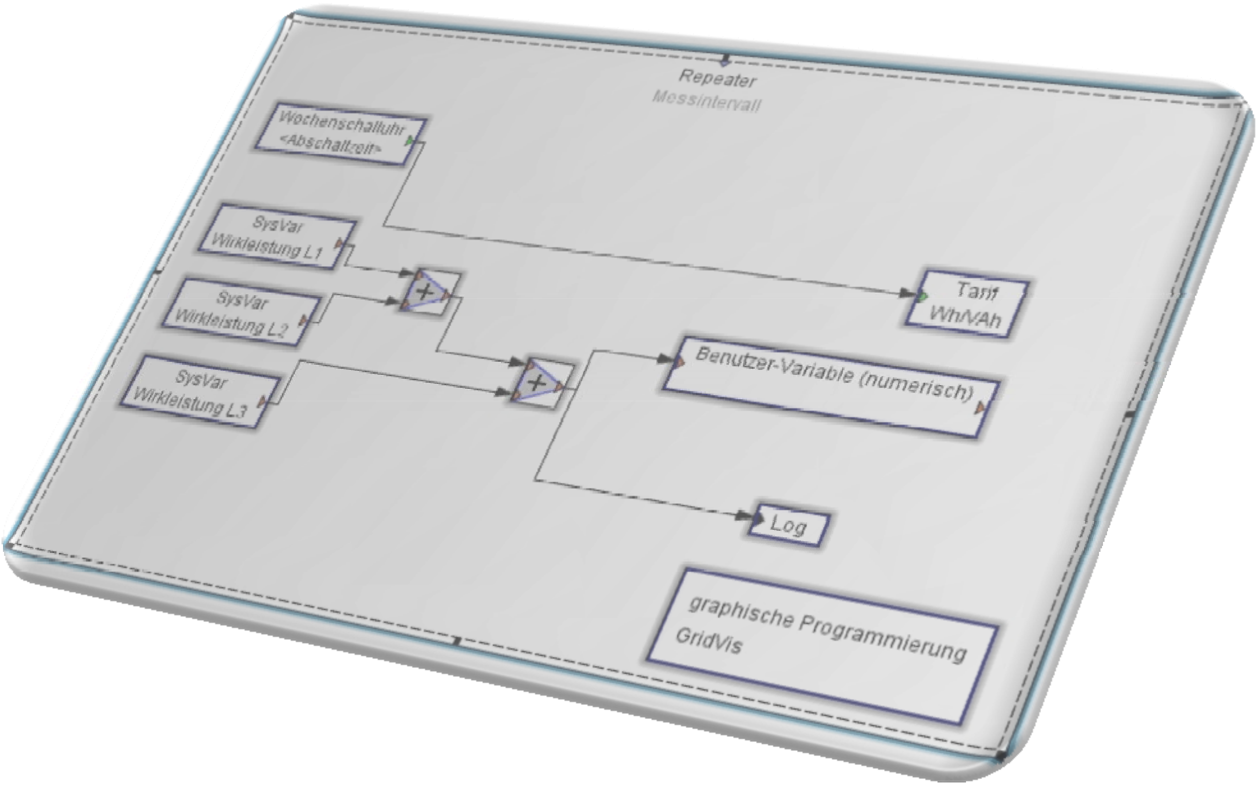
Graphic Programming GridVis

Operation Manual



Janitza electronics GmbH
Vor dem Polstück 1
D-35633 Lahnau
Support Tel. (0 64 41) 9642-22
Fax (0 64 41) 9642-30
E-mail: info@janitza.de

Contents

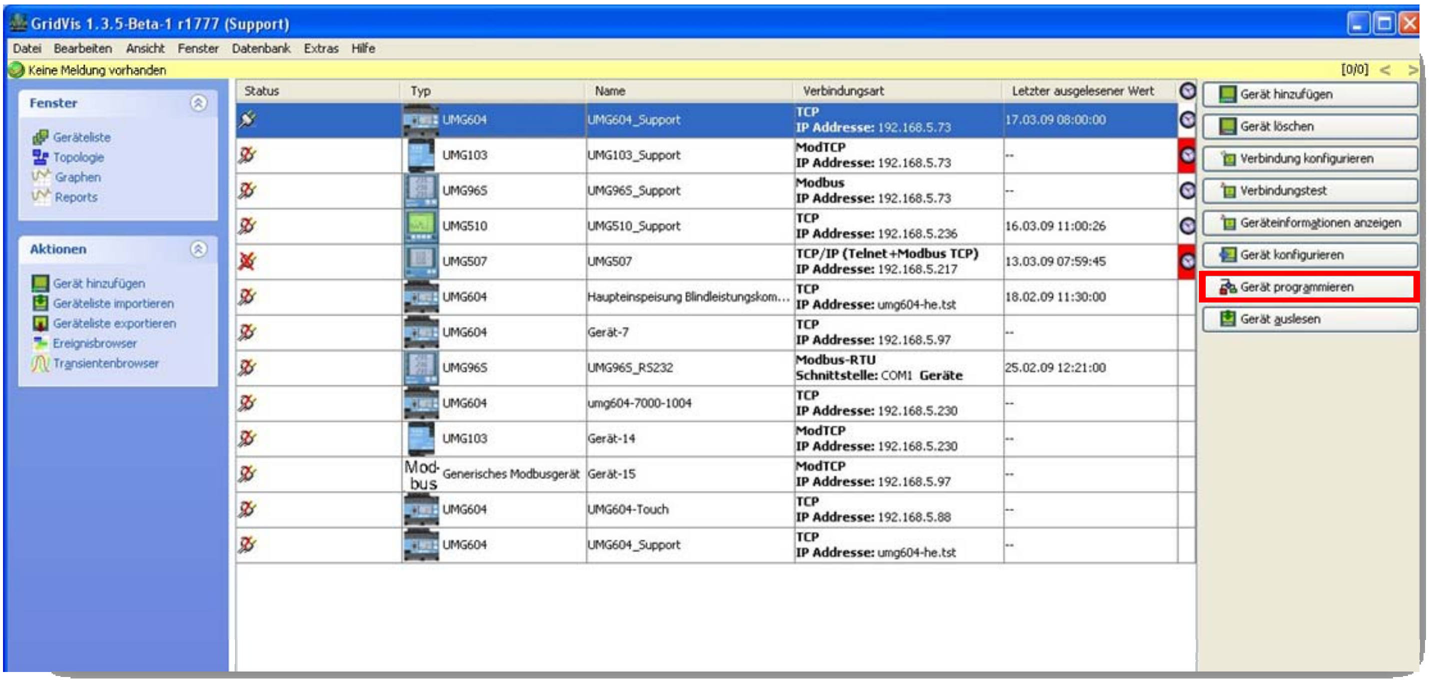
Introduction

Graphic programming serves the purpose of creating logical links or mathematical functions and configuring them. Your own digital outputs can be recorded and digital inputs can be analysed. Furthermore, registers from external units can be analysed or recorded using the modbus. For example, with the “graphic programming” tool, you can configure and analyse the UMG 604 according to your own special applications. Even violations of limit values, time switch functions or the recording of special values are freely configurable with graphic programming. Several example applications are presented in the following chapters and explanations are provided for the handling of the graphic programme editor. The actual programming language “Jasic” is created in the background and can always be accessed by changing the tab; more information on this is provided later. The personally created programmes can be recorded as a file on the computer or can be directly transferred to the measuring instrument. 7 memory spaces, each with 128 KB of memory capacity, are available for the storage of your programmes. All examples in this operating manual are included in the “examples” folder and can be inputted or changed using the editor.

More information on the UMG series measuring instruments and the GridVis software is available at www.janitza.de

You enter the graphic programme editor as follows:

- Open up GridVis
- Implement a UMG 604 (for example)
- Select the instrument (highlighted in blue)
- Choose “program device“ from the right-hand selection list (see illustration)
- The programme editor opens



Status	Typ	Name	Verbindungsart	Letzter ausgelesener Wert
	UMG604	UMG604_Support	**TCP** **IP Addresse:** 192.168.5.73	17.03.09 08:00:00
	UMG103	UMG103_Support	**ModTCP** **IP Addresse:** 192.168.5.73	--
	UMG96S	UMG96S_Support	**Modbus** **IP Addresse:** 192.168.5.73	--
	UMG510	UMG510_Support	**TCP** **IP Addresse:** 192.168.5.236	16.03.09 11:00:26
	UMG507	UMG507	**TCP/IP (Telnet+Modbus TCP)** **IP Addresse:** 192.168.5.217	13.03.09 07:59:45
	UMG604	Haupteinspeisung Blindleistungskom...	**TCP** **IP Addresse:** umg604-he.tst	18.02.09 11:30:00
	UMG604	Gerät-7	**TCP** **IP Addresse:** 192.168.5.97	--
	UMG96S	UMG96S_RS232	**Modbus-RTU** **Schnittstelle:** COM1 **Geräte**	25.02.09 12:21:00
	UMG604	umg604-7000-1004	**TCP** **IP Addresse:** 192.168.5.230	--
	UMG103	Gerät-14	**ModTCP** **IP Addresse:** 192.168.5.230	--
	Mod-bus Generisches Modbusgerät	Gerät-15	**ModTCP** **IP Addresse:** 192.168.5.97	--
	UMG604	UMG604-Touch	**TCP** **IP Addresse:** 192.168.5.88	--
	UMG604	UMG604_Support	**TCP** **IP Addresse:** umg604-he.tst	--

