


## 13.3. Syntax for reading and writing data in SMS mode

Basic syntax:

**Reading**
   *#[value representation][internal resource]/[% address space address]*

**Writing**
   *#[value representation][internal resource]/[% address space address]=value*

103

**Value representation** (the way of displaying or interpreting of the value):
D or none  decimally
H          hexadecimally (for registers always in four digits e.g. 002F)
B          binary (for registers always in 16 digits e.g. 0000000000010111)
S          string (read only). Presents ASCII text stored in registers. NULL (0x0000)
           character should be used as end of string marker.

**% address space address** - access to internal variables
I          bits in binary inputs space
Q          bits in binary outputs space
AI         Input Registers (16 bit)
R          Internal Registers (16 bit)
M          internal flags (Not deleted and zeroed during module reset)
S          internal flags (deleted and zeroed during module reset)

Examples of syntax that may along with simple text be a part of the SMS.

| | |
|---|---|
| #%R15 | value of Register 15 decimally |
| #H%AI4 | value of Input Register 4 |
| #%M123 | value of M123 flag |
| #H%R80#H%R81 | value of 32 bit Register DREG1 (memory map) 16 bit |
| #%R70=255 | setting Register R70 to value of 255 |
| #H%R70=FF | setting Register R70 to value 255 |
| #%R1000=2 | faulty resource address, replay „Err" |
| #%R70=100000 | faulty value, reply #%R70=Err |
| #S%R64 | text coded in ASCII beginning in internal register which address is 64 decimally and ending in first following register with NULL sign (0x0000). |

Notice:
- Valid syntax will be replaced in received SMS with current value of particular variable or else unchanged text will be returned.
- One SMS may read several variables.
- If length of received SMS (text + length of variables replacing mnemonics) exceeds 160 characters, (along with time stamp and optional status) the SMS will be truncated so that it ends with time stamp and optional status.
- Syntax is not case sensitive.
- Addresses in commands should be in decimal notation.
- 32 bit Registers occupy two cells in 16 bit register space. Access to 32 bit registers goes through 16 bit Registers.
- Commands modifying internal resources values are executable only for received SMS.
- Upon reception of SMS starting with $ sign activates "silent mode" and no confirmation is sent to originator.
- Confirmation SMS starts with '>' sign.

## 13.4. Unlocking writing to internal registers

Procedure when data writing protection is set to Yes :
When data overwriting protection option is set, the module does not process frames trying to change internal resources. Unblocking requires sending data along with password. If received frame contents a valid password, the module allows remote modification of internal resources for 5 minutes or until it receives a frame with empty or invalid password.

Command format:

| module's ID (1 byte) | command code (3 bytes) | Password (n bytes) | Password end (1 byte) | Modbus CRC (2 bytes) |
|---|---|---|---|---|
| ID | 0x71, 0x06 0x00 | Password text | 0x00 | CRC_L, CRC_H |

Example:

Module settings
ID=5
Password="ABCDE"
HEX:       0x05, 0x71, 0x06, 0x00, 'A', 'B', 'C', 'D', 'E', 0x00, 0x98, 0x70
Decimally: 5, 113, 6, 0, 'A', 'B', 'C', 'D', 'E', 0, 152, 112


## 13.5. **Working with dynamic IP addressing**

In order to configure MT-202 module to work in Proxy mode do following:

- In MTManager select Data frame format: Proxy
- Set Proxy server IP: has to be static, public IP address of central (receiving) computer
- In Authorized IP numbers type the serial number 255.255.255.255


The structure of configuration file for MT DataProvider for Proxy mode:

```
<?xml version="1.0"?>
<opc>

<configure net_mode="dynamic" udp_port="7110" timestamp="system"
csv_log="true" csv_path="C:\" debug="true"/>

<network name="mt202"
udp_port=""
ip_receiver=""
ip_header_receiver="011.004.006.002"
ip_header_sender="255.255.255.255"
timeout="10" retries="4" add_crc="true"
csv_msg_log="true" debug="false" enable="true">

<modbus name="id13" id="13" type="registers" address="0"
size="5" interval="25" debug="false" enable="true"/>

<modbus name="id13" id="13" type="binary_inputs" address="0"
size="8" interval="25" debug="false" enable="true"/>

</network>

</opc>
```

comment:
```
<!-- udp_port="" port assigned when remote module reports-->
<!-- ip_receiver="" IP assigned when remote module reports-->
<!-- ip_header_receiver="011.004.006.002" serial number of remote module-->
<!-- ip_header_sender="255.255.255.255" serial number of computer with public, static IP
]-->
```

## 13.6. Data formats

MT-202 module gives the user a choice of data frame format :

- standard - standard operating mode. The units communicate directly with each other in the APN using static IP addresses allocated permanently to used SIM cards. In this mode, IP addresses of communicating modules are written into authorized units lists.

- Open - The same configuration as standard. The only difference is lack of frame protection and opened header format of UDP frame allowing the user to create his own user access system.

UDP data frame format (port 7110)

| Header | | | Data block |
|---|---|---|---|
| Frame identifier 16 bit, 2 bytes (H,L) | Data block size in bytes, 2 bytes (H,L) | Frame identifier 16 bit, 2 bytes (H,L) | Data |

1. Frame identifier is used for data flow control, f.e. elimination of repeated frames. When sending consecutive data frames, this number shall be incremented.
2. Max. data block size is 1408 bytes.
3. For access to the module MODBUS frames are used. They are placed in data block without ending CRC.

| Data block (MODBUS frame) | | |
|---|---|---|
| Unit ID (1 byte) | Unit ID (1 byte) | Unit ID (1 byte) |

4. Upon reception of correct data frame the sender has to be informed by a receipt in a form of an UDP frame bearing only the header of received frame.

- Proxy - In this configuration, units communicate with each other through an external server. The server IP is written in Proxy server IP configuration variable. This mode allows using modules with SIM cards without assigned static address. Modules receive a randomly selected IP address during network login procedure. The Module establishes communication (sends and receives packets) only with Proxy server. Since modules in this mode are identified by serial numbers tables and lists describing, other modules in the network hold serial numbers instead of IP addresses. Dynamically addressed modules do not have the possibility of remote configuration or remote programming.

- UDP Standard - in this data module communicates using the Modbus frame encapsulated in standard UDP frame. This allows user to use Modbus/UDP drivers provided from other companies, but disables data delivery control system (confirmation of correct data frame reception mechanism). GPRS transmission retries number and GPRS transmission timeout parameters are unavailable in that operating mode.

## 13.7. Trigger inputs

During operation, the internal system of **MT-2O2** module creates a number of variables related to its inputs/outputs and to module diagnostics. Triggering inputs and triggering flags in conjunction with rules processing enable instantaneous reaction in occurring states.

User has access to following triggering inputs:

| input | Description |
|---|---|
| FS1_ups | = 1, no voltage on UPS pin |
| FS1_gprs | = 1, information of logging out of GPRS network |
| P1...P32 | Program flags P1...P32 (definable in user program) |
| TMR1...TMR4 | flags from Asynchronous clocks TMR1,TMR2 and synchronous TMR3, TMR4 |

## 13.8. Flags

During operation **MT-2O2** module governs a number of binary flags (assuming value *True* or *False*) that trigger rules processing and remote diagnostics.

The User has access to following flags:

| Flag | Resources attached | Description |
|---|---|---|
| Bi In 0->1 | FS1_ups FS1_gprs P1...P32 TMR1...TMR4 | Flag assuming value *True* after change of binary resource from 0 to 1 |
| Bi In 1->0 | | Flag assuming value *True* after change of binary resource from 1 to 0 |
| Bi In Chg | | Flag assuming value *True* after any change of binary input |

## 13.9. RM-12O

Converter module **RM-12O** is used as an intermediate in communication between telemetry module and the heat meter employing M-BUS transmission protocol. RM-120 performs the conversion of RS-232 signal to electrical M-Bus signal.

| Terminal | Description |
|----------|-------------|
| + (MBus) | Pin + M-Bus |
| - (MBus) | Pin - M-Bus |
| PE2 | ground for M-Bus circuit |
| - (24-36VDC) | converter module power supply (pin - ) |
| + (24-36VDC) | converter module power supply (pin + ) |
| TXD | RS232 sender output |
| RXD | RS232 sender input |
| COM | RS232 circuit ground |
| RTS | optional signaling for computers COM connection (not used for MT module) |
| PE1 | RS232 circuit ground |

