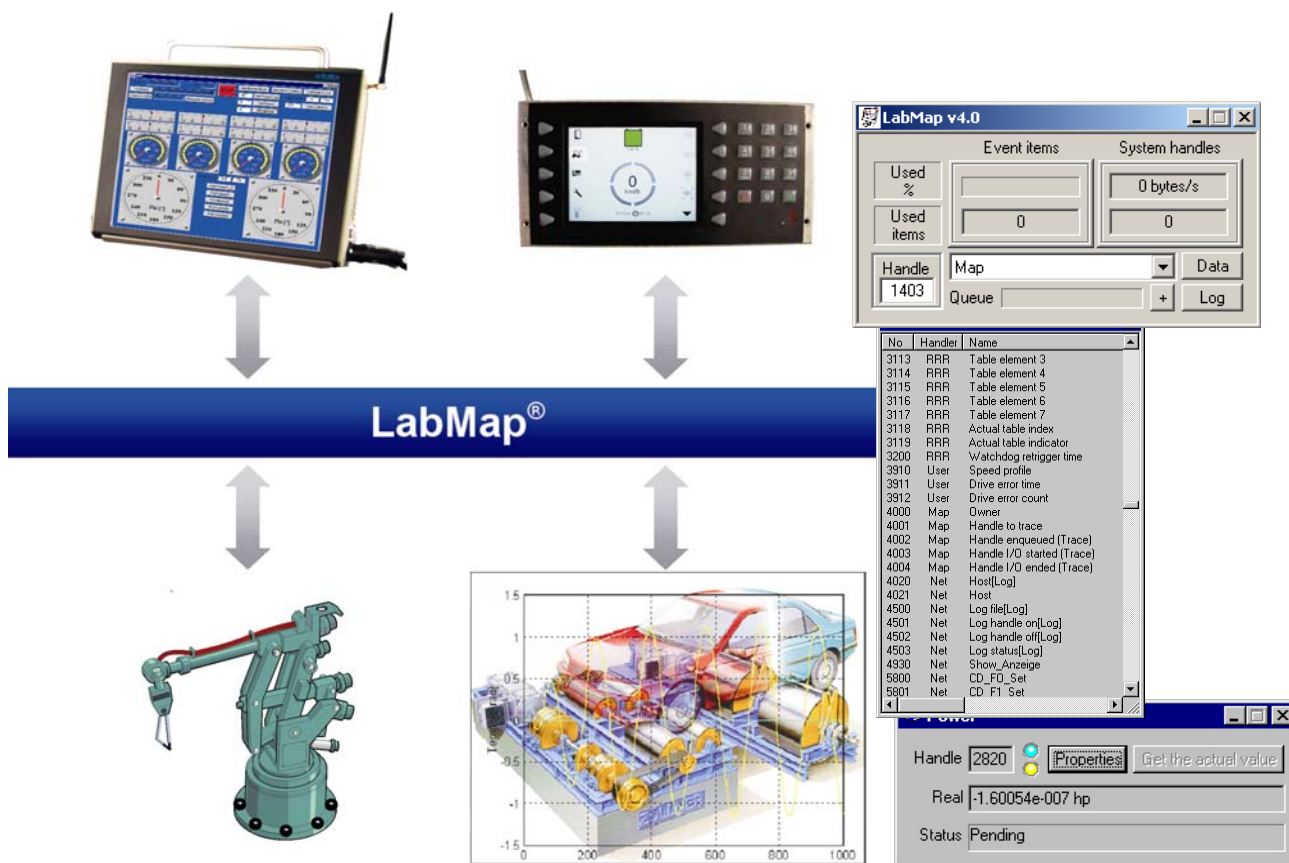


Softwarebus LabMap® User guide



Author: Dipl.-Ing. R. zum Beck

The 2nd edition, April 2004

Contents:

1. Introduction	4
2. Concepts.....	5
2.1 Interfaces	8
3. LabMap[®] Installation.....	9
3.1 Adding Interfaces	9
3.1.1 Creating Handles for User and Virtual Interface.....	10
3.1.2 Creating Handles for ModBus TCP/IP Interface.....	11
3.1.3 Creating Handles for Net Interface.....	11
4. Using Wago modules	12
4.1 Starting up ETHERNET TCP/IP fieldbus node	13
4.1.1 Note the MAC-ID and establish the fieldbus node.....	13
4.1.2 Connecting PC and fieldbus node.....	14
4.1.3 Determining IP addresses.....	14
4.1.4 Allocating the IP address to the fieldbus node	15
4.1.5 Testing the function of the fieldbus node	18
4.1.6 Reading out the information as HTML pages.....	19
4.2 Accessing Wago modules with LabMap[®]	20
4.2.1 Analogue input modules:.....	21
4.2.2 Analogue output modules:	21
4.2.3 Digital input modules:.....	21
4.2.4 Digital output modules:	21
5. Access Library-Functions from LabMap[®] in LabView[®]	22
6. LabMap[®] Interface for WinFACT[®]/Boris.....	26
6.1 Installation of the LabMap[®] / WinFACT[®] Interface Dll's	26
6.2 Using the LabMap[®] / WinFACT[®] Interface	26