Illustration 1

The programme editor

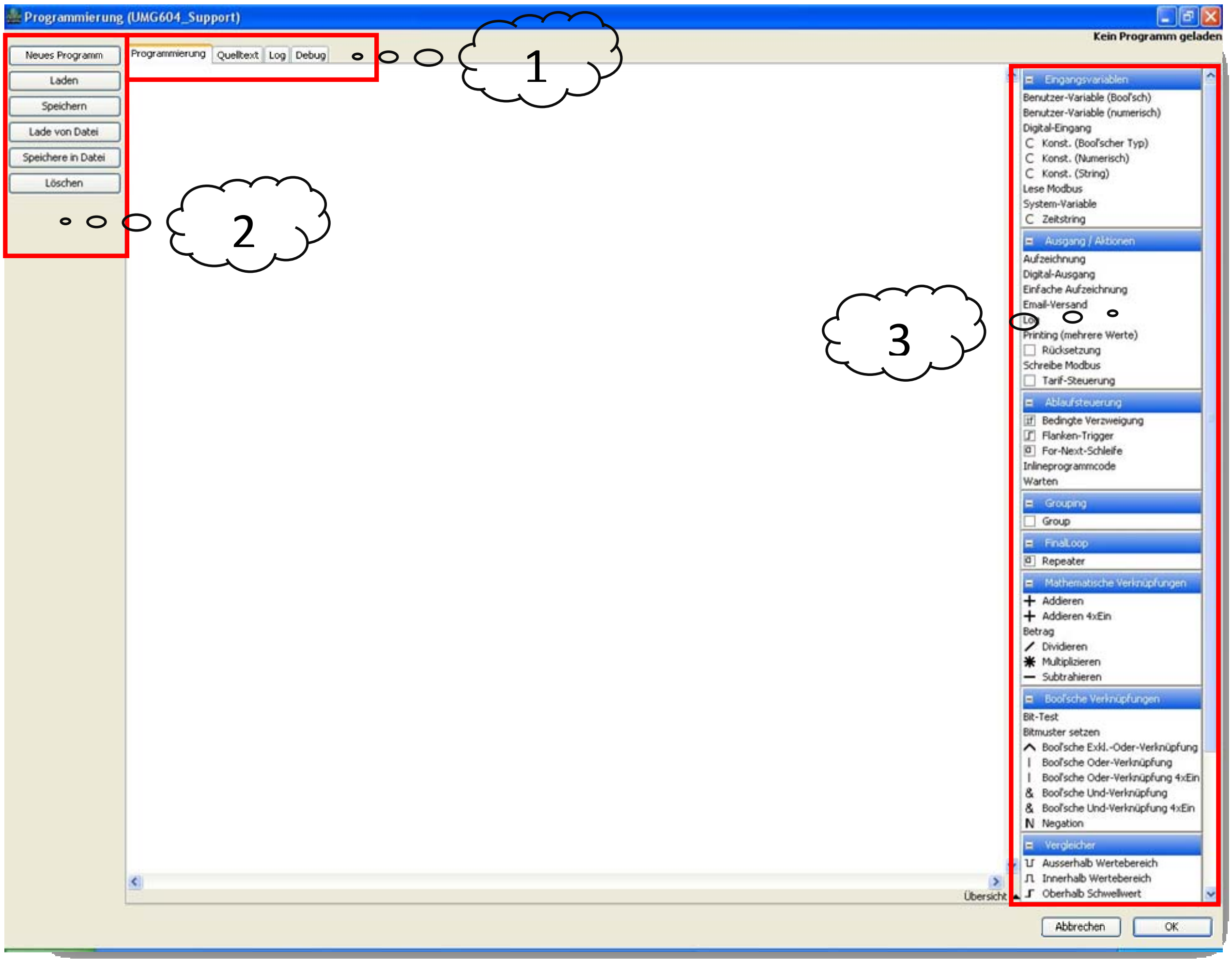


Illustration 2

1. Tab selection: change between the source text, graphic programming and the debug or log functions.
2. Menu bar: store and input the programme code in the instrument or as a jas file on the computer.
3. Toolbar: selection of functions; selected functions are pulled into the programme field with drag and drop.

⚠ Applications which are not transferred to the instrument or are not saved in a file are lost!

Variables

The programme editor offers various types of variables which are described in the following section. All variables can be dragged into the programme field using drag and drop. In order to open the configuration window for the individual variables, you must double-click the left button of the mouse in the variable box.

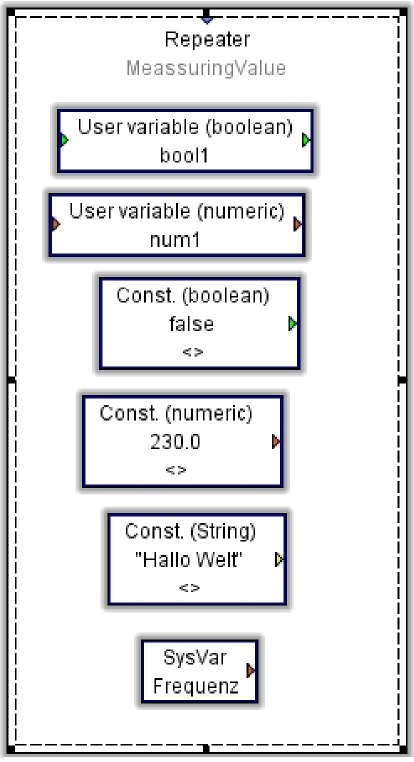


Illustration 3

User variables: user variables are personally set up variables which can be maintained locally or globally. A local variable can only be called-up within the respective programme and can only be used locally. If a variable is declared as global, it can be called up in a different programme and can be read out or recorded using the modbus register. The auto save function allows storage of the value if a power failure occurs (persistence). User variables can be recorded in Boolean form (digital 1/0) or in numerical form.

Constants: constants are fixed values which can be used to work in the programme code. The fixed value can be entered with a double left click of the mouse on the variable box. There is also a difference between Boolean and numerical forms here. The exception is the string; a text can be lodged with this variable.

System variables: system variables are measurement values which are provided by the instrument. All measurement data and calculated values such as “consumption“ can be called up with this variable and can be processed in the programme code.

Not all variables can be connected to each other; this also applies to all other functions which can be selected. The outputs are marked in various colours.

Green = Boolean / red = numeric / yellow = string / black = all types / blue = request

The repeater

Illustration 4

The repeater is the basis for the programme. Programming is started with this module. Just like all other modules, this module is dragged into the programming field with drag and drop. The repeater is used for permanent processing of the programming code. All overlying modules are processed at the configured time lapses. The processing time is set by double-clicking the left button of the mouse on the module. Selection options are: no delay, measurement intervals, second, minute, hour and day.

The function code is not processed without a repeater. The repeater should be positioned first.

Analysis of the digital input

The following illustration shows the analysis of two digital inputs. The result is directed with a digital output "logically AND requested". The example application is saved in the examples file and can be introduced using the programme editor in the menu bar → load document. You can configure the digital input with a double click of the left mouse button. The modules are made available in the tool bar and can be freely selected and coordinated with drag and drop. The links can be created from pin to pin by holding down the left button of the mouse.

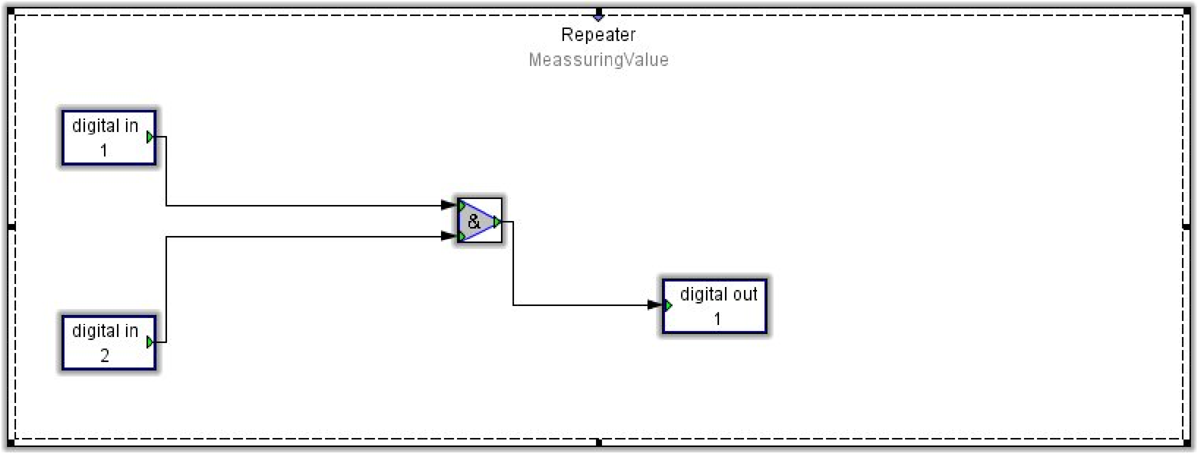


Illustration 5

Digital output readout

The following example shows the switching of a digital output after a certain incident. In this case the incident is the monitoring of the L1 voltage. The threshold value is 238V and is specified with a constant.