### 13.10. Memory map

#### 13.10.1. Binary inputs space

**Binary inputs (bit addressable - command 02)**

| Address | + 0 | + 1 | + 2 | + 3 | + 4 | + 5 | + 6 | + 7 | Description | Virtual Registers |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | — | — | — | — | — | — | — | — | Reserved | VREG_BI0 |
| 0x0008 | — | — | — | — | — | — | — | — | Reserved | |
| 0x0010 | — | — | — | — | — | — | — | — | Reserved | VREG_BI1 |
| 0x0018 | — | — | — | — | — | — | — | — | Reserved | |
| 0x0020 | — | — | — | — | — | — | — | — | Reserved | VREG_BI2 |
| 0x0028 | FS1_fs | FS1_ovr | FS1_ups | — | FS1_prog | FS1_gprs | FS1_event | FS1_sms | **FS1_fs = 1 -** first cycle of the program **FS1_ovr = 1 -** delayed cycle start (previous cycle longer than 100ms) **FS1_ups = 1 -** no supply on UPS pin **FS1_prog = 1 -** error detected in user's program, program stopped. **FS1_gprs = 1 -** module logged out of GPRS. On logon the bit is reset. On power on FS1_gprs = 0 **FS1_event = 1 -** event queue overflow - data **FS1_sms = 1 -** event queue overflow - SMS | |
| 0x0030 | — | — | — | — | — | — | — | — | Reserved | VREG_BI3 |
| 0x0038 | — | — | — | — | — | — | — | — | Reserved | |

| 0x0040 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | Counter outputs C1 .. C8 | VREG_BI4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0048 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | Timer outputs T1 .. T8 | |
| 0x0050 | — | — | — | — | — | — | — | — | Reserved | |
| … | … | … | … | … | … | … | … | … | Reserved | |
| 0x0098 | — | — | — | — | — | — | — | — | Reserved | |
| 0x00A0 | TMR1 | TMR2 | TMR3 | TMR4 | 0 | 0 | 0 | 0 | User defined timer flip over info | VREG_BI10 |
| 0x00A8 | SL1_ok | SL2_ok | SL3_ok | SL4_ok | SL5_ok | SL6_ok | SL7_ok | SL8_ok | Serial communication status bits for Modbus Mirror, Macmat Slave GazModem and M-Bus LEC modes | |
| 0x00B0 | SL9_ok | SL10_ok | SL11_ok | SL12_ok | SL13_ok | SL14_ok | SL15_ok | SL16_ok | Serial communication status bits for Modbus Mirror mode | VREG_BI11 |
| 0x00B8 | — | — | — | — | — | — | — | — | — | |
| … | … | … | … | … | … | … | … | … | … | …. |
| 0x02B0 | — | — | — | — | — | — | — | — | — | VREG_BI43 |
| 0x02B8 | MT2MT_1 | MT2MT_2 | MT2MT_3 | MT2MT_4 | MT2MT_5 | MT2MT_6 | MT2MT_7 | MT2MT_8 | Bits informing about MT2MT buffer modification with values received with unsolicited messages. The bit number corresponds to the sender's index in Authorized numbers table in the module's configuration, thus referring to the sending module's IP address. Bits are set for one program cycle. | |
| 0x02C0 | MT2MT_9 | MT2MT_10 | MT2MT_11 | MT2MT_12 | MT2MT_13 | MT2MT_14 | MT2MT_15 | MT2MT_16 | | VREG_BI44 |
| 0x02C8 | MT2MT_17 | MT2MT_18 | MT2MT_19 | MT2MT_20 | MT2MT_21 | MT2MT_22 | MT2MT_23 | MT2MT_24 | | |
| 0x02D0 | MT2MT_25 | MT2MT_26 | MT2MT_27 | MT2MT_28 | MT2MT_29 | MT2MT_30 | MT2MT_31 | MT2MT_32 | | VREG_BI45 |
| 0x02D8 | MT2MT_33 | MT2MT_34 | MT2MT_35 | MT2MT_36 | MT2MT_37 | MT2MT_38 | MT2MT_39 | MT2MT_40 | | |
| 0x02E0 | MT2MT_41 | MT2MT_42 | MT2MT_43 | MT2MT_44 | MT2MT_45 | MT2MT_46 | MT2MT_47 | MT2MT_48 | | VREG_BI46 |
| 0x02E8 | MT2MT_49 | MT2MT_50 | MT2MT_51 | MT2MT_52 | MT2MT_53 | MT2MT_54 | MT2MT_55 | MT2MT_56 | | |
| 0x02F0 | MT2MT_57 | MT2MT_58 | MT2MT_59 | MT2MT_60 | MT2MT_61 | MT2MT_62 | MT2MT_63 | MT2MT_64 | | VREG_BI47 |
| 0x02F8 | MT2MT_65 | MT2MT_66 | MT2MT_67 | MT2MT_68 | MT2MT_69 | MT2MT_70 | MT2MT_71 | MT2MT_72 | | |
| 0x0300 | MT2MT_73 | MT2MT_74 | MT2MT_75 | MT2MT_76 | MT2MT_77 | MT2MT_78 | MT2MT_79 | MT2MT_80 | | VREG_BI48 |
| 0x0308 | MT2MT_81 | MT2MT_82 | MT2MT_83 | MT2MT_84 | MT2MT_85 | MT2MT_86 | MT2MT_87 | MT2MT_88 | | |
| 0x0310 | MT2MT_89 | MT2MT_90 | MT2MT_91 | MT2MT_92 | MT2MT_93 | MT2MT_94 | MT2MT_95 | MT2MT_96 | | VREG_BI49 |
| 0x0318 | MT2MT_97 | MT2MT_98 | MT2MT_99 | MT2MT_100 | MT2MT_101 | MT2MT_102 | MT2MT_103 | MT2MT_104 | | |
| 0x0320 | MT2MT_105 | MT2MT_106 | MT2MT_107 | MT2MT_108 | MT2MT_109 | MT2MT_110 | MT2MT_111 | MT2MT_112 | | VREG_BI50 |
| 0x0328 | MT2MT_113 | MT2MT_114 | MT2MT_115 | MT2MT_116 | MT2MT_117 | MT2MT_118 | MT2MT_119 | MT2MT_120 | | |
| 0x0330 | MT2MT_121 | MT2MT_122 | MT2MT_123 | MT2MT_124 | MT2MT_125 | MT2MT_126 | MT2MT_127 | MT2MT_128 | | VREG_BI51 |

| Addr | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0338 | MT2MT_129 | MT2MT_130 | MT2MT_131 | MT2MT_132 | MT2MT_133 | MT2MT_134 | MT2MT_135 | MT2MT_136 | | |
| 0x0340 | MT2MT_137 | MT2MT_138 | MT2MT_139 | MT2MT_140 | MT2MT_141 | MT2MT_142 | MT2MT_143 | MT2MT_144 | | VREG_BI52 |
| 0x0348 | MT2MT_145 | MT2MT_146 | MT2MT_147 | MT2MT_148 | MT2MT_149 | MT2MT_150 | MT2MT_151 | MT2MT_152 | | |
| 0x0350 | MT2MT_153 | MT2MT_154 | MT2MT_155 | MT2MT_156 | MT2MT_157 | MT2MT_158 | MT2MT_159 | MT2MT_160 | | VREG_BI53 |
| 0x0358 | MT2MT_161 | MT2MT_162 | MT2MT_163 | MT2MT_164 | MT2MT_165 | MT2MT_166 | MT2MT_167 | MT2MT_168 | | |
| 0x0360 | MT2MT_169 | MT2MT_170 | MT2MT_171 | MT2MT_172 | MT2MT_173 | MT2MT_174 | MT2MT_175 | MT2MT_176 | | VREG_BI54 |
| 0x0368 | MT2MT_177 | MT2MT_178 | MT2MT_179 | MT2MT_180 | MT2MT_181 | MT2MT_182 | MT2MT_183 | MT2MT_184 | | |
| 0x0370 | MT2MT_185 | MT2MT_186 | MT2MT_187 | MT2MT_188 | MT2MT_189 | MT2MT_190 | MT2MT_191 | MT2MT_192 | | VREG_BI55 |
| 0x0378 | MT2MT_193 | MT2MT_194 | MT2MT_195 | MT2MT_196 | MT2MT_197 | MT2MT_198 | MT2MT_199 | MT2MT_200 | | |
| 0x0380 | MT2MT_201 | MT2MT_202 | MT2MT_203 | MT2MT_204 | MT2MT_205 | MT2MT_206 | MT2MT_207 | MT2MT_208 | | VREG_BI56 |
| 0x0388 | MT2MT_209 | MT2MT_210 | MT2MT_211 | MT2MT_212 | MT2MT_213 | MT2MT_214 | MT2MT_215 | MT2MT_216 | | |
| 0x0390 | MT2MT_217 | MT2MT_218 | MT2MT_219 | MT2MT_220 | MT2MT_221 | MT2MT_222 | MT2MT_223 | MT2MT_224 | | VREG_BI57 |
| 0x0398 | MT2MT_225 | MT2MT_226 | MT2MT_227 | MT2MT_228 | MT2MT_229 | MT2MT_230 | MT2MT_231 | MT2MT_232 | | |
| 0x03A0 | MT2MT_233 | MT2MT_234 | MT2MT_235 | MT2MT_236 | MT2MT_237 | MT2MT_238 | MT2MT_239 | MT2MT_240 | | VREG_BI58 |
| 0x03A8 | MT2MT_241 | MT2MT_242 | MT2MT_243 | MT2MT_244 | MT2MT_245 | MT2MT_246 | MT2MT_247 | MT2MT_248 | | |
| 0x03B0 | MT2MT_249 | MT2MT_250 | MT2MT_251 | MT2MT_252 | MT2MT_253 | MT2MT_254 | MT2MT_255 | MT2MT_256 | | |
| 0x03B8 | --- | FS2_apn | 0 | 0 | 0 | 0 | FS2_new | FS2_stop | **FS2_new** - informs on downloading of new program. This flag is reset at every program stop or power up. **FS2_stop** - Informs that program was stopped. This flag is reset only on power up or download of new program. **FS2_apn** - 1 reflects APN login state, 0 - logged out | VREG_BI59 |