7. Examples	28
7.1 Virtual Channels to limit input values	28
7.2 Feedback System with Boris, Wago modules and LabMap[®]	29
7.2.1 WinFACT [®] /Boris feedback model	30
7.2.2 LabMap [®] register configuration:.....	32
7.3 Filtering a signal with Virtual-Registers	33
7.3.1 LabMap [®] register configuration:.....	34
7.3.2 Using the filter	34
7.3.3 Example filter test.....	35
7.4 Monitoring data from LabMap[®] with LabView[®]	36
7.4.1 Getting access to LabMap [®] with the LabView [®] VI's.....	37
7.4.2 Using the LM_LabView_test.vi.....	39
7.5 Monitoring data from LabMap[®] with Matlab[®].....	40
7.5.1 Working with LabMap [®] /Matlab [®] blocks	41
7.5.2 Using the LM_Matlab_test.mdl.....	41
8. Distributing data over the network with LabMap[®]	43
8.1 Example Distribute data over the network.....	45
Appendix: Wago Modul description	47
Bibliography.....	57

1. Introduction

LabMap® is a 32-bit MS-Windows® multithreading application providing a high level interface for industrial communication, control, measurement, and logging. It allows asynchronous viewing and modification of the system variables from a potentially unlimited number of applications running on the same or different LAN/WAN network hosts. **LabMap®** has an open architecture supporting smooth integration of new hardware and software components running on different bus architectures. It has integrated support of measurement units and time stamping of the system variables. The software functions under Microsoft Windows® (x86 platform).

2. Concepts

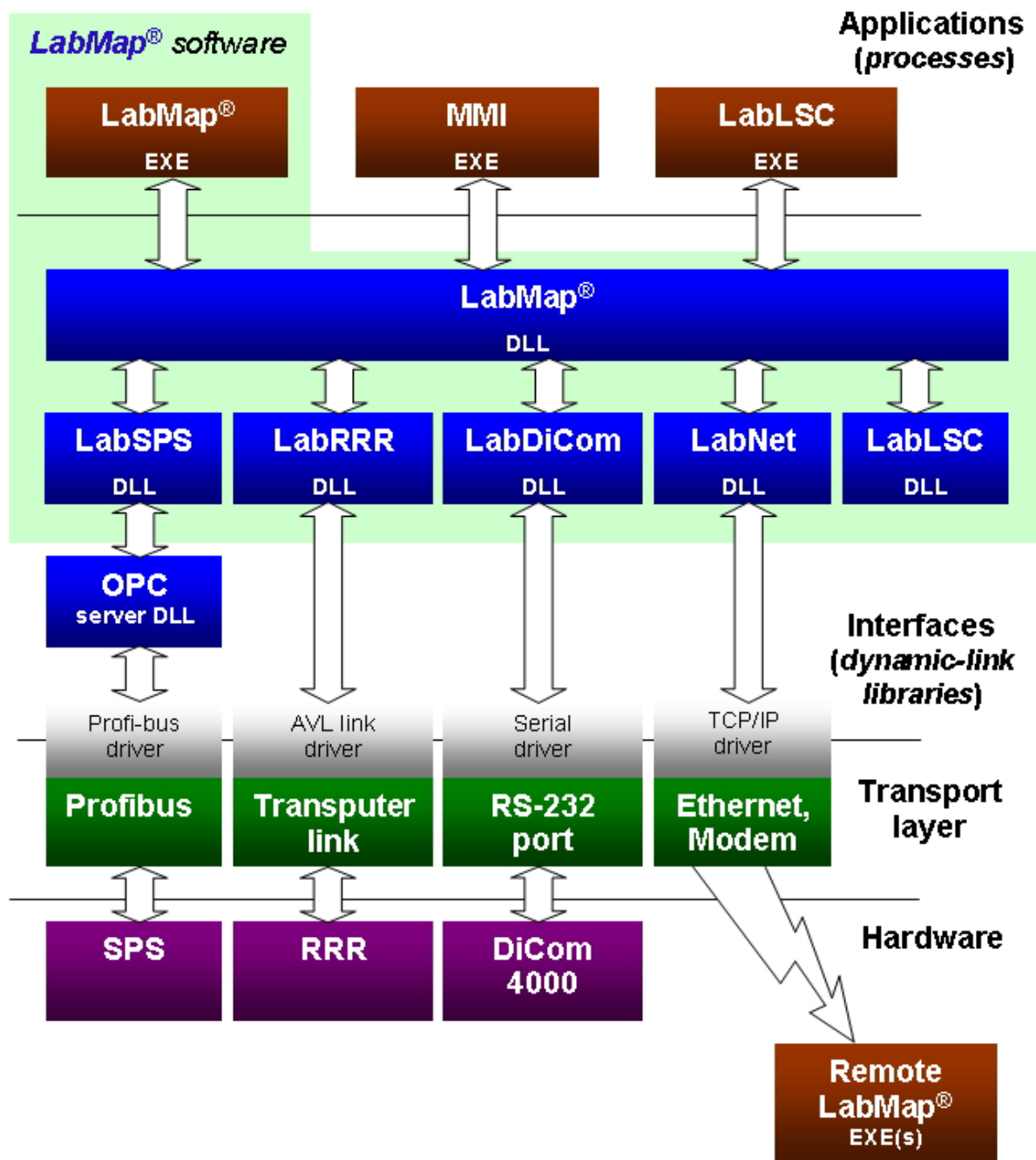
LabMap[®] operates as a software bus system. It offers an abstraction of the application level from the hardware specific and decoupling the hardware interface modules from the application level. In other words there are two levels of abstraction supported by the bus system:

- The application interface.
- The hardware driver interface

The application abstraction interface exposes LabMap[®] as a set of variables (registers). Each register has a type, value, timestamp and I/O direction. The software structure allows user-written applications to access the system variables and parameters over the standard interface independently from the actual hardware configuration. The programmer's interface is provided by LabMap.dll. It implements user requests using the hardware specific interfaces. The software has an open architecture. Hardware interfaces can be added at needs without modification of LabMap.dll and user applications (see LabMap[®] Handbook).

The hardware abstraction interface of LabMap[®] allows easy integration of new hardware devices and communication protocols into the system. Each class of hardware devices or communication protocol supported by LabMap[®] is represented by a hardware interface. The hardware interface is responsible for implementation of the basic I/O operations upon the registers assigned to it. LabMap[®] uses a messages mechanism for interacting with the hardware interfaces. A hardware interface is developed as dynamic-link library which code can be maintained independently from the LabMap[®] core.

The general structure of LabMap[®] is shown on the next picture:



The software structure allows user-written applications to access the system variables and parameters over the standard interface independently from the actual hardware configuration. The programmer interface is provided by LabMap.dll. It implements user requests using the hardware specific interfaces. The software has an open architecture. Hardware interfaces can be added at needs without modification of LabMap.dll and user applications. The following hardware interfaces are currently supported:

- AK interface implemented by the LabAK.dll provides access to the devices
- supporting the AK protocol.

- *DiCom* interface implemented by LabDiCom.dll provides access to the AVL[®] DiCom 4000 device.
- *ExtFix* interface implemented by LabExtFix.dll provides access to the external fixation device used in the Medizinische Universität zu Lübeck.
- *FLG* interface implemented by LabFLG.dll provides access to an arbitrary number of FLG servers.
- *Net* interface implemented by the LabNet.dll. The Net interface allows building systems distributed over TCP/IP networks. The interface is delivered as a plug-in upon request.
- *LSC* interface implemented by the LabLSC.dll. The extensible software interface is provided for the LSC application maintained by AVL Zoellner GmbH. The LSC application is a software extension of LabMap[®] software. It responds to no hardware directly, rather uses other interfaces to implement a complex operation visible for other applications as simple variables.
- *ModBus* interface implemented by the LabModBus.dll provides access to MODBUS/TCP hosts.
- *NI-CAN* interface implemented by the LabNICAN.dll provides access to the CAN-bus (the **C**ontroller **A**rea **N**etwork) via an adapter card by National Instruments.
- *PLU* interface implemented by the LabPLU.dll provides access to the Pierburg PLU 401 device.
- *RRR* interface implemented by the LabRRR.dll. The RRR (German abbreviation of "dynamometer controlling computer") is a computer (based on the inmos[®] T805 processor) controlling in real-time the roller dynamometer. The RRR is connected to the host computer via transputer link using AVL[®] 6801A01 transputer link card. For more information, please, refer to Zoellner documentation.
- OPC (named SPS) interface implemented by the LabSPS.dll. The LabSPS.dll uses the OPC interface to communicate with the devices controlled by an OPC server. For the interface description see.
- Remote control interface implemented by LabRC.dll provides access to Volltronic's remote control device.
- *Virtual interface* implemented by LabVirt.dll provides registers calculated using some formula.
- User interface implemented by LabUser.dll provides user variables.