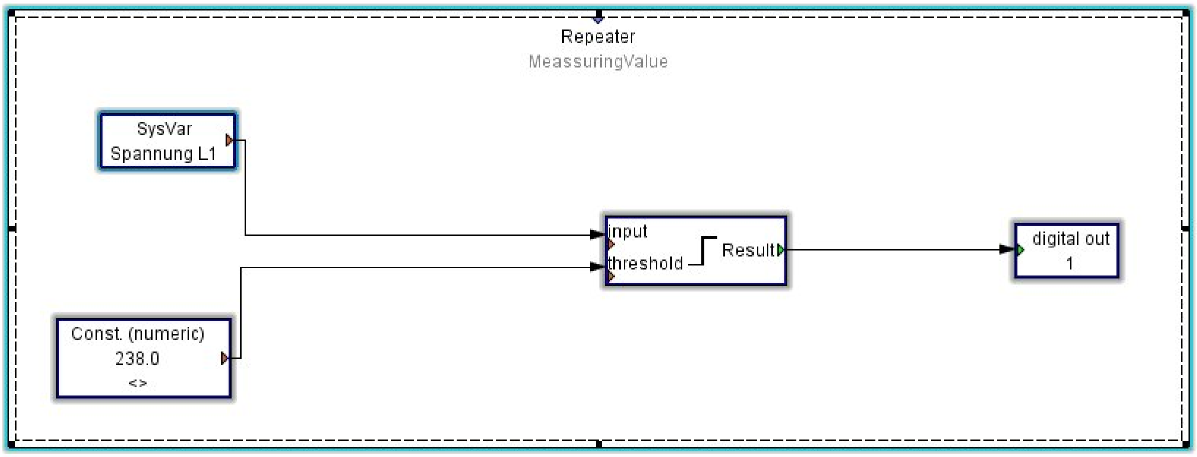


Illustration 6

Boolean links

The following illustration shows the Boolean links which are available in the programme editor. All links can be interconnected in all variations.

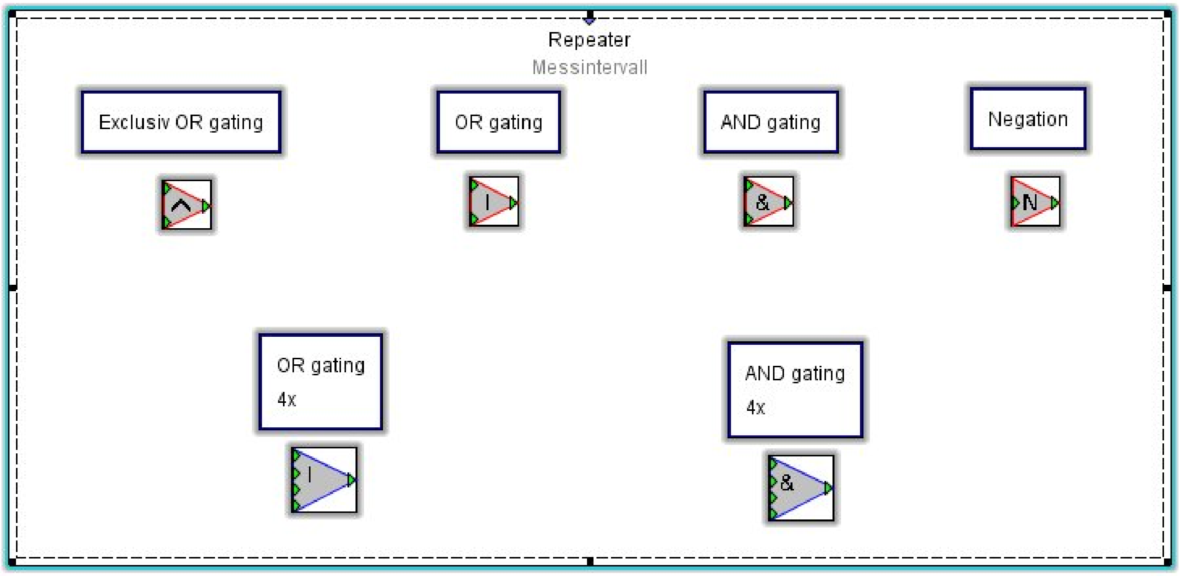


Illustration 7

In illustration 8, we can find an example with AND and OR gates.

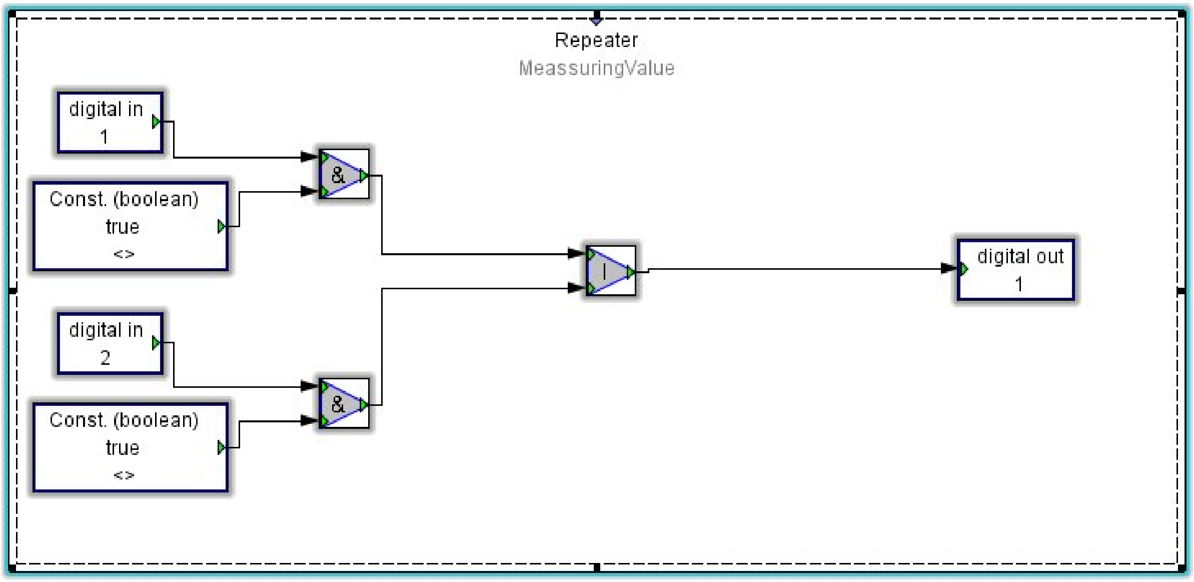


Illustration 8

Mathematics

You can carry out arithmetical computing operations using mathematical connections.
Illustration 9 shows the addition of the individual effective consumptions and the sum is recorded in a user variable. The second example shows the read out of the L1 voltage system variable; the nominal value 230V is subtracted from the actual value and the difference is directed to a user variable.

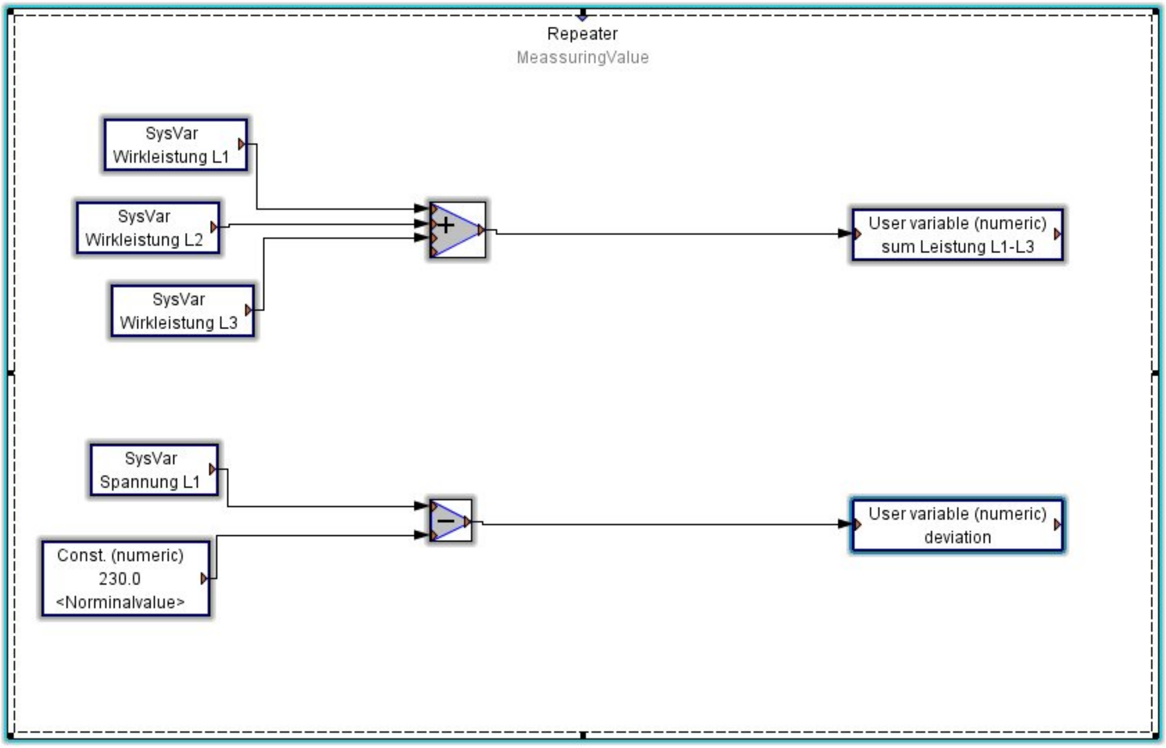
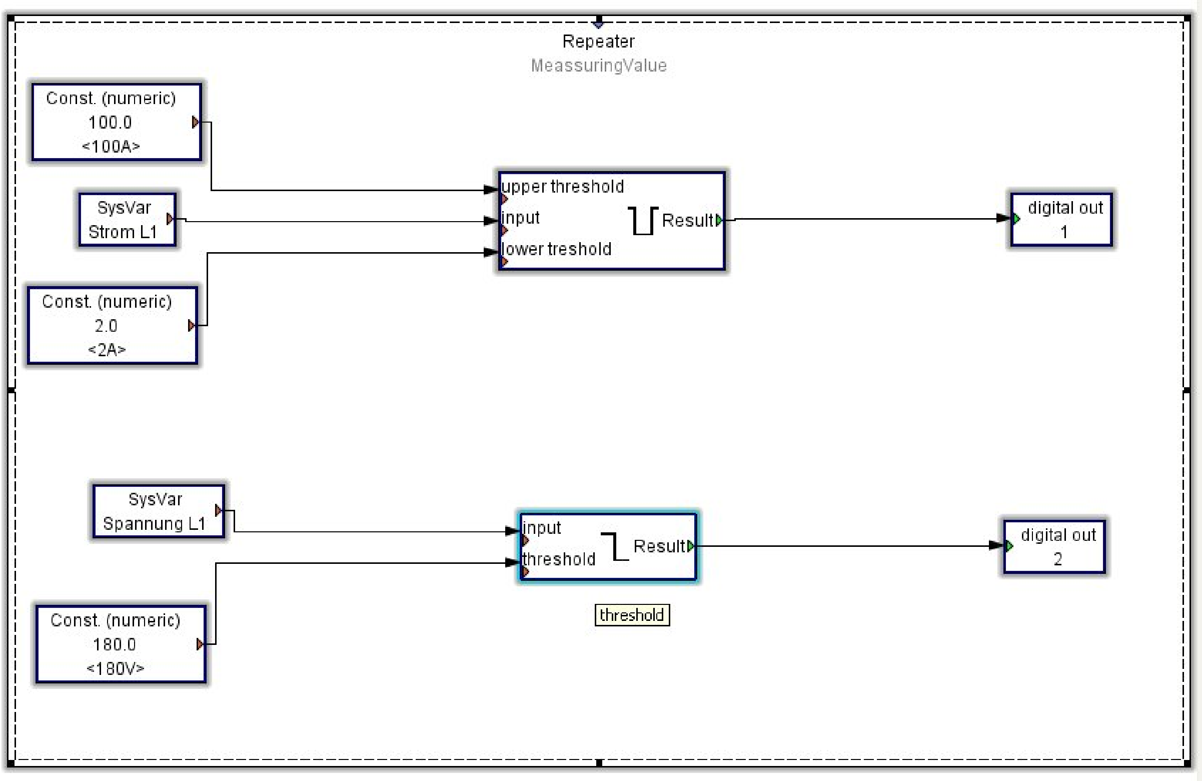