## 13.10.2. Binary outputs space

**Binary outputs (bit addressable - command 02)**

| Address | + 0 | + 1 | + 2 | + 3 | + 4 | + 5 | + 6 | + 7 | Description | Virtual registers |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | — | — | — | — | — | — | — | — | Reserved | VREG_BO0 |
| 0x0008 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | Program flags enabling rule based data transmission on event. | |
| 0x0010 | P9 | P10 | P11 | P12 | P13 | P14 | P15 | P16 | | VREG_BO1 |
| 0x0018 | CLK_C1 | CLK_C2 | CLK_C3 | CLK_C4 | CLK_C5 | CLK_C6 | CLK_C7 | CLK_C8 | Counting inputs C1..C8 (raising edge) | |
| 0x0020 | RST_C1 | RST_C2 | RST_C3 | RST_C4 | RST_C5 | RST_C6 | RST_C7 | RST_C8 | Counter resetting inputs C1..C8 (active state 1) | VREG_BO2 |
| 0x0028 | EN_T1 | EN_T2 | EN_T3 | EN_T4 | EN_T5 | EN_T6 | EN_T7 | EN_T8 | Strobe inputs for timers T1..T8 (active state 1) | |
| 0x0030 | RST_T1 | RST_T2 | RST_T3 | RST_T4 | RST_T5 | RST_T6 | RST_T7 | RST_T8 | Reset inputs for timers T1..T8 (active state 1) | VREG_BO3 |
| 0x0038 | PS1_stop | — | GPRS_data | — | P2RCV_err | P2SND_err | — | — | **PS1_stop** = 1 - Program stop **GPRS_data** - bit set at every reception of a data frame via GPRS **P2RCV_err** = 1 - byte reception error (f.e. transmission parameter or buffer overflow), reset by user program. **P2SND_err** = 1 - buffer transmission error (f.e. wrong size), reset by user program. | |
| 0x0040 | P17 | P18 | P19 | P20 | P21 | P22 | P23 | P24 | Program flags enabling rule based data transmission on event. | VREG_BO4 |
| 0x0048 | P25 | P26 | P27 | P28 | P29 | P30 | P31 | P32 | | |
| 0x0050 | BU80 | BU81 | BU82 | BU83 | BU84 | BU85 | BU86 | BU87 | General purpose user fags | VREG_BO5 |
| 0x0058 | BU88 | BU89 | BU90 | BU91 | BU92 | BU93 | BU94 | BU95 | General purpose user fags | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | General purpose user fags | ... |
| 0x00F0 | BU240 | BU241 | BU242 | BU243 | BU244 | BU245 | BU246 | BU247 | General purpose user fags | VREG_BO15 |
| 0x00F8 | BU248 | BU249 | BU250 | BU251 | BU252 | BU253 | BU254 | BU255 | General purpose user fags | |

## 13.10.3. Analogue inputs space

**Input Registers (16 bit - command 04H)**

| Address | Description | Symbol |
|---------|-------------|--------|
| 0x0000 | — | Reserved |
| … | … | Reserved |
| 0x0005 | — | Reserved |
| 0x0006 | RTC - seconds (00…59) | RTC_Sec |
| 0x0007 | RTC - minutes (00…59) | RTC_Min |
| 0x0008 | RTC - hours (00…23) | RTC_Hour |
| 0x0009 | RTC - day of week (1 - Sunday, 7 - Saturday) | RTC_DofW |
| 0x000A | RTC - day of month (1…31) | RTC_Day |
| 0x000B | RTC - month (1…12) | RTC_Mon |
| 0x000C | RTC - year (2000…2099) | RTC_Year |
| 0x000D | Number of sent bytes for GPRS since power up (32 bit unsigned value) | (high 16 bit) |
| 0x000E | | (low 16 bit) |
| 0x000F | Number of received bytes for GPRS since power up (32 bit unsigned value) | (high 16 bit) |
| 0x0010 | | (low 16 bit) |
| 0x0011 | — | — |
| … | … | … |
| 0x0070 | — | — |
| 0x0072 | GPRS statistics- Transmission | (high 16 bits) |
| 0x0073 | Number of frames Type 0 (retries not accounted) | (low 16 bits) |
| 0x0074 | GPRS statistics - Reception | (high 16 bits) |
| 0x0075 | Number of retries | (low 16 bits) |
| 0x0076 | GPRS statistics- Transmission | (high 16 bits) |
| 0x0077 | Number of not delivered (not confirmed) frames | (low 16 bits) |
| 0x0078 | GPRS statistics - Reception | ((high 16 bits) |
| 0x0079 | Frames Type 0 | (low 16 bits) |
| 0x007A | GPRS statistics - Reception | (high 16 bits) |
| 0x007B | Frames Type 1 | (low 16 bits) |
| 0x007C | GPRS statistics - Reception | (high 16 bits) |
| 0x007D | Rejected frames (Type 0) - module busy | (low 16 bits) |
| 0x007E | Number of attempts since power up | informing on failed access attempts due to invalid password |
| 0x007F | Last attempt time - Year | |
| 0x0080 | Last attempt time - Month | |
| 0x0081 | Last attempt time - Day | |
| 0x0082 | Last attempt time - hours | |
| 0x0083 | Last attempt time - minutes | |
| 0x0084 | GSM signal strength (Max value 188) | GSM_SGN_LEV (word) |
| 0x0085 | Firmware version (y.xx) y - High byte, xx - Low byte | FIRMWARE_VER (word) |
| 0x0086 | Reserved for GAZMODEM extension in MBUS_LEC mode | |
| … | … | |
| 0x00A7 | Reserved for GAZMODEM extension in MBUS_LEC mode | |
| | | |
| 0x00BC | Number of program lines executed in previous cycle | PRG_CLINE (word) |
| 0x00BD | Elapsed program execution time in previous cycle | PRG_CTIME (word) |
| 0x00BE | Help register for function in user program | AUX_RET1 (word) |
| 0x00BF | Help register for function in user program | AUX_RET2 (word) |
| 0x00C0 | Parameter 1 | PAR_1 (word) |

| | | | |
|---|---|---|---|
| ... | ... | | ... |
| 0x00FF | Parameter 64 | | PAR_64 (word) |
| 0x0100 | --- | | --- |
| ... | ... | | ... |
| 0x01FF | --- | | --- |
| ... | ... | | ... |
| 0x500 | Parameter 65 | | PAR_64 (word) |
| ... | ... | | ... |
| 0x53F | Parameter 128 | | PAR_128 (word) |