LabMap.exe provides dialogue based user interface. The executable and DLL files should be placed into the Windows home directory and the system registry should be configured. LabMap[®] may be started either manually by invoking the LabMap.exe or automatically by starting an application that uses any of interface DLLs.

2.1 Interfaces

Hardware and software interfaces required are:

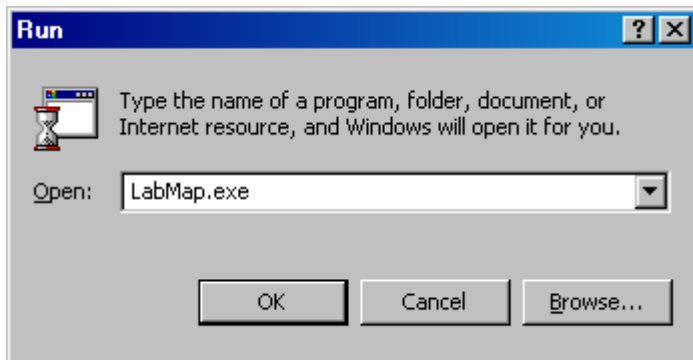
- ModBus interface implemented by the LabModBus.dll provides access to MODBUS/TCP hosts.
- Virtual interface implemented by LabVirt.dll provides registers calculated using some formula.
- User interface implemented by LabUser.dll provides user variables.
- Remote access interface implemented by LabNet.dll provides server and client configuration over Ethernet.

3. LabMap® Installation

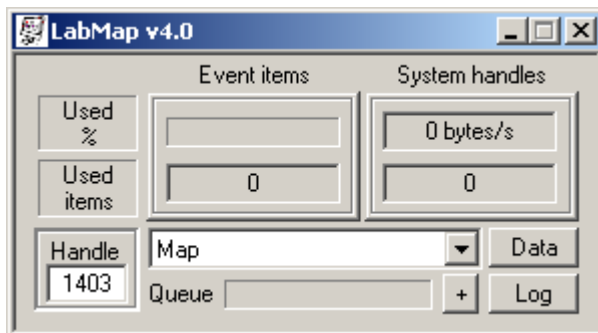
To install LabMap® software insert the CD into your drive, the CD normally will starts automatically. Otherwise double-click on setup.exe.

3.1 Adding Interfaces

Start the LabMap.exe from the menu bar Start/Programms/cbb software GmbH/LabMap or from system Start menu Run.

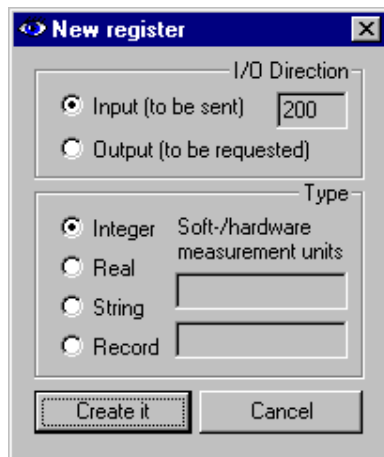


After running LabMap® the main panel starts.



To add specified interfaces to the actual configuration create a handle of the type of the interface.

3.1.1 Creating Handles for User and Virtual Interface



Type a handle number (e.g. 200) in the Handle field and press RETURN and New Register dialog pops up.

Here select the appropriate parameters and press the Create it dialog button.

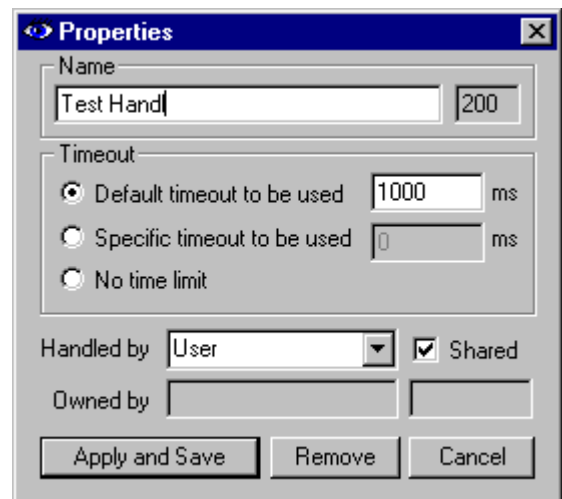
The Properties dialog pops up.