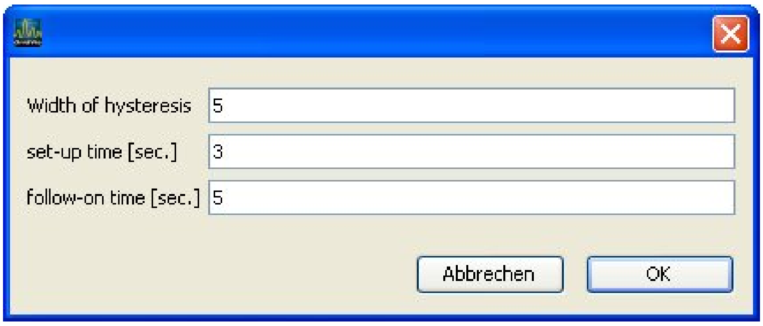
Illustration 9

Limit value monitoring (comparer)

In the limit value monitoring example, we can see two variations for monitoring a value. The first example shows the monitoring of the L1 current whereby constants are used to establish the threshold values. If the predefined value is exceeded, a digital "1" signal is given to digital output 1. The second example works like the first one but this one only has a lower limit; 180V in this case.



The limit value function modules can be configured as follows:


Width of hysteresis 5

set-up time [sec.] 3

follow-on time [sec.] 5

Illustration 10

The setting of the hysteresis can be significant if you often have fluctuating values around the threshold value.

Call requests (group)

You can manage the processing of the function code with the help of the group function. The functions in a group are only processed upon request. This feature can be beneficial in various cases. The following two examples (illustrations 11 and 12) show different applications for a group call request.

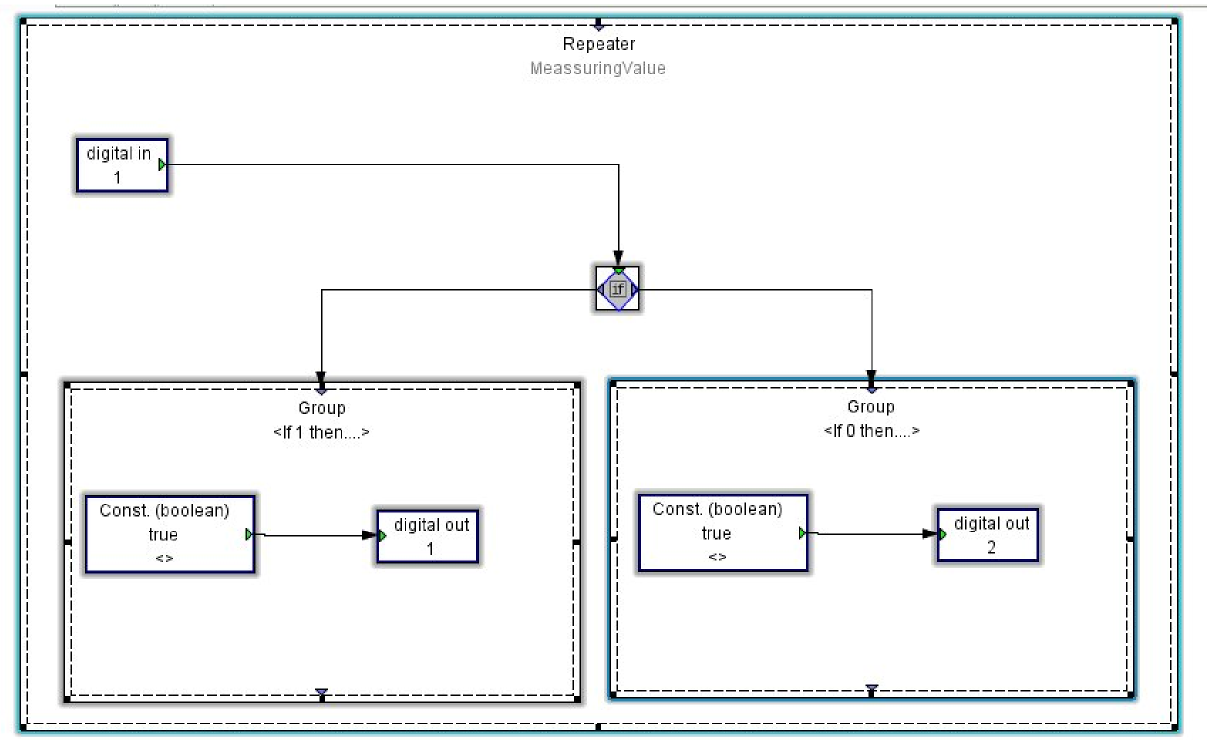


Illustration 11

In illustration 11, the processing of the groups is dependent upon an IF instruction. The status of the digital input now decides which group should be processed. In this case, both groups could never be processed together.

Several function connections can be conveniently copied with the help of the group

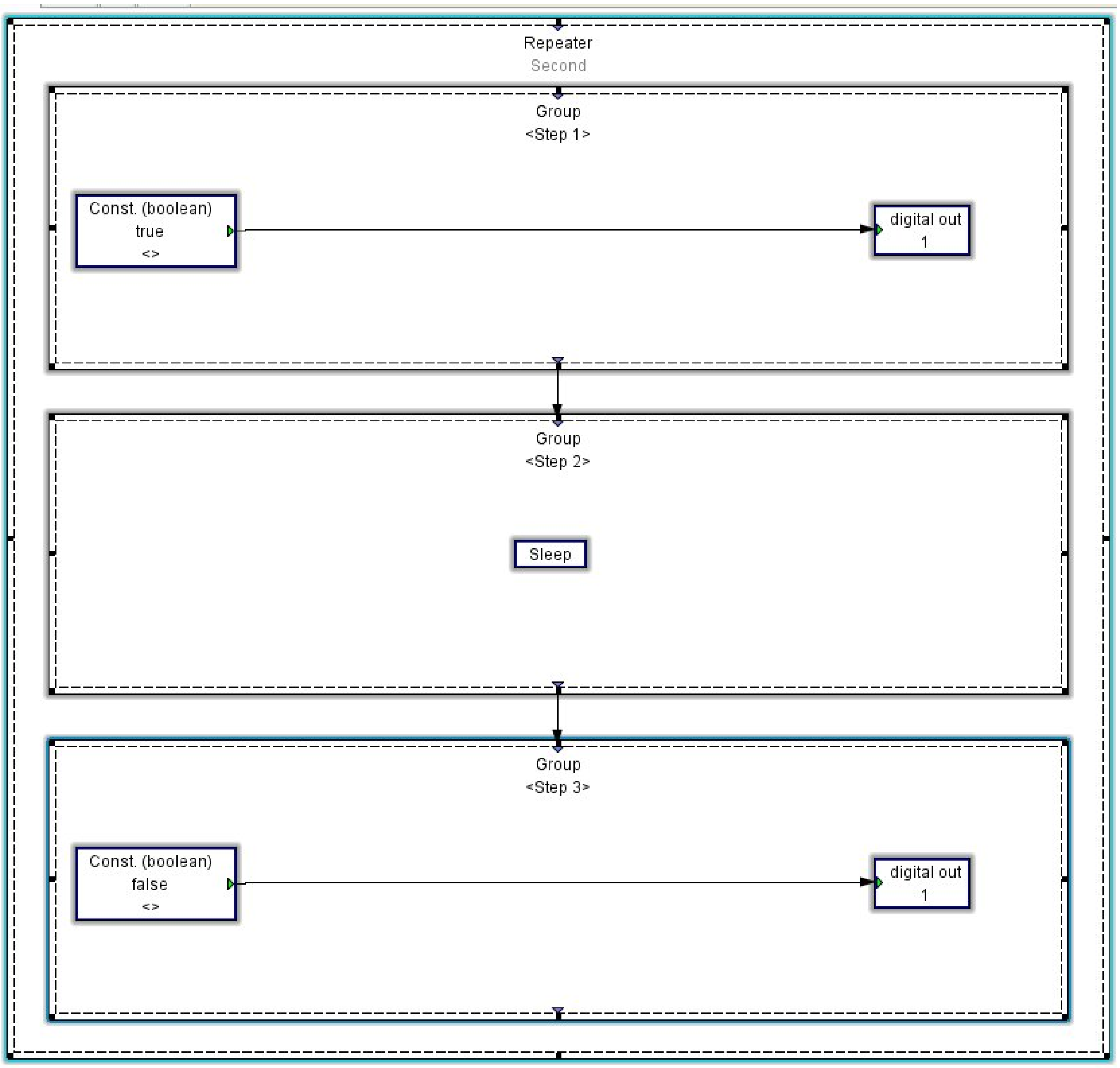


Illustration 12

The process chain option is shown in illustration 12. The function code is processed step by step (in sequence).