## 13.10.4. Internal Registers space

**Internal registers space (read command 03H, write 06H or 10H)**
**(Not zeroed at reset)**

| Address | Description | Symbol | HIGH byte | LOW byte |
|---|---|---|---|---|
| 0x0000 | — | Reserved | — | |
| ... | ... | Reserved | ... | |
| 0x001F | — | Reserved | — | |
| 0x0020 | 16 bit counter - C1 (threshold value) | PV_C1 | 16 bit value | |
| 0x0021 | 16 bit counter - C2 (threshold value) | PV_C2 | 16 bit value | |
| 0x0022 | 16 bit counter - C3 (threshold value) | PV_C3 | 16 bit value | |
| 0x0023 | 16 bit counter - C4 (threshold value) | PV_C4 | 16 bit value | |
| 0x0024 | 16 bit counter - C5 (threshold value) | PV_C5 | 16 bit value | |
| 0x0025 | 16 bit counter - C6 (threshold value) | PV_C6 | 16 bit value | |
| 0x0026 | 16 bit counter - C7 (threshold value) | PV_C7 | 16 bit value | |
| 0x0027 | 16 bit counter - C8 (threshold value) | PV_C8 | 16 bit value | |
| 0x0028 | 16 bit Timer - T1 (threshold value) | PV_T1 | 16 bit value | |
| 0x0029 | 16 bit Timer - T2 (threshold value) | PV_T2 | 16 bit value | |
| 0x002A | 16 bit Timer - T3 (threshold value) | PV_T3 | 16 bit value | |
| 0x002B | 16 bit Timer - T4 (threshold value) | PV_T4 | 16 bit value | |
| 0x002C | 16 bit Timer - T5 (threshold value) | PV_T5 | 16 bit value | |
| 0x002D | 16 bit Timer - T6 (threshold value) | PV_T6 | 16 bit value | |
| 0x002E | 16 bit Timer - T7 (threshold value) | PV_T7 | 16 bit value | |
| 0x002F | 16 bit Timer - T8 (threshold value) | PV_T8 | 16 bit value | |
| 0x0030 | 16 bit counter - C1 (current value) | REG_C1 | 16 bit value | |
| 0x0031 | 16 bit counter - C2 (current value) | REG_C2 | 16 bit value | |
| 0x0032 | 16 bit counter - C3(current value) | REG_C3 | 16 bit value | |
| 0x0033 | 16 bit counter - C4 (current value) | REG_C4 | 16 bit value | |
| 0x0034 | 16 bit counter - C5 (current value) | REG_C5 | 16 bit value | |
| 0x0035 | 16 bit counter - C6 (current value) | REG_C6 | 16 bit value | |
| 0x0036 | 16 bit counter - C7 (current value) | REG_C7 | 16 bit value | |
| 0x0037 | 16 bit counter - C8 (current value) | REG_C8 | 16 bit value | |
| 0x0038 | 16 bit Timer - T1 (current value) | REG_T1 | 16 bit value | |
| 0x0039 | 16 bit Timer - T2 (current value) | REG_T2 | 16 bit value | |
| 0x003A | 16 bit Timer - T3 (current value) | REG_T3 | 16 bit value | |
| 0x003B | 16 bit Timer - T4 (current value) | REG_T4 | 16 bit value | |
| 0x003C | 16 bit Timer - T5 (current value) | REG_T5 | 16 bit value | |
| 0x003D | 16 bit Timer - T6 (current value) | REG_T6 | 16 bit value | |
| 0x003E | 16 bit Timer - T7 (current value) | REG_T7 | 16 bit value | |
| 0x003F | 16 bit Timer - T8 (current value) | REG_T8 | 16 bit value | |
| 0x0040 | 16 bit Program register (unsigned value) | REG1 | 16 bit value | |
| 0x0041 | 16 bit Program register (unsigned value) | REG2 | 16 bit value | |
| 0x0042 | 16 bit Program register (unsigned value) | REG3 | 16 bit value | |
| 0x0043 | 16 bit Program register (unsigned value) | REG4 | 16 bit value | |

| 0x0044 | 16 bit Program register (unsigned value) | REG5 | 16 bit value | |
|---|---|---|---|---|
| 0x0045 | 16 bit Program register (unsigned value) | REG6 | 16 bit value | |
| 0x0046 | 16 bit Program register (unsigned value) | REG7 | 16 bit value | |
| 0x0047 | 16 bit Program register (unsigned value) | REG8 | 16 bit value | |
| 0x0048 | 16 bit Program register (unsigned value) | REG9 | 16 bit value | |
| 0x0049 | 16 bit Program register (unsigned value) | REG10 | 16 bit value | |
| 0x004A | 16 bit Program register (unsigned value) | REG11 | 16 bit value | |
| 0x004B | 16 bit Program register (unsigned value) | REG12 | 16 bit value | |
| 0x004C | 16 bit Program register (unsigned value) | REG13 | 16 bit value | |
| 0x004D | 16 bit Program register (unsigned value) | REG14 | 16 bit value | |
| 0x004E | 16 bit Program register (unsigned value) | REG15 | 16 bit value | |
| 0x004F | 16 bit Program register (unsigned value) | REG16 | 16 bit value | |
| 0x0050 | 32 bit Program register (signed value) | DREG1 | (High 16 bits) | |
| 0x0051 | | | (Low 16 bits) | |
| 0x0052 | 32 bit Program register (signed value) | DREG2 | (High 16 bits) | |
| 0x0053 | | | (Low 16 bits) | |
| 0x0054 | 32 bit Program register (signed value) | DREG3 | (High 16 bits) | |
| 0x0055 | | | (Low 16 bits) | |
| 0x0056 | 32 bit Program register (signed value) | DREG4 | (High 16 bits) | |
| 0x0057 | | | (Low 16 bits) | |
| 0x0058 | 32 bit Program register (signed value) | DREG5 | (High 16 bits) | |
| 0x0059 | | | (Low 16 bits) | |
| 0x005A | 32 bit Program register (signed value) | DREG6 | (High 16 bits) | |
| 0x005B | | | (Low 16 bits) | |
| 0x005C | 32 bit Program register (signed value) | DREG7 | (High 16 bits) | |
| 0x005D | | | (Low 16 bits) | |
| 0x005E | 32 bit Program register (signed value) | DREG8 | (High 16 bits) | |
| 0x005F | | | (Low 16 bits) | |
| 0x0060 | 16 bit Program register (unsigned value) | XREG1 | 16 bit value | |
| ... | ... | ... | ... | |
| 0x025F | 16 bit Program register (unsigned value) | XREG512 | 16 bit value | |
| 0x0260 | — | Reserved | — | |
| ... | ... | Reserved | ... | |
| 0x0271 | — | Reserved | — | |
| | | | | |

**Registers holding last received via GPRS status of remote module**

| 0x0272 | Input space | RMT_IN | I8..I1 | IQ8..IQ1 |
|---|---|---|---|---|
| 0x0273 | Remote module ID + output space | RMT_ID_OUT | ID | Q8..Q1 |
| 0x0274 | Input AN1 | RMT_AN1 | 16 bit value | |
| 0x0275 | Input AN2 | RMT_AN2 | 16 bit value | |
| ... | ... | ... | ... | |
| 0x0280 | 32 bit activity time counter on input Q1 [s] | CNT_ON_Q1 | (High 16 bits) | |
| 0x0281 | 32 bit activity time counter on input Q1 [s] | | (Low 16 bits) | |
| 0x0282 | 32 bit activity time counter on input Q2 [s] | CNT_ON_Q2 | (High 16 bits) | |
| 0x0283 | 32 bit activity time counter on input Q2 [s] | | (Low 16 bits) | |
| ... | ... | ... | ... | |
| 0x028E | 32 bit activity time counter on input Q8 [s] | CNT_ON_Q8 | (High 16 bits) | |
| 0x028F | 32 bit activity time counter on input Q8 [s] | | (Low 16 bits) | |
| 0x0290 | 32 bit activity time counter on input I1 [s] | CNT_ON_I1 | (High 16 bits) | |
| 0x0291 | 32 bit activity time counter on input I1 [s] | | (Low 16 bits) | |
| ... | ... | ... | ... | |
| 0x029E | 32 bit activity time counter on input I8 [s] | CNT_ON_I8 | (High 16 bits) | |
| 0x029F | 32 bit activity time counter on input I8 [s] | | (Low 16 bits) | |

| Mirror | Device status - Mirror | | HIGH byte | LOW byte |
|---|---|---|---|---|
| 0x03E4 | Inputs space | MT_IN | I8..I1 | IQ8..IQ1 |
| 0x03E5 | Outputs space | MT_OUT | 0..0 | Q8..Q1 |
| 0x03E6 | Input AN1 (copy of input register 0x0004) | MT_AN1 | 16 bit value | |
| 0x03E7 | Input AN2 (copy of input register 0x0005) | MT_AN2 | 16 bit value | |
| | **Registers for FlexSerial mode data services** | | | |
| 0x03FE | Data size in reception buffer | P2RCV_NO | 0..512 | |
| 0x03FF | Data size to be sent from transmission buffer | P2SND_NO | 0..512 | |
| 0x0400 | reception buffer | P2RCV_B1 | — | byte 1 |
| 0x0401 | reception buffer | P2RCV_B2 | — | byte 2 |
| … | … | … | …… | |
| 0x05FF | reception buffer | P2RCV_B512 | — | byte 512 |
| 0x0600 | transmission buffer | P2SND_B1 | — | byte 1 |
| 0x0601 | transmission buffer | P2SND_B2 | — | byte 2 |
| … | … | … | … | … |
| 0x07FF | transmission buffer | P2SND_B512 | — | byte 512 |