Give the new Handle a name and set the Timeout properties and then in the Handled by field at first time type the name of the interface, here "User".

For the Virtual Interface type the name "Virt".

For further information see [LabMap® Handbook section 5.11](#).

Apply and Save the properties dialog.



3.1.2 Creating Handles for ModBus TCP/IP Interface

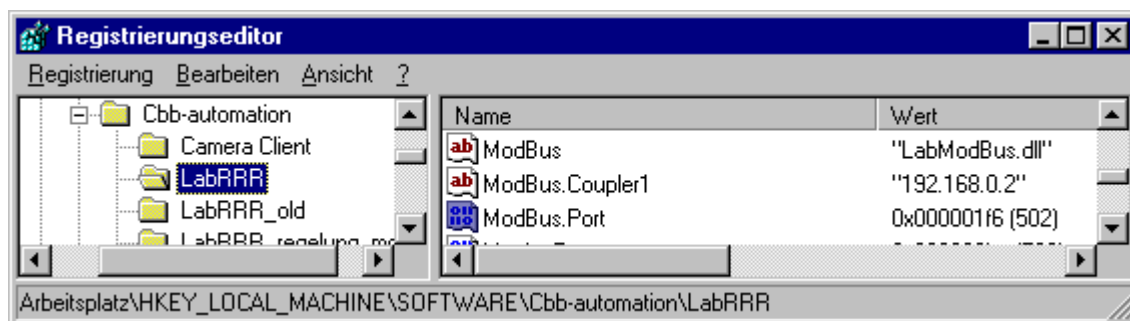
Set up the ModBus Hardware as described in section 5.1 and follow the same procedure as in section 3.1.1. In the properties dialog insert the properties for the ModBus modules in brackets after the name. For e.g.:

Name [Coupler1.type=i.addr=0.len=16] .

For further information see the [LabMap®](#) Handbook section 5.13.

Create and define the required keys for the ModBus interface in the Registry. These keys:

- *ModBus*
- *ModBus.Coupler1*
- *ModBus.Port*



The Coupler value is the IP address from the Ethernet ModBus coupler given when the coupler is first installed by the installation software program *Wago BootP Server* from Wago. The port value is a predefined port number of the Ethernet coupler.

3.1.3 Creating Handles for Net Interface

Follow the same procedure as in section 3.1.1.

The name of the Net interface is Net.

In the properties dialog insert the properties for the Net Interface in brackets after the name. For e.g.:

Speed [2000.SERVER_NAME]

Where *Speed* is the name assigned to the Net handle, *2000* is the handle number on the remote computer and the *SERVER_NAME* is the one assigned to the host. For the necessary client and server configuration and all required keys refer to Section 5.4 of the [LabMap®](#) handbook).

4. Using Wago modules

The **WAGO-I/O SYSTEM** comprises of various components which allow the creation of modular and user specific fieldbus nodes for various fieldbusses.