Resetting energy

There are several options for resetting the energy. In the following illustration, the energy is reset every day using the time switch function module.

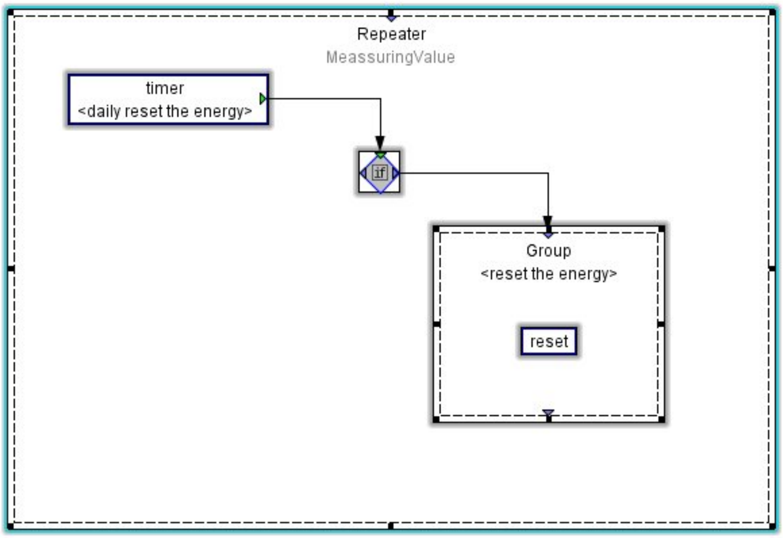
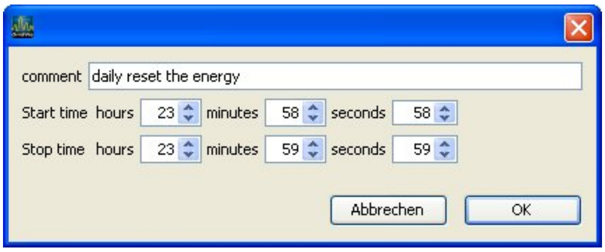


Illustration 13

The configuration of the time switch is displayed in illustration 14. A one second pulse resets the energy at 23:59 p.m.



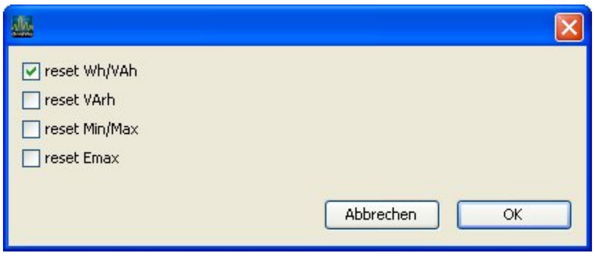
comment: daily reset the energy

Start time hours 23 minutes 58 seconds 58

Stop time hours 23 minutes 59 seconds 59

Illustration 14

Illustration 15 shows the configuration of the reset function module.



- ☑ reset Wh/VAh
- ☐ reset VArh
- ☐ reset Min/Max
- ☐ reset Emax

Illustration 15

Despatch e-mail

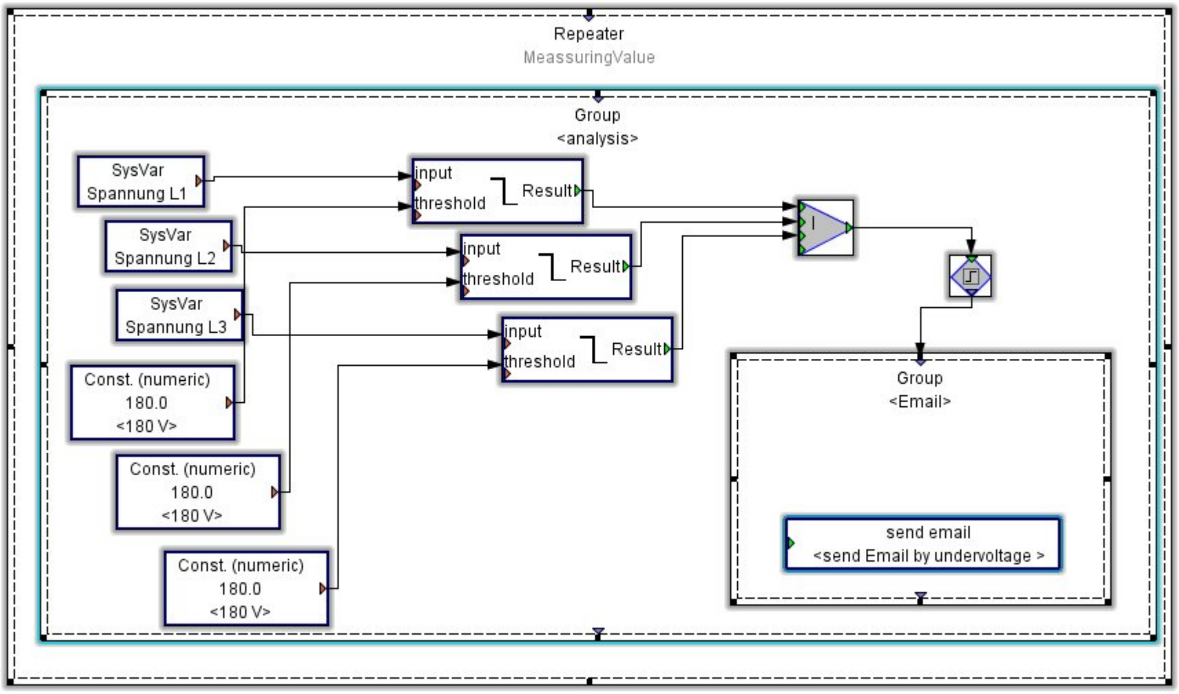
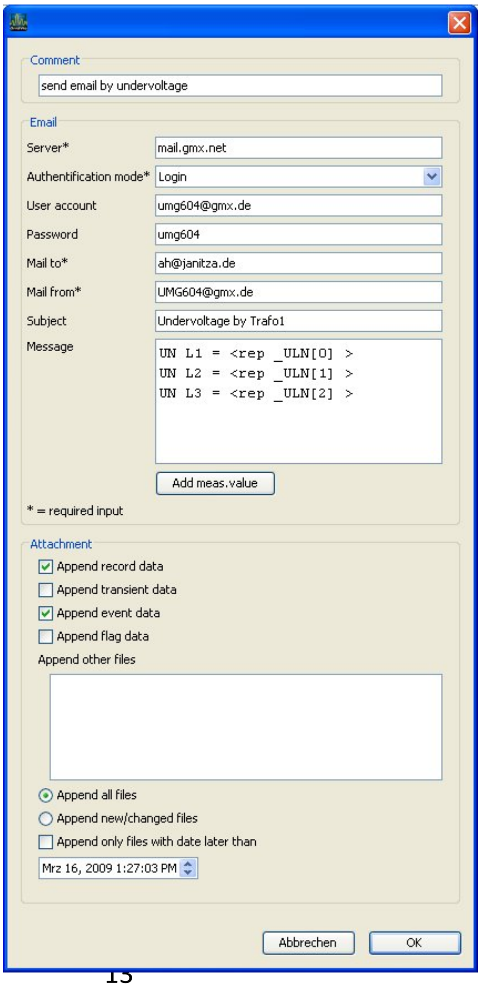


Illustration 16



Comment

send email by undervoltage

Email

Server* mail.gmx.net

Authentification mode* Login

User account umg604@gmx.de

Password umg604

Mail to* ah@janitza.de

Mail from* UMG604@gmx.de

Subject Undervoltage by Trafo1

Message

```
UN L1 = <rep _ULN[0] >
UN L2 = <rep _ULN[1] >
UN L3 = <rep _ULN[2] >
```

* = required input

Attachment

☑ Append record data

☐ Append transient data

☑ Append event data

☐ Append flag data

Append other files

● Append all files

○ Append new/changed files

☐ Append only files with date later than

Mrz 16, 2009 1:27:03 PM

Illustration 17

In this example, an e-mail is despatched when the limit value falls below 180V. All three phases are monitored.

Illustration 17 shows the configuration of the “despatch e-mail” function module. Your e-mail provider can give you information on your outbox server (server) and the authentication mode. The e-mail address is entered in the user cell and in “mail from“. The e-mail title is issued in the subject cell. The message cell is for the contents of the e-mail; text and measurement values can be entered here. You also have the option of sending recordings and incidents as an attachment.

The outbox server of your provider must be entered in the server cell.

Reading and recording modbus slave units

With the modbus “read” module, you have the opportunity of reading out the register addresses from slave units. These values can then be recorded in user variables and can be processed further.