| RTC | **Registers for module time modification (for block writing only, command 0x10)** | | | |
|---|---|---|---|---|
| 0x2700 | RTC – seconds (00…59) | | 16 bit value | |
| 0x2701 | RTC – minutes (00…59) | | 16 bit value | |
| 0x2702 | RTC – hours (00…23) | | 16 bit value | |
| 0x2703 | RTC – day of week (1 - Saturday, 7 – Sunday) | | 16 bit value | |
| 0x2704 | RTC – day of month (1…31) | | 16 bit value | |
| 0x2705 | RTC – month (1…12) | | 16 bit value | |
| 0x2706 | RTC - year (2000…2099) | | 16 bit value | |
| 0x2707 | RTC - negated bits of RTC registers +1 (protection against accidental RTC settings modification) | | 16 bit value | |

## 13.10.5. Auxiliary resources for GazModem mode

**Input registers space**
**Registers holding signals read from gas counters.**

| Start Address | | | Name | Description |
|---|---|---|---|---|
| hex | dec | MODBUS | | |
| 0x31 | 049 | 30050 | MC1SYG1 | High byte – status, Low - signals |
| 0x32 | 050 | 30051 | MC2SYG1 | High byte – status, Low - signals |
| … | … | … | … | … |
| 0x40 | 64 | 30065 | MC16SYG1 | High byte – status, Low - signals |
| 0x41 | 65 | 30066 | MC1SYG2 | High byte – status, Low - signals |
| … | … | … | … | … |
| 0x50 | 80 | 30081 | MC16SYG2 | High byte – status, Low - signals |
| 0x51 | 81 | 30082 | MC1SYG3 | High byte – status, Low - signals |
| … | … | … | … | … |
| 0x60 | 96 | 30097 | MC16SYG3 | High byte – status, Low - signals |
| 0x61 | 97 | 30098 | MC1SYG4 | High byte – status, Low - signals |
| … | … | … | … | … |
| 0x70 | 112 | 30113 | MC16SYG4 | High byte – status, Low - signals |

**Registers holding current data read from gas counters (up to 8 variables from one counter).**

| Start Address | | | Name | Description |
|---|---|---|---|---|
| Hex | dec | MODBUS | | |
| 0x100 | 256 | 30257 | MC1VAR1_H | 32 bit (H..L) floating point |
| 0x101 | 257 | 30258 | MC1VAR1_L | |
| 0x102 | 258 | 30259 | MC1VAR2_H | 32 bit (H..L) floating point |
| 0x103 | 259 | 30260 | MC1VAR2_L | |
| 0x104 | 260 | 30261 | MC1VAR3_H | 32 bit (H..L) floating point |
| 0x105 | 261 | 30262 | MC1VAR3_L | |
| ... | ... | ... | ... | ... |
| 0x10E | 270 | 30271 | MC1VAR8_H | 32 bit (H..L) floating point |
| 0x10F | 271 | 30272 | MC1VAR8_L | |
| 0x110 | 272 | 30273 | MC2VAR1_H | 32 bit (H..L) floating point |
| 0x111 | 273 | 30274 | MC2VAR1_L | |
| ... | ... | ... | ... | ... |
| 0x11E | 286 | 30287 | MC2VAR8_H | 32 bit (H..L) floating point |
| 0x11F | 287 | 30288 | MC2VAR8_L | |
| ... | ... | ... | ... | ... |
| 0x1FE | 510 | 30511 | MC16VAR8_H | 32 bit (H..L) floating point |
| 0x1FF | 511 | 30512 | MC16VAR8_L | |

**Holding registers space**
**Registers threshold values HH, H, L, LL for current values. Alarm bits informing on comparison results are in binary inputs space.**

| Start Address | | | Name | Description |
|---|---|---|---|---|
| Hex | dec | MODBUS | | |
| 0x400 | 1024 | 41025 | MC1LL1_H | LL threshold for VAR1 of counter 1 |
| 0x401 | 1025 | 41026 | MC1LL1_L | 32 bit (H..L) floating point |
| 0x402 | 1026 | 41027 | MC1L1_H | L threshold for VAR1 of counter 1 |
| 0x403 | 1027 | 41028 | MC1L1_L | 32 bit (H..L) floating point |
| 0x404 | 1028 | 41029 | MC1H1_H | H threshold for VAR1 of counter 1 |
| 0x405 | 1029 | 41030 | MC1H1_L | 32 bit (H..L) floating point |
| 0x406 | 1030 | 41031 | MC1HH1_H | HH threshold for VAR1 of counter 1 |
| 0x407 | 1031 | 41032 | MC1HH1_L | 32 bit (H..L) floating point |
| 0x408 | 1032 | 41033 | MC1LL2_H | LL threshold for VAR1 of counter 1 |
| 0x409 | 1033 | 41034 | MC1LL2_L | 32 bit (H..L) floating point |
| ... | ... | ... | ... | ... |
| 0x40E | 1038 | 41039 | MC1HH2_H | HH threshold for VAR2 of counter 1 |
| 0x40F | 1039 | 41040 | MC1HH2_L | 32 bit (H..L) floating point |
| 0x410 | 1040 | 41041 | MC1LL3_H | LL threshold for VAR3 of counter 1 |
| 0x411 | 1041 | 41042 | MC1LL3_L | 32 bit (H..L) floating point |
| ... | ... | ... | ... | ... |
| 0x43E | 1086 | 41087 | MC1HH8_H | HH threshold for VAR8 of counter 1 |
| 0x43F | 1087 | 41088 | MC1HH8_L | 32 bit (H..L) floating point |
| 0x440 | 1088 | 41089 | MC2LL1_H | LL threshold for VAR1 of counter 2 |
| 0x441 | 1089 | 41090 | MC2LL1_L | 32 bit (H..L) floating point |
| ... | ... | ... | ... | ... |
| 0x47E | 1150 | 41151 | MC2HH8_H | HH threshold for VAR8 of counter 2 |
| 0x47F | 1151 | 41152 | MC2HH8_L | 32 bit (H..L) floating point |
| 0x480 | 1152 | 41153 | MC3LL1_H | LL threshold for VAR2 of counter 3 |
| 0x481 | 1153 | 41154 | MC3LL1_L | 32 bit (H..L) floating point |
| ... | ... | ... | ... | ... |
| 0x7FE | 2046 | 42047 | MC16HH8_H | HH threshold for VAR8 of counter 16 |
| 0x7FF | 2047 | 42048 | MC16HH8_L | 32 bit (H..L) floating point |

**Binary outputs space**
**Alarm bits informing on comparison results with current values of gas counters.**

| Start Address | | | Name | Description |
|---|---|---|---|---|
| hex | dec | MODBUS | | |
| 0x0B8 | 184 | 10185 | MC1V1LL | LL alarm bit for VAR1, counter 1 |
| 0x0B9 | 185 | 10186 | MC1V1L | L alarm bit for VAR1, counter 1 |
| 0x0BA | 186 | 10187 | MC1V1H | H alarm bit for VAR1, counter 1 |
| 0x0BB | 187 | 10188 | MC1V1HH | HH alarm bit for VAR1, counter 1 |
| 0x0BC | 188 | 10189 | MC1V2LL | LL alarm bit for VAR2, counter 1 |
| 0x0BD | 189 | 10190 | MC1V2L | L alarm bit for VAR2, counter 1 |
| 0x0BE | 190 | 10191 | MC1V2H | H alarm bit for VAR2, counter 1 |
| 0x0BF | 191 | 10192 | MC1V2HH | HH alarm bit for VAR2, counter 1 |
| 0x0C0 | 192 | 10193 | MC1V3LL | LL alarm bit for VAR3, counter 1 |
| … | … | … | … | … |
| 0x0D7 | 215 | 10216 | MC1V8HH | HH alarm bit for VAR8, counter 1 |
| 0x0D8 | 216 | 10217 | MC2V1LL | LL alarm bit for VAR1, counter 2 |
| … | … | … | … | … |
| 0x0F7 | 247 | 10248 | MC2V8HH | HH alarm bit for VAR8, counter 2 |
| 0x0F8 | 248 | 10249 | MC3V1LL | LL alarm bit for VAR1, counter 2 |
| … | … | … | … | … |
| 0x2B7 | 695 | 10696 | MC16V8HH | HH alarm bit for VAR8, counter 16 |