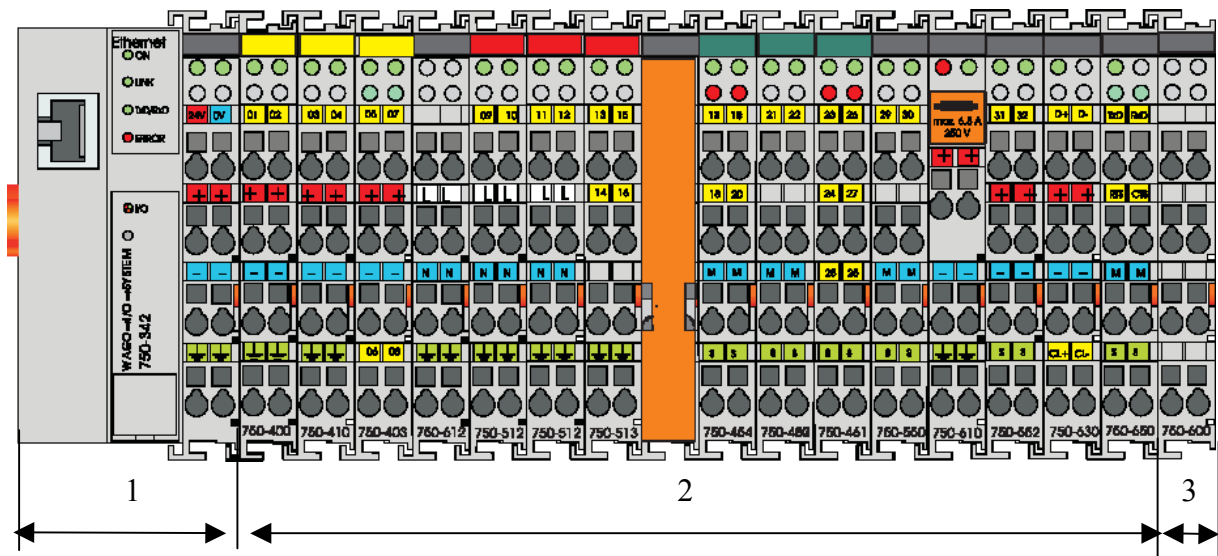


Fig. 4.1: Structure of a fieldbus node with the WAGO-I/O SYSTEM

General:

In all cases a fieldbus node comprises of a fieldbus coupler (1) or a programmable fieldbus controller (1) as head station, a number of I/O modules (2) and an end module (3), which forms the end of the system.

1 – Coupler/Controller:

With its I/O functions the coupler/controller forms the logic operation between the fieldbus used and the field area. All control tasks necessary for the perfect operation of the I/O are performed by the coupler/controller. The connection to different fieldbus systems is made using the corresponding coupler/controller e.g. for PROFIBUS, INTERBUS, CAN, MODBUS etc. A retrofitting to a different fieldbus system by changing the coupler is possible. As opposed to the coupler the controller is fitted with additional PLC functions. This permits signal pre-processing, which can considerably reduce the data quantity in the network. In the case of a fieldbus failure the controller can process the control program independently. Alternatively the controller can guide the node into a defined condition. Plant modules become independent testable units due to the controller. In the delivered condition, in other words without the user program, the controller behaves as a coupler.

2 - I/O modules

The input and output of the process data is made at the I/O modules. I/O modules are available for various tasks in accordance with varying requirements. Available are digital and analog input and output modules, I/O modules for angle and path measurement as well as communication modules.

3 - End module

The node end module is indispensable. It is always fitted as the last module, to guarantee the internal node communication. The end module has no I/O function.

4.1 Starting up ETHERNET TCP/IP fieldbus node

This chapter shows the step-by-step procedure for starting up a WAGO ETHERNET TCP/IP fieldbus node. The following also contains a description of how to read out the coupler-internal HTML pages.

Attention

This description is given as an example and is limited to the execution of a local startup of an individual ETHERNET fieldbus node with a computer running under windows which is not connected to a network. Direct Internet connection should only be performed by an authorized network administrator and is, therefore, not described in this manual. The procedure contains the following steps:

1. Noting the MAC-ID and establishing the fieldbus node
2. Connecting the PC and fieldbus node
3. Determining the IP address
4. Allocation of the IP address to the fieldbus node
5. Function of the fieldbus tests
6. Reading out information as HTML pages

4.1.1 Note the MAC-ID and establish the fieldbus node

Before establishing your fieldbus node, please note the hardware address (MAC-ID) of your ETHERNET fieldbus coupler. This is located on the rear of the fieldbus coupler and on the self-adhesive tearoff label on the side of the fieldbus coupler. MAC-ID of the fieldbus coupler will be in this format:

-----.

4.1.2 Connecting PC and fieldbus node

Connect the assembled ETHERNET TCP/IP fieldbus node via a hub or directly to the PC using a 10Base-T cable.

Attention

For a direct connection, a crossover cable is required instead of a parallel cable.

Now start the PC, functioning as master and BootP server, and switch on the voltage supply on the fieldbus coupler (DC 24 V power pack). Once the operating voltage has been switched on, the initialization starts. The fieldbus coupler determines the configuration of the bus modules and creates the process image. During the startup the 'I/O' LED (Red) flashes at high frequency. When the 'I/O' LED and the 'ON' LED light up green, the fieldbus coupler is ready for operation. If an error has occurred during startup, it is indicated as an error code by the 'I/O'-LED flashing (red).