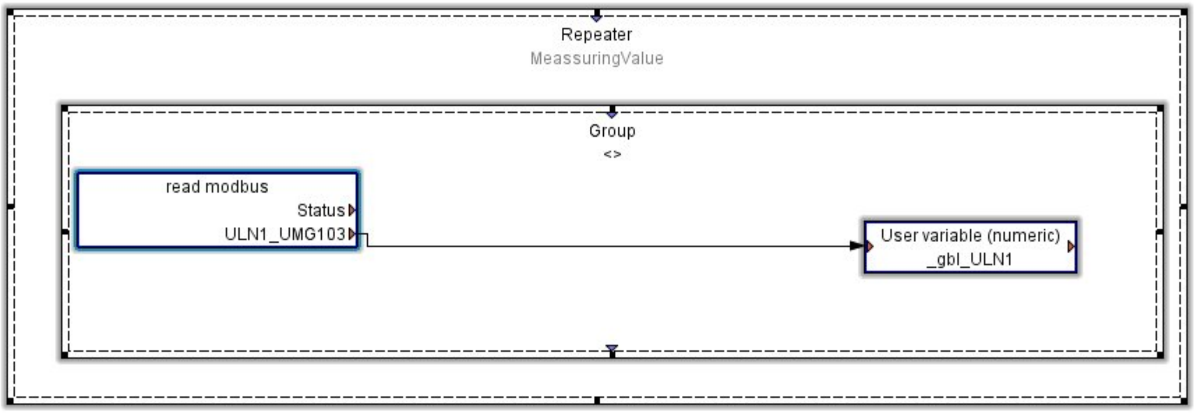
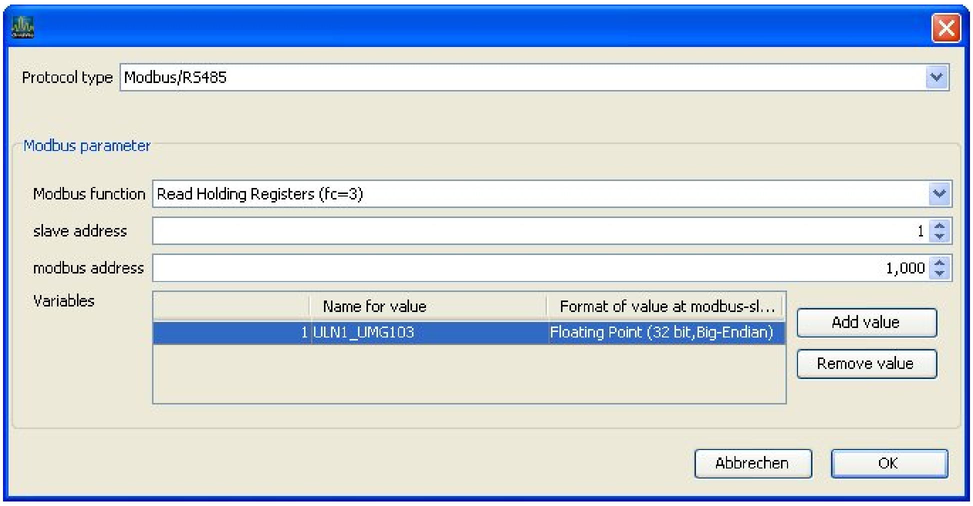


Illustration 18

Illustration 19 presents the configuration of the “read modbus” function module. The unit‘s address and the modbus register address is entered here (start register). In the bottom section, you have the opportunity of lodging the variables and issuing names. A new pin is led outwards for each variable (value) and this value can then be connected further. Attention must be paid that the right register format is configured. The format also determines the size of the register for several values.



Protocol type: Modbus/RS485

Modbus parameter

Modbus function: Read Holding Registers (fc=3)

slave address: 1

modbus address: 1,000

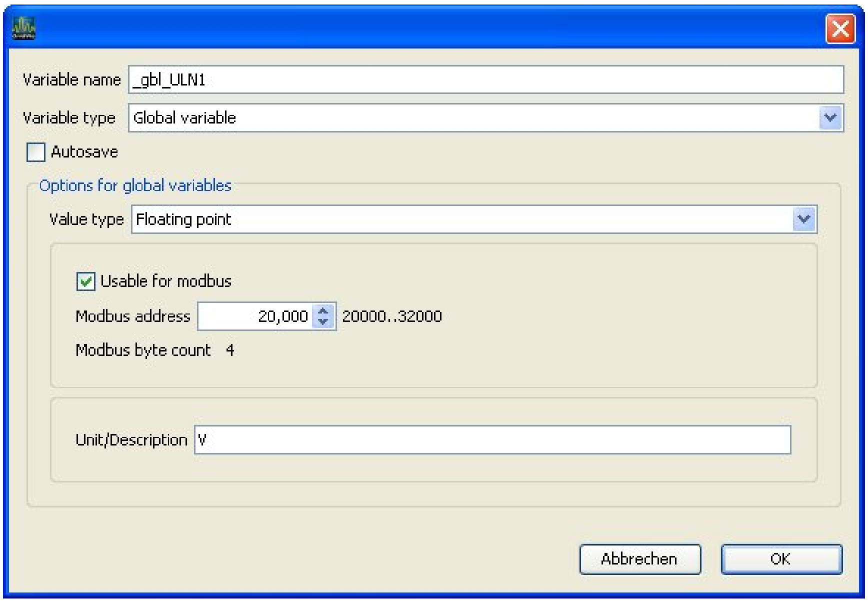
Variables

	Name for value	Format of value at modbus-sl...
1	ULN1_UMG103	Floating Point (32 bit,Big-Endian)

Illustration 19

The modbus function must correspond with the slave which is to be read out otherwise the correct values cannot be inputted!

The following illustration shows the configuration of the user variable. The variable is kept global and is made available in the internal modbus register on address 20000.



Variable name: _gbl_ULN1

Variable type: Global variable

☐ Autosave

Options for global variables

Value type: Floating point

☑ Usable for modbus

Modbus address: 20,000 20000..32000

Modbus byte count 4

Unit/Description: V

Illustration 20

All global variables must start with **_gbl**!

There is the opportunity of specifying modbus register addresses in illustration 21. This function can be used to transfer internal values to a slave unit.

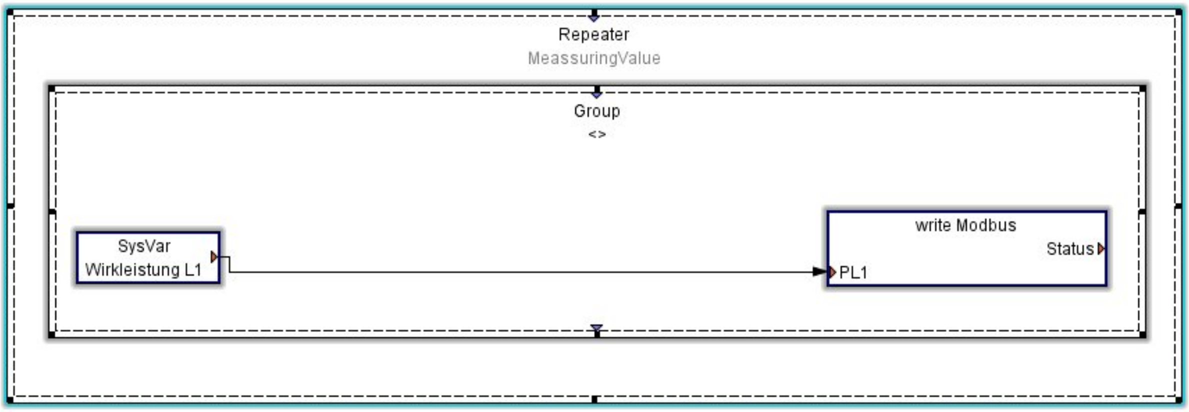
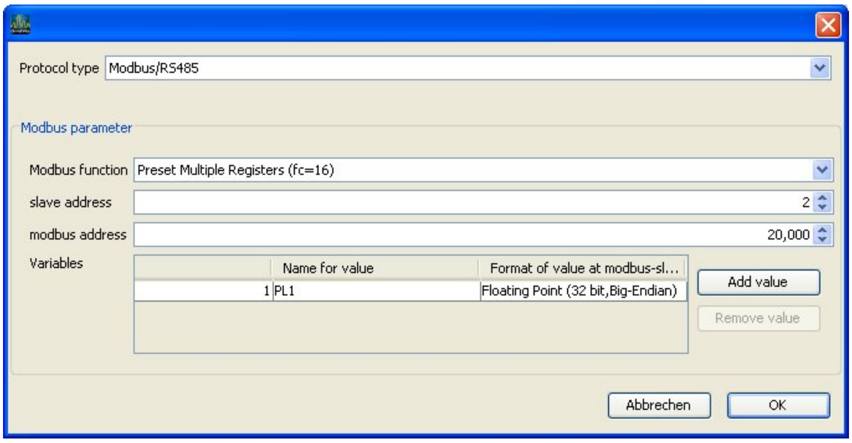


Illustration 21

The following illustration shows the configuration for the “write modbus” function module. The settings are the same as the “read modbus” function modules.



Protocol type: Modbus/RS485

Modbus parameter

Modbus function: Preset Multiple Registers (fc=16)

slave address: 2

modbus address: 20,000

Variables

	Name for value	Format of value at modbus-sl...
1	PL1	Floating Point (32 bit,Big-Endian)

Illustration 22

The read and write modbus is an extra option for the GridVis software. This licence must be paid for! If the licence has not been bought, the modules have a grey background.

For a serial readout or specification of a slave, the interface must be configured to the master!