**Bits informing on state of communication with gas meter**

| Start Address | | | Name | Description |
|---|---|---|---|---|
| hex | dec | MODBUS | | |
| 0A8 | 168 | 10169 | SL1_ok | 1 – proper communication with counter 1 |
| 0A9 | 169 | 10170 | SL2_ok | 1 – proper communication with counter 2 |
| 0AA | 170 | 10171 | SL3_ok | 1 – proper communication with counter 3 |
| 0AB | 171 | 10172 | SL4_ok | 1 – proper communication with counter 4 |
| 0AC | 172 | 10173 | SL5_ok | 1 – proper communication with counter 5 |
| 0AD | 173 | 10174 | SL6_ok | 1 – proper communication with counter 6 |
| 0AE | 174 | 10175 | SL7_ok | 1 – proper communication with counter 7 |
| 0AF | 175 | 10176 | SL8_ok | 1 – proper communication with counter 8 |
| 0B0 | 176 | 10177 | SL9_ok | 1 – proper communication with counter 9 |
| 0B1 | 177 | 10178 | SL10_ok | 1 – proper communication with counter 10 |
| 0B2 | 178 | 10179 | SL11_ok | 1 – proper communication with counter 11 |
| 0B3 | 179 | 10180 | SL12_ok | 1 – proper communication with counter 12 |
| 0B4 | 180 | 10181 | SL13_ok | 1 – proper communication with counter 13 |
| 0B5 | 181 | 10182 | SL14_ok | 1 – proper communication with counter 14 |
| 0B6 | 182 | 10183 | SL15_ok | 1 – proper communication with counter 15 |
| 0B7 | 183 | 10184 | SL16_ok | 1 – proper communication with counter 16 |

### 13.10.6. Auxiliary resources for M-Bus mode

**Input registers space**

| Start Address | | | Description |
|---|---|---|---|
| hex | dec | MODBUS | |
| 086 | 134 | 30135 | data read from the gas counter |
| 100 | 256 | 30257 | data read from counter LEC 1 |

| 140 | 320 | 30321 | data read from counter LEC 2 |
|---|---|---|---|
| 180 | 384 | 30385 | data read from counter LEC 3 |
| 1C0 | 448 | 30449 | data read from counter LEC 4 |
| 200 | 512 | 30513 | data read from counter LEC 5 |
| 240 | 576 | 30577 | data read from counter LEC 6 |
| 280 | 640 | 30641 | data read from counter LEC 7 |
| 2C0 | 704 | 30705 | data read from counter LEC 8 |
| 300 | 768 | 30769 | data read from counter LEC 9 |
| 340 | 832 | 30833 | data read from counter LEC 10 |
| 380 | 896 | 30897 | data read from counter LEC 11 |
| 3C0 | 960 | 30961 | data read from counter LEC 12 |
| 400 | 1024 | 31025 | data read from counter LEC 13 |
| 440 | 1088 | 31089 | data read from counter LEC 14 |
| 480 | 1152 | 31153 | data read from counter LEC 15 |
| 4C0 | 1216 | 31217 | data read from counter LEC 16 |

**Structure of data read from gas counter.**

| Offset | Type Rx[High,Low] | Description |
|---|---|---|
| + 0 | word 16 bit | seconds (0..59) |
| + 1 | word 16 bit | minutes (0..59) (tb) |
| + 2 | word 16 bit | hours (0..23) |
| + 3 | word 16 bit | day (1..31) |
| + 4 | word 16 bit | month (1..12) |
| + 5 | word 16 bit | year (2000..2099) |
| + 6 | float 32 bit (H,L) | Vn0 Volume [$m^3$] |
| + 8 | float 32 bit (H,L) | Vn1 Volume [$m^3$] |
| + 10 | float 32 bit (H,L) | Qn Flow in normal conditions [$m^3/h$] |
| + 12 | float 32 bit (H,L) | Qr Flow in actual conditions [$m^3/h$] |
| + 14 | float 32 bit (H,L) | P Absolute pressure [kPa] |
| + 16 | float 32 bit (H,L) | T Gas temperature [$\circ$C] |
| + 18 | float 32 bit (H,L) | F Correction coefficient |
| + 20 | float 32 bit (H,L) | K1 Compression coefficient |
| Volume at the beginning of hour or after establishing the communication for calculation of hourly flow. | | |
| + 22 | word 16 bit | seconds (0..59) |
| + 23 | word 16 bit | minutes (0..59) (th) |
| + 24 | word 16 bit | hours (0..23) |
| + 25 | word 16 bit | day (1..31) |
| + 26 | word 16 bit | month (1..12) |
| + 27 | word 16 bit | year (2000..2099) |
| + 28 | float 32 bit (H,L) | Vh0 Volume [m3] |
| + 30 | float 32 bit (H,L) | Vh1 Volume [m3] |
| + 32 | word 16 bit | Vh Flow at beginning of hour [$m^3$]<br>Vn10 = Vn1 * 1e4 + Vn0<br>Vh10 = Vh1 * 1e4 + Vh0<br>Vh = (Vn10 - Vh10) |
| + 33 | word 16 bit | Qh hourly flow in current hour [$m^3/h$]<br>Qh = Vh + Qhp * (60 – tb + th) / 60 |
| +34 | word 16 bit | Qhp flow in previous hour [$m^3$] |

**Structure of input data read from heat meter**

| Offset | Type Rx[High,Low] | Description | |
|---|---|---|---|
| + 0 | 4 bytes R0[LL,L], R1[H,HH] | Ident. Nr | Header of received MBUS frame |
| + 2 | 2 bytes R2[L,H] | Manufacture | |
| + 3 | 2 bytes R3[Version,Medium] | Version + Medium | |
| + 4 | 2 bytes R4[Access,Status] | Access Nr + Status | |
| + 5 | 2 bytes R5[L,H] | Signature | |
| + 6 | dword 32 bit (H,L) | LEC - timestamp | LEC - timestamp |
| + 8 | dword 32 bit (H,L) | LEC - errorflag | LEC - error flags |
| + 10 | word 16 bit | seconds (0…59) | MT202 - timestamp |
| + 11 | word 16 bit | minutes (0…59) | |
| + 12 | word 16 bit | hours (0…23) | |
| + 13 | word 16 bit | day of week (1…7) | |
| + 14 | word 16 bit | day (1…31) | |
| + 15 | word 16 bit | month (1..12) | |
| + 16 | word 16 bit | year (2000..2099) | |
| + 17 | word 16 bit | high bits inform, which fields were read in current frame. | |
| + 18 | float 32 bit (H,L), 0001 | Temperature of supply flow [°C] | |
| + 20 | float 32 bit (H,L), 0002 | Temperature of return flow [°C] | |
| + 22 | float 32 bit (H,L), 0004 | Flow [m3/h] | |
| + 24 | float 32 bit (H,L), 0008 | Effect [W] | |
| + 26 | float 32 bit (H,L), 0010 | Volume [m3] | |
| + 28 | float 32 bit (H,L), 0020 | Energy [J] | |
| + 30 | float 32 bit (H,L), 0040 | Working time [h] | |
| + 32 | float 32 bit (H,L), 0080 | Auxiliary water meter 1 [m3] | |
| + 34 | float 32 bit (H,L), 0100 | Auxiliary water meter 2 [m3] | |
| + 36 | float 32 bit (H,L), 0200 | Auxiliary water meter 3 [m3] | |
| + 38 | float 32 bit (H,L), 0400 | Auxiliary water meter 4 [m3] | |
| + 40 | float 32 bit (H,L), 0800 | Max flow [m3/h] | |
| + 42 | float 32 bit (H,L), 1000 | Max effect [W] | |
| + 44 | float 32 bit (H,L), 2000 | Reserved for | |
| + 46 | float 32 bit (H,L), 4000 | Reserved for | |
| + 48 | float 32 bit (H,L), 8000 | Reserved for | |
| + 50 | dword 32 bit (H,L) | "Identification Number" from frame header in binary form, read from heat meter | |
| + 52 | dword 32 bit (H,L) | "Identification Number" assigned during MT module configuration | |
| +54 | word 16 bit | Temperature of supply flow[x10 °C] | |
| +55 | word 16 bit | Temperature of return flow[x10 °C] | |