4.1.3 Determining IP addresses

If your PC is already connected to an ETHERNET network, it is very easy to determine the IP address of your PC. To do this, proceed as follows:

1. Go to the **Start menu** on your screen, menu item **Settings** and click on **Control Panel**.
2. Double click the icon **Network**.
The network dialog window will open.
3. Under Windows NT: Select the register: **Protocols** and mark the entry *TCP/IP protocol*.
Attention
If the entry is missing, please install the respective TCP/IP component and restart your PC. The Windows-NT installation CD is required for the installation.
4. Subsequently, click the button "Properties...".
The IP address and the subnet mask are found in the 'IP address' tab. If applicable, the gateway address of your PC is found in the 'Gateway' tab.
5. Please write down the values:
IP address PC: ----- . ----- . ----- . -----
Subnet mask: ----- . ----- . ----- . -----
Gateway: ----- . ----- . ----- . -----
6. Now select a desired IP address for your fieldbus node.
Attention
When selecting your IP address, ensure that it is in the same local network in which your PC is located.
7. Please note the IP address you have chosen:
IP address fieldbus node: ----- . ----- . ----- . -----

4.1.4 Allocating the IP address to the fieldbus node

The following describes how to allocate the IP address for the fieldbus node using the WAGO BootP server by way of an example. You can download a free copy from WAGO over the Internet under:

<http://www.wago.com/wagoweb/usa/eng/support/downloads/index.htm>.

Attention

The IP address can be allocated in a direct connection via a crossover cable or via a parallel cable and a hub. An allocation over a switch is not possible.

BootP table

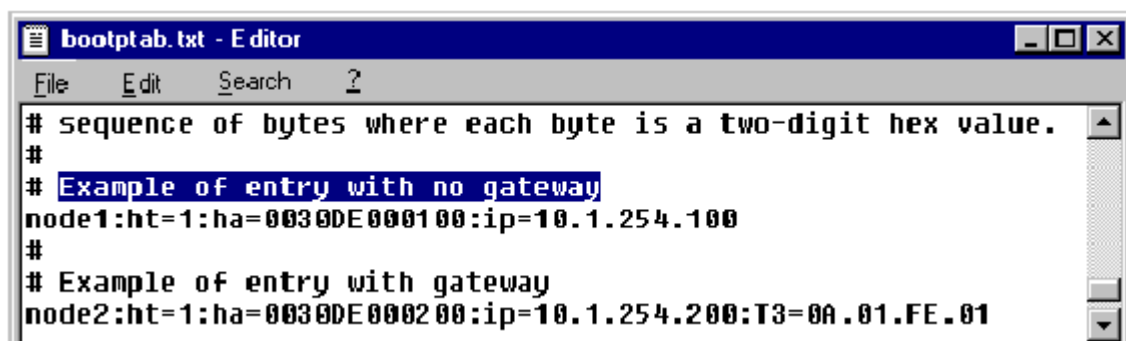
Note

Prerequisite for the following steps is the correct installation of the WAGO BootP server.

1. Go to the **Start menu**, menu item **Programs / WAGO Software / WAGO BootP Server** and click on **WAGO BootP Server configuration**.

An editable table will appear: "bootptab.txt".

This table displays the data basis for the BootP server. Directly following the list of all notations used in the BootP table there are two examples for the allocation of an IP address. **"Example of entry with no gateway"** and **"Example of entry with gateway"**.



```

bootptab.txt - Editor
File Edit Search ?
# sequence of bytes where each byte is a two-digit hex value.
#
# Example of entry with no gateway
node1:ht=1:ha=0030DE000100:ip=10.1.254.100
#
# Example of entry with gateway
node2:ht=1:ha=0030DE000200:ip=10.1.254.200:T3=0A.01.FE.01
  
```

Fig. 4.2 Boot table

The examples mentioned above contain the following information:

Declaration	Meaning
node1, node2	Any name can be given for the node here.
ht=1	Specify the hardware type of the network here. The hardware type for ETHERNET is 1. (The numbers are described in <i>RFC1700</i>)
ha=0030DE000100 ha=0030DE000200	Specify the hardware address or the MAC-ID of the ETHERNET fieldbus coupler (hexadecimal).
ip= 10.1.254.100 ip= 10.1.254.200	Enter the IP address of the ETHERNET fieldbus coupler (decimal) here.
T3=0A.01.FE.01	Specify the gateway IP address here. Write the address in hexadecimal form.
sm=255.255.0.0	In addition enter the Subnet-mask of the subnet (decimal), where the ETHERNET fieldbus coupler belongs to.

No gateway is required for the local network described in this example. Therefore, the first example: "**Example of entry with no gateway**" can be used.