Recording user-defined variables

You can record a user-defined variable with an average value with the “recording” function module. The following example shows the reading of the L1 voltage from a slave unit; the value is directed to an internal user variable. The produced variable is entered in the recording module and is stored in the ring buffer. This saved value can then be analysed later using the GridVis software e.g. by using the graph function.

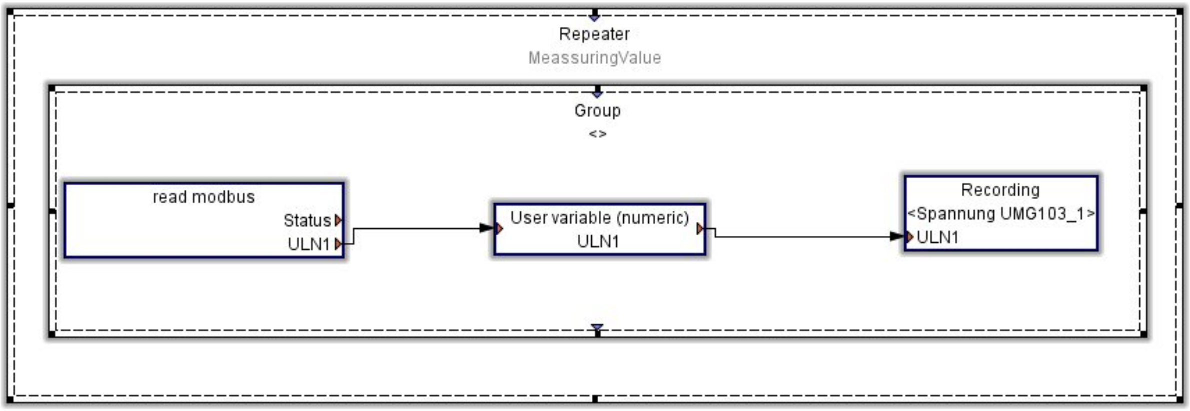
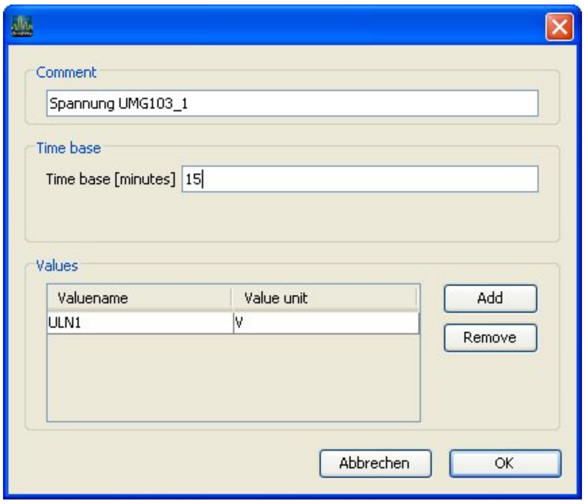


Illustration 23

Illustration 24 shows the configuration of the recording function. The time basis determines the average at which the recording should take place. In the below “values” frame, the value names must be given to the variables. It is also possible to parameterise several values. For each variable which is lodged here, a new pin is directed outwards and can then be connected.


Comment: Spannung UMG103_1

Time base [minutes]: 15

Valuename	Value unit
ULN1	V

Illustration 24

All user-defined recordings are saved in the GridVis database under the name “user-defined values“. The index shows the names of the value names.

Values and measurement data is only made available in the GridVis database when this is read out from the measuring instrument.

“Log” diagnosis analysis of the graphic programme code

The log module can be used for diagnosis or analysis of the programme code. The log function can also be used for troubleshooting. The input pin is black and can be supplied with any variable.

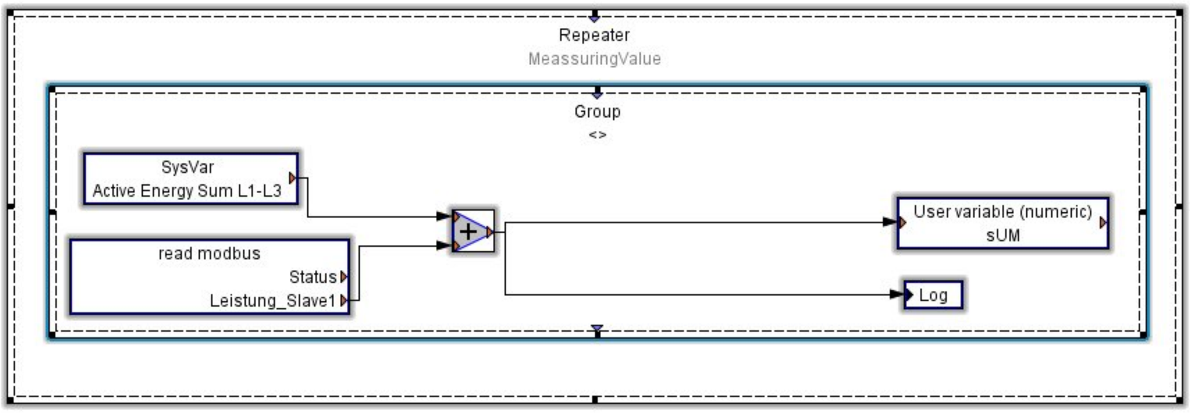
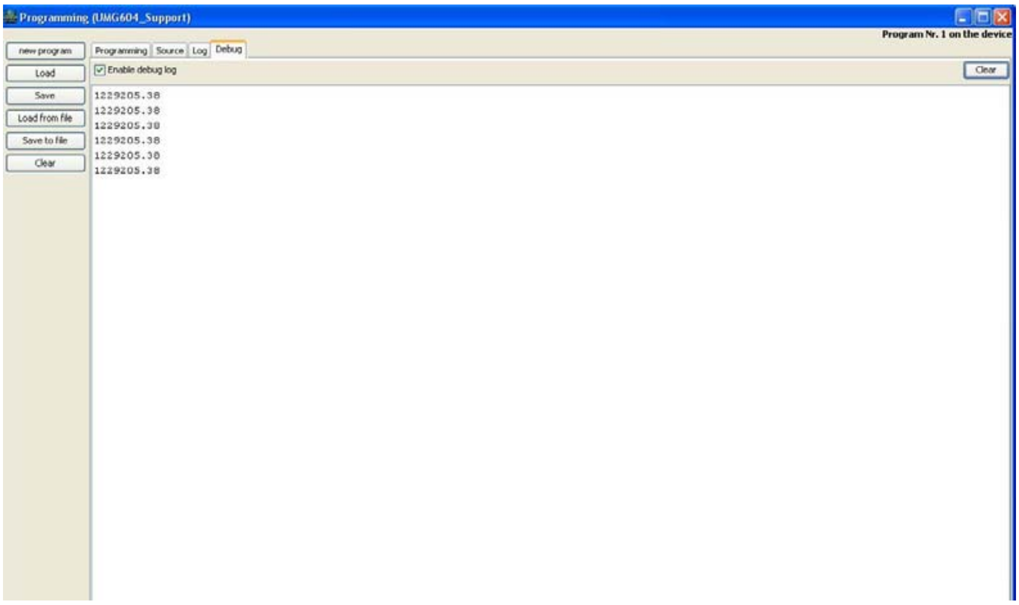


Illustration 25

The programme code may only be transferred to the measuring instrument once the programme has been processed and can be analysed.

We can now view the log value in the debug tab. The “Enable Debug Log” box must be ticked. The current value list can be deleted with the clear button.



☑ Enable debug log

```
1229205.38
1229205.38
1229205.38
1229205.38
1229205.38
1229205.38
```

Illustration 26

Tariff switchover with weekly time switch

A signal can be issued at specified times with the help of the weekly time switch. A main process time, auxiliary process time switchover, has been created in the following example. The consumption value is hereby created in two separate variables and can be analysed separately at a later date.

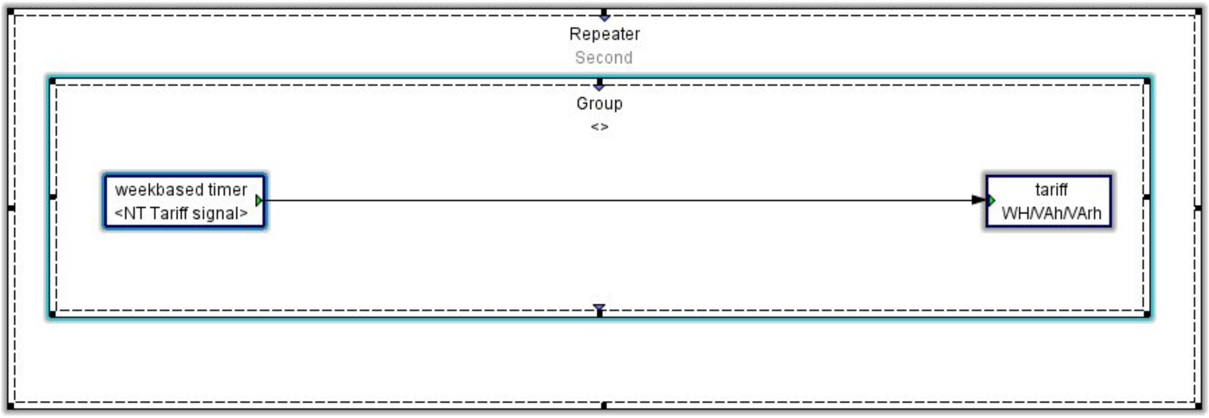
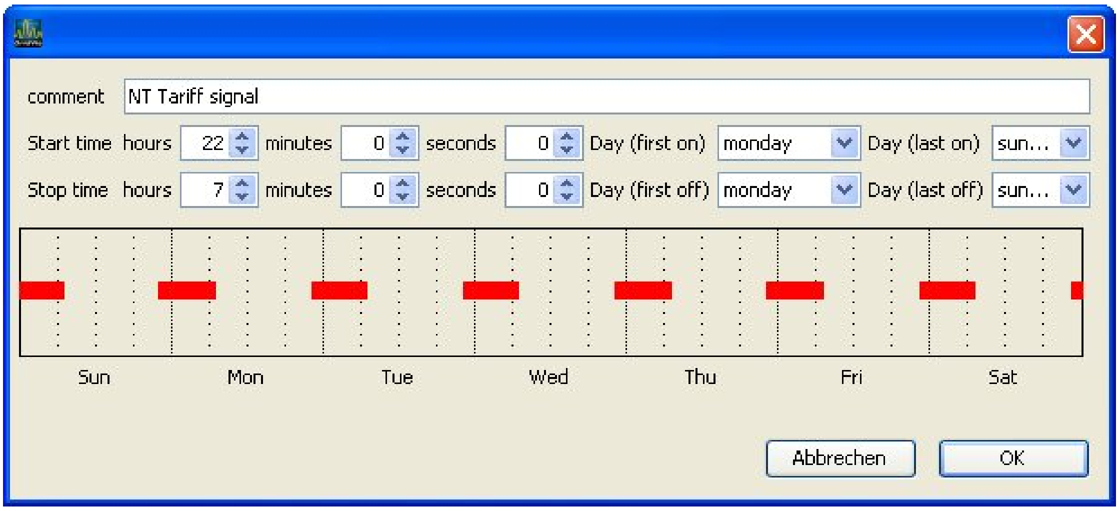