**Holding registers space**
**Alarm thresholds placement (float type variables 32 bit HL, 2 Registers)**

| Start Address | | | Description |
|---|---|---|---|
| hex | dec | MODBUS | |
| 400 | 1024 | 41025 | LEC1, lower threshold - Supply temp. [°C] |
| 402 | 1026 | 41027 | LEC1, upper threshold - Supply temp. [°C] |
| 404 | 1028 | 41029 | LEC1, lower threshold - Return temp. [°C] |
| 406 | 1030 | 41031 | LEC1, upper threshold - Return temp. [°C] |
| 408 | 1032 | 41033 | LEC1, lower threshold – Flow [m3/h] |
| 40A | 1034 | 41035 | LEC1, upper threshold – Flow [m3/h] |
| 40C | 1036 | 41037 | LEC1, lower threshold - Effect [W] |

| Start Address | | | Description |
|---|---|---|---|
| hex | dec | MODBUS | |
| 40E | 1038 | 41039 | LEC1, upper threshold - Effect [W] |
| 410 | 1040 | 41041 | LEC1, lower threshold – Volume [m3] |
| 412 | 1042 | 41043 | LEC1, upper threshold – Volume [m3] |
| 414 | 1044 | 41045 | LEC1, lower threshold - Energy [J] |
| 416 | 1046 | 41047 | LEC1, upper threshold- Energy [J] |
| 418 | 1048 | 41049 | LEC1, lower threshold - Working time [h] |
| 41A | 1050 | 41051 | LEC1, upper threshold - Working time [h] |
| 41C | 1052 | 41053 | LEC1, Alarm hourly consumption for Aux. water meter 1 [m3/h] |
| 41E | 1054 | 41055 | LEC1, upper threshold - Auxiliary water meter 1 [m3] |
| 420 | 1056 | 41057 | LEC1, Alarm hourly consumption for Aux. water meter 2 [m3/h] |
| 422 | 1058 | 41059 | LEC1, upper threshold - Auxiliary water meter 2 [m3] |
| 424 | 1060 | 41061 | LEC1, Alarm hourly consumption for Aux. water meter 3 [m3/h] |
| 426 | 1062 | 41063 | LEC1, upper threshold - Auxiliary water meter 3 [m3] |
| 428 | 1064 | 41065 | LEC1, Alarm hourly consumption for Aux. water meter 4 [m3/h] |
| 42A | 1066 | 41067 | LEC1, upper threshold - Auxiliary water meter 4 [m3] |
| 42C | 1068 | 41069 | LEC1, lower threshold – Max flow [m3/h] |
| 42E | 1070 | 41071 | LEC1, upper threshold - Max flow [m3/h] |
| 430 | 1072 | 41073 | LEC1, lower threshold - Max effect [W] |
| 432 | 1074 | 41075 | LEC1, upper threshold - Max effect [W] |
| 434 | 1076 | 41077 | Reserved for |
| 436 | 1078 | 41079 | Reserved for |
| 438 | 1080 | 41081 | Reserved for |
| 43A | 1082 | 41083 | Reserved for |
| 43C | 1084 | 41085 | Reserved for |
| 43E | 1086 | 41087 | Reserved for |
| 440.. | 1088.. | 41089.. | LEC2 |
| 480.. | 1152.. | 41153.. | LEC3 |
| 4C0.. | 1216.. | 41217.. | LEC4 |
| 500.. | 1280.. | 41281.. | LEC5 |
| 540.. | 1344.. | 41345.. | LEC6 |
| 580.. | 1408.. | 41409.. | LEC7 |
| 5C0.. | 1472.. | 41473.. | LEC8 |
| 600.. | 1536.. | 41537.. | LEC9 |
| 640.. | 1600.. | 41601.. | LEC10 |
| 680.. | 1664.. | 41665.. | LEC11 |
| 6C0.. | 1728.. | 41729.. | LEC12 |
| 700.. | 1792.. | 41793.. | LEC13 |
| 740.. | 1856.. | 41857.. | LEC14 |
| 780.. | 1920.. | 41921.. | LEC15 |
| 7C0.. | 1984.. | 41985.. | LEC16 |

**Binary inputs space**
**Alarm bits placement**

| Start Address | | | Description |
|---|---|---|---|
| hex | dec | MODBUS | |
| bits informing on status of communication with heat counters | | | |
| 0A8 | 168 | 10169 | proper communication with counter 1 (SL1_ok) |
| 0A9 | 169 | 10170 | proper communication with counter 2 (SL2_ok) |
| 0AA | 170 | 10171 | proper communication with counter 3 (SL3_ok) |

| Start Address | | | Description |
|---|---|---|---|
| hex | dec | MODBUS | |
| 0AB | 171 | 10172 | proper communication with counter 4 (SL4_ok) |
| 0AC | 172 | 10173 | proper communication with counter 5 (SL5_ok) |
| 0AD | 173 | 10174 | proper communication with counter 6 (SL6_ok) |
| 0AE | 174 | 10175 | proper communication with counter 7 (SL7_ok) |
| 0AF | 175 | 10176 | proper communication with counter 8 (SL8_ok) |
| 0B0 | 176 | 10177 | proper communication with counter 9 (SL9_ok) |
| 0B1 | 177 | 10178 | proper communication with counter 10 (SL10_ok) |
| 0B2 | 178 | 10179 | proper communication with counter 11 (SL11_ok) |
| 0B3 | 179 | 10180 | proper communication with counter 12 (SL12_ok) |
| 0B4 | 180 | 10181 | proper communication with counter 13 (SL13_ok) |
| 0B5 | 181 | 10182 | proper communication with counter 14 (SL14_ok) |
| 0B6 | 182 | 10183 | proper communication with counter 15 (SL15_ok) |
| 0B7 | 183 | 10184 | proper communication with counter 16 (SL16_ok) |
| | | | alarm bits for heat counters |
| 0B8 | 184 | 10185 | LEC1, lower threshold - Supply temp. [°C] |
| 0B9 | 185 | 10186 | LEC1, upper threshold - Supply temp. [°C] |
| 0BA | 186 | 10187 | LEC1, lower threshold- Return temp. [°C] |
| 0BB | 187 | 10188 | LEC1, upper threshold - Return temp. [°C] |
| 0BC | 188 | 10189 | LEC1, lower threshold - Flow [m3/h] |
| 0BD | 189 | 10190 | LEC1, upper threshold - Flow [m3/h] |
| 0BE | 190 | 10191 | LEC1, lower threshold - Effect [W] |
| 0BF | 191 | 10192 | LEC1, upper threshold - Effect [W] |
| 0C0 | 192 | 10193 | LEC1, lower threshold - Volume [m3] |
| 0C1 | 193 | 10194 | LEC1, upper threshold - Volume [m3] |
| 0C2 | 194 | 10195 | LEC1, lower threshold - Energy [J] |
| 0C3 | 195 | 10196 | LEC1, upper threshold - Energy [J] |
| 0C4 | 196 | 10197 | LEC1, lower threshold - Working time [h] |
| 0C5 | 197 | 10198 | LEC1, upper threshold - Working time [h] |
| 0C6 | 198 | 10199 | Reserved for |
| 0C7 | 199 | 10200 | LEC1, upper threshold - Auxiliary water meter 1 [m3] |
| 0C8 | 200 | 10201 | Reserved for |
| 0C9 | 201 | 10202 | LEC1, upper threshold - Auxiliary water meter 2 [m3] |
| 0CA | 202 | 10203 | Reserved for |
| 0CB | 203 | 10204 | LEC1, upper threshold - Auxiliary water meter 3 [m3] |
| 0CC | 204 | 10205 | Reserved for |
| 0CD | 205 | 10206 | LEC1, upper threshold - Auxiliary water meter 4 [m3] |
| 0CE | 206 | 10207 | LEC1, lower threshold - Max. flow [m3/h] |
| 0CF | 207 | 10208 | LEC1, upper threshold - Max. flow [m3/h] |
| 0D0 | 208 | 10209 | LEC1, lower threshold - Max. effect [W] |
| 0D1 | 209 | 10210 | LEC1, upper threshold - Max. effect [W] |
| 0D2 | 210 | 10211 | Reserved for |
| 0D3 | 211 | 10212 | Reserved for |
| 0D4 | 212 | 10213 | Reserved for |
| 0D5 | 213 | 10214 | Reserved for |
| 0D6 | 214 | 10215 | Reserved for |
| 0D7 | 215 | 10216 | Reserved for |
| 0D8.. | 216.. | 10217.. | LEC2 |
| 0F8.. | 248.. | 10249.. | LEC3 |
| 118.. | 280.. | 10281.. | LEC4 |
| 138.. | 312.. | 10313.. | LEC5 |
| 158.. | 344.. | 10345.. | LEC6 |
| 178.. | 376.. | 10377.. | LEC7 |
| 198.. | 408.. | 10409.. | LEC8 |
| 1A8.. | 440.. | 10441.. | LEC9 |

| Start Address | | | Description |
|---|---|---|---|
| hex | dec | MODBUS | |
| 1C8.. | 472.. | 10473.. | LEC10 |
| 1E8.. | 504.. | 10505.. | LEC11 |
| 218.. | 536.. | 10537.. | LEC12 |
| 238.. | 568.. | 10569.. | LEC13 |
| 258.. | 600.. | 10601.. | LEC14 |
| 278.. | 632.. | 10633.. | LEC15 |
| 298.. | 664.. | 10665.. | LEC16 |
| alarm and information bits for GAZMODEM (configuration port) | | | |
| 3B8 | 952 | 10953 | proper communication with gas counter |