Illustration 27

Illustration 28 shows the configuration of the weekly time switch. The signal which is to be issued is parameterised with the initialisation and shutdown times.

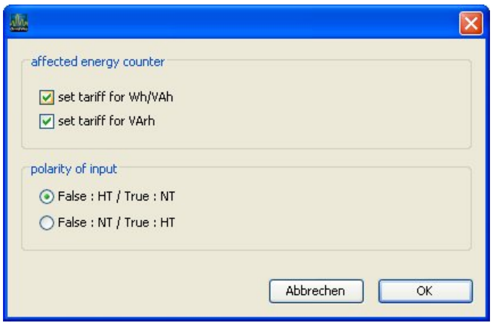


comment: NT Tariff signal

Start time hours 22 minutes 0 seconds 0 Day (first on) monday Day (last on) sun…

Stop time hours 7 minutes 0 seconds 0 Day (first off) monday Day (last off) sun…

Illustration 28



affected energy counter

☑ set tariff for Wh/VAh

☑ set tariff for VArh

polarity of input

● False : HT / True : NT

○ False : NT / True : HT

The left-hand configuration is for tariff switchover.

Working with the timer

We can work with the timer when we need specific time intervals in the programming. This is also applied in a group. The timer starts when the group is called up. The next group is called up after the parameterised time has elapsed.

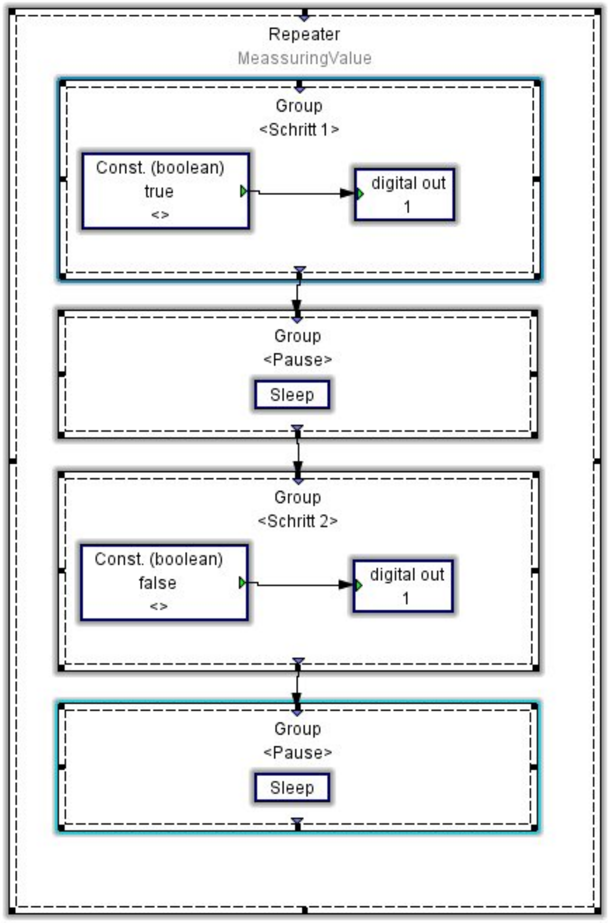
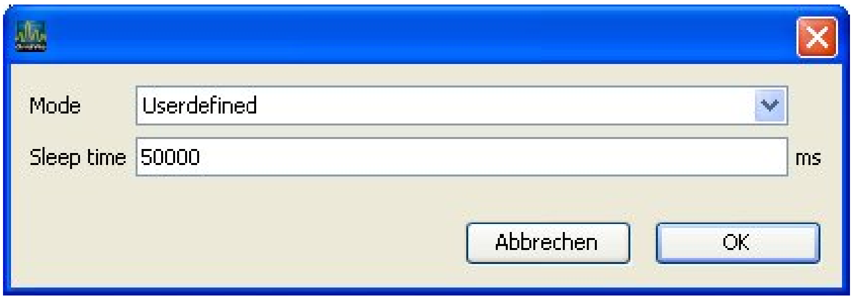


Illustration 29

Illustration 31 shows the configuration window for the timer.


Mode: Userdefined

Sleep time: 50000 ms

Illustration 30

IF requests „IF...THEN”

Illustration 32 shows programming with an IF request; the interconnection of several IF requests is possible. If you point the cursor on the IF function pin, you receive information about the status with which the pin is switched. After an IF function, there are two groups which can then be processed depending upon the status.

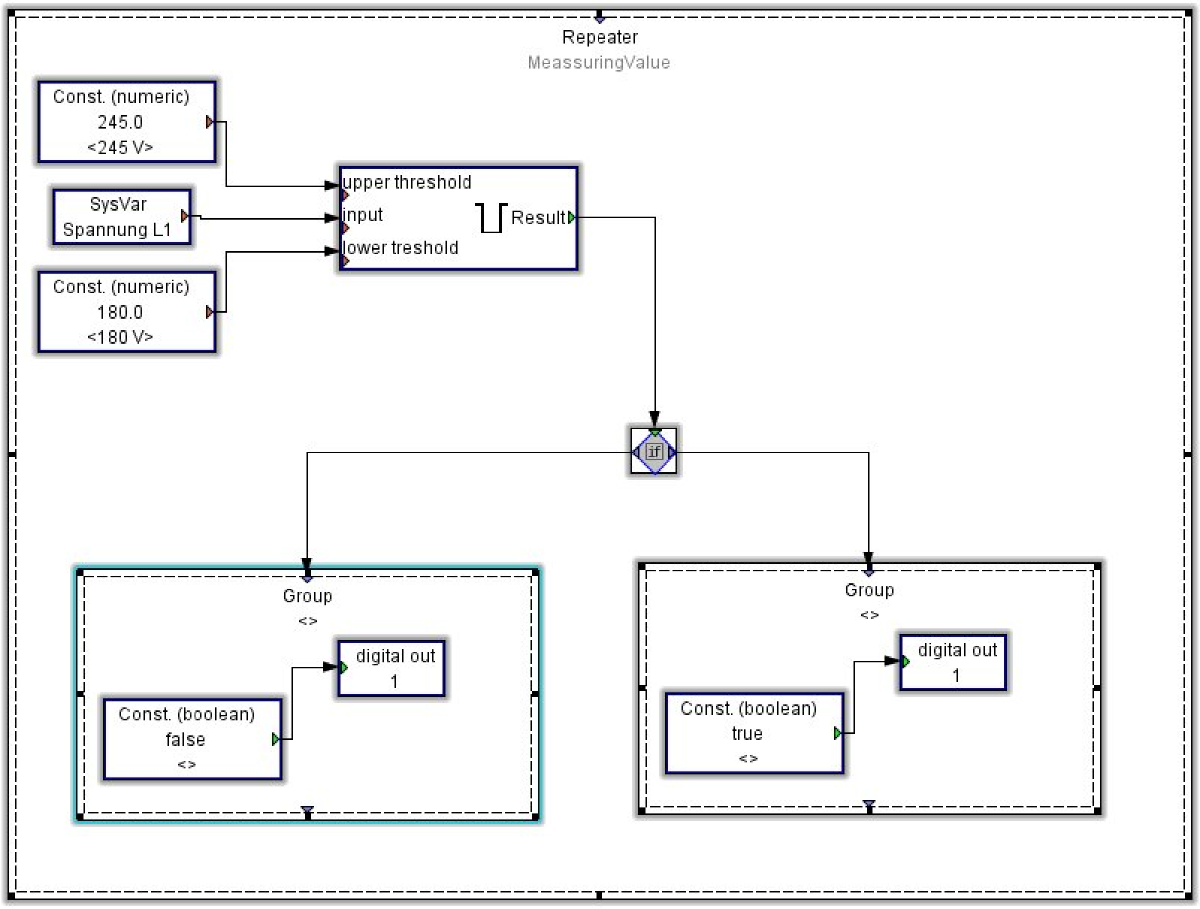


Illustration 31

Hints and tips

- ℹ The function code is only processed after transfer to the unit.
- ℹ Connect the pins by holding down the left button on the mouse.
- ℹ You will be asked for a programme after saving. Seven programmes are available and the previously existing code is overwritten.
- ℹ Modules can be deleted and copied with a right click.
- ℹ Work with the groups; these can be easily copied and then reconfigured.