## 13.10.7. Auxiliary resources of NMEA 0183 mode

**Binary outputs space
information bits**

| Address of bit | | | Register address | Description | Validity register Address.bit |
|---|---|---|---|---|---|
| Hex | dec | MODBUS | | | |
| 0x80 | 128 | 129 | 0x402 | Latitude | 0x401.0 |
| 0x81 | 129 | 130 | 0x404 | Longitude | 0x401.1 |
| 0x82 | 130 | 131 | 0x406 | Altitude over sea level | 0x401.2 |
| 0x83 | 131 | 132 | 0x408 | Speed Over Ground | 0x401.3 |
| 0x84 | 132 | 133 | 0x40A | Course Over Ground | 0x401.4 |
| 0x85 | 133 | 134 | 0x40C | Number of used satellites | 0x401.5 |
| 0x86 | 134 | 135 | 0x40E | Atmospheric pressure | 0x401.6 |
| 0x87 | 135 | 136 | 0x410 | Air temperature | 0x401.7 |
| 0x88 | 136 | 137 | 0x412 | Relative humidity | 0x401.8 |
| 0x89 | 137 | 138 | 0x414 | Dew point | 0x401.9 |
| 0x8A | 138 | 139 | 0x416 | Wind direction | 0x401.10 |
| 0x8B | 139 | 140 | 0x418 | Wind speed | 0x401.11 |
| 0x8C | 140 | 141 | 0x41A | Relative wind direction | 0x401.12 |
| 0x8D | 141 | 142 | 0x41C | Relative wind speed | 0x401.13 |
| 0x8E | 142 | 143 | 0x41E | Theoretical wind direction | 0x401.14 |
| 0x8F | 143 | 144 | 0x420 | Theoretical wind speed | 0x401.15 |
| 0x90 | 144 | 145 | 0x422 | Relative wind chill factor | 0x400.0 |
| 0x91 | 145 | 146 | 0x424 | Theoretical wind chill factor | 0x400.1 |

Reading values from NMEA frame and recording it to Register is marked by setting of corresponding bit. Users wanting to detect new recording have to reset respective bit and await consecutive setting it to 1 marking a new recording into corresponding register.

**Holding registers space**
Mentioned Registers hold information read from NMEA frames.
All values are stored in 2 16 bit registers in form of 32 bit signed integer, scaled appropriately for storing decimal values. Registers are ordered in HL order, which is High 16 bits are placed in first register (with lower address).

Example 1:
The value 1234 in register representing numbers with 2 decimals FIX(2) will be recorded as integer value 123400.
After conversion to 16 bit form = 0x1E208
RegH = 0x0001 (1)
RegL = 0xE208 (57862)

Example 2:
Value -10,3 FIX(1)
Integer value in 32 bit register = -103
After conversion to 16 bit form = 0xFFFFFF99
RegH = 0xFFFF (65535)
RegL = 0xFF99 (65433)


Description registers
Addresses in tables point to placement of first register holding High order word of the number. Low order word lays in next register.

| Address | 0x400 (1024) | **Register of validity of NMEA variables** |
|---------|-------------|-----|
|  | Modbus (41025) |  |

This register is used to control whether values read from NMEA frames held in registers are valid (up to date).
A High bit signals that data in corresponding register is valid, while Low bit signals that register holds outdated data. This bit is set High each time the value is written to corresponding register and reset when data is older than defined during module configuration time of signaling variables validity.
Bit assignment is described in information bits table

| Address | 0x402 (1026) | **Latitude** |
|---------|-------------|-----|
|  | Modbus (41027) |  |
| Format 1 | FIX(5) | degrees (S), minutes (M), fractions of minute (m) [SSMM.mmmmm] |
| Format 2 | FIX(7) | degrees (S), fractions of degrees (s) [SS.sssssss] |
| Frames | $GPRMC, $GPGGA, $GPGLL, |  |

Latitude North is represented by positive number while South is represented by negative number.

| Address | 0x404 (1028) | **Longitude** |
|---------|-------------|-----|
|  | Modbus (41029) |  |
| Format 1 | FIX(5) | degrees (S), minutes (M), fractions of minute (m) [SSSMM.mmmmm] |
| Format 2 | FIX(7) | degrees (S), fractions of degrees (s) [SSS.sssssss] |
| Frames | $GPRMC, $GPGGA, $GPGLL |  |

Longitude East is represented by positive number while West is represented by negative number.

| Address | 0x406 (1030) | **Altitude above sea level** |
|---------|-------------|-----|
|  | Modbus (41031) |  |
| Format | FIX(1) | Unit meter [m] |
| Frames | $GPGGA |  |
| Address | 0x408 (1032) | **Speed Over Ground** |
|  | Modbus (41033) |  |
| Format | FIX(1) | Unit kilometers/hour [km/h] |
| Frames | $GPRMC, $GPVTG |  |

| Address | 0x40A (1034) | Course Over Ground | |
|---|---|---|---|
| | Modbus (41035) | | |
| Format | FIX(1) | Unit degrees[° True] | |
| Frames | $GPRMC, $GPVTG | | |
| Address | 0x40C (1036) | Number of satellites in use | |
| | Modbus (41037) | | |
| Format | FIX(0) | | |
| Frames | $GPGGA | | |
| Address | 0x40E (1038) | Barometric pressure | |
| | Modbus (41039) | | |
| Format | FIX(0) | Unit [hPa] | |
| Frames | $WIMDA | | |
| Address | 0x410 (1040) | Air temperature | |
| | Modbus (41041) | | |
| Format | FIX(1) | Unit degrees Celsius [°C] | |
| Frames | $WIMDA | | |
| Address | 0x412 (1042) | Relative humidity | |
| | Modbus (41043) | | |
| Format | FIX(1) | Unit [%] | |
| Frames | $WIMDA | | |
| Address | 0x414 (1044) | Dew point | |
| | Modbus (41045) | | |
| Format | FIX(1) | Unit degrees Celsius [°C] | |
| Frames | $WIMDA | | |
| Address | 0x416 (1046) | Wind direction | |
| | Modbus (41047) | | |
| Format | FIX(1) | Unit degrees[°C] | |
| Frames | $WIMDA, $WIMWD | | |
| Address | 0x418 (1048) | Wind speed | |
| | Modbus (41049) | | |
| Format | FIX(1) | Unit kilometers/hour [km/h] | |
| Frames | $WIMDA, $WIMWD | | |
| Address | 0x41A (1050) | Relative wind direction | |
| | Modbus (41051) | | |
| Format | FIX(1) | Unit degrees [°] | |
| Frames | $WIMWV (relative) | | |

| | | |
|---|---|---|
| Address | 0x41C (1052) | **Relative wind speed** |
| | Modbus (41053) | |
| Format | FIX(1) | Unit kilometers/hour[km/h] |
| Frames | $WIMWV (relative) | |
| Address | 0x41E (1054) | **Theoretical wind direction** |
| | Modbus (41055) | |
| Format | FIX(1) | Unit degrees [°] |
| Frames | $WIMWV (theoretical) | |
| Address | 0x420 (1056) | **Theoretical wind speed** |
| | Modbus (41057) | |
| Format | FIX(1) | Unit kilometers/hour [km/h] |
| Frames | $WIMWV (theoretical) | |
| Address | 0x422 (1058) | **Relative wind chill temperature** |
| | Modbus (41059) | |
| Format | FIX(1) | Unit degrees [°] |
| Frames | $WIXDR | |
| Address | 0x424 (1060) | **Theoretical wind chill temperature** |
| | Modbus (41061) | |
| Format | FIX(1) | Unit degrees [°] |
| Frames | $WIXDR | |