2. Move the mouse pointer to the text line:
"node1:ht=1:ha=0030DE000100:ip=10.1.254.100" and mark the 12 character hardware address which is entered after ha=...
Enter the MAC-ID of your own network coupler.
3. If you want to give your fieldbus node a name, delete the name "node1"
And enter any name in its place.
4. To assign the coupler a desired IP address, mark the IP address specified in the example which is entered after ip=...
Replace it with the IP address you have selected.
5. Because the second example is not necessary at present, insert a "#" in front of the text line of the second example:
"# node2:ht=1:ha=003 0DE 0002 00:ip=10.1.254.200:T3=0A.01.FE.01", so that this line will be ignored.
Note
To address more fieldbus nodes, enter a corresponding text line showing the corresponding entries for each node.
6. Save the altered settings in this text file "bootptab.txt". To do this go to the **File** menu, menu item **Save**, and close the editor. **BootP Server**
7. Now open the dialog window for the WAGO BootP server by going to the **Start menu** on your screen surface, menu item **Program / WAGO Software / WAGO BootP Server** and click on **WAGO BootP Server**.
8. Click on the "**Start**" button in the opened dialog window. This will activate the inquiry/response mechanism of the BootP protocol. A series of messages will be displayed in the BootP server. The error messages indicate that some services (i.e. port 67, port 68) in the operating system have not been defined.

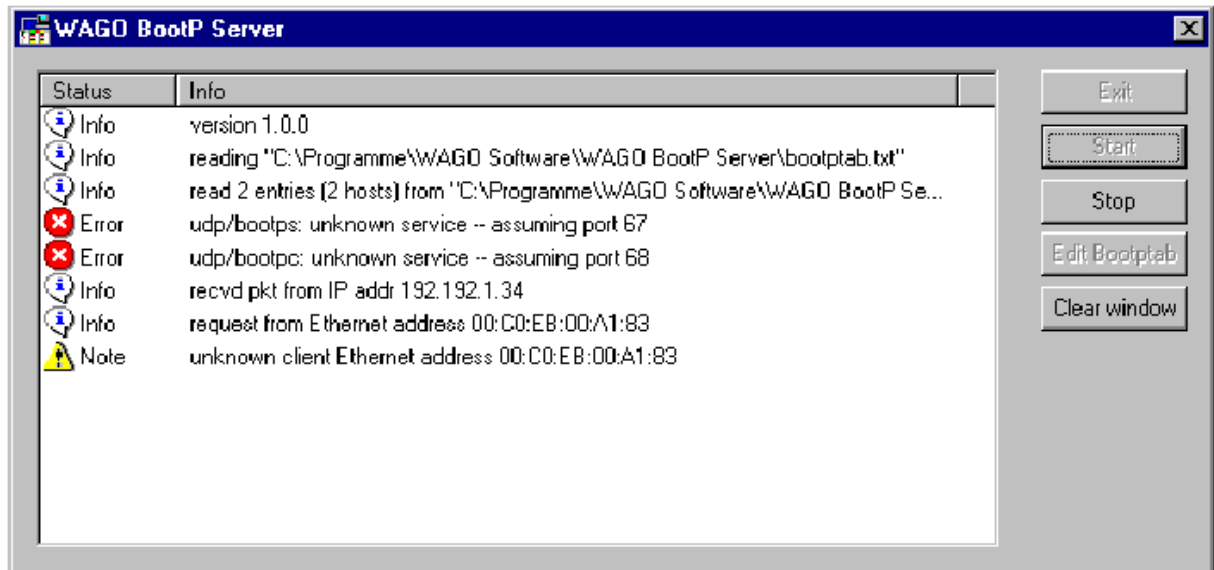


Fig. 4.3 Dialog window of the WAGO BootP server with messages

9. Now it is important to restart the coupler by resetting the hardware . This ensures that the new IP address will be accepted by the coupler. To do this, cycle power to the fieldbus coupler for approx. 2 seconds. Following this, the IP address in the coupler is permanently stored and maintained even once the coupler is removed or following a longer voltage failure.
10. Subsequently, click on the "Stop" button and then on the "Exit" button, to close the BootP Server again.

4.1.5 Testing the function of the fieldbus node

1. To test the communication with the coupler and the correct assignment of the IP address call up the DOS prompt under **Start menu / Program /MSDOS Prompt**.
2. Enter the command: "**ping**" with the IP address you have assigned in the following form:
ping [space] XXXX . XXXX . XXXX . XXXX (=IP address).
Example: ping 10.1.254.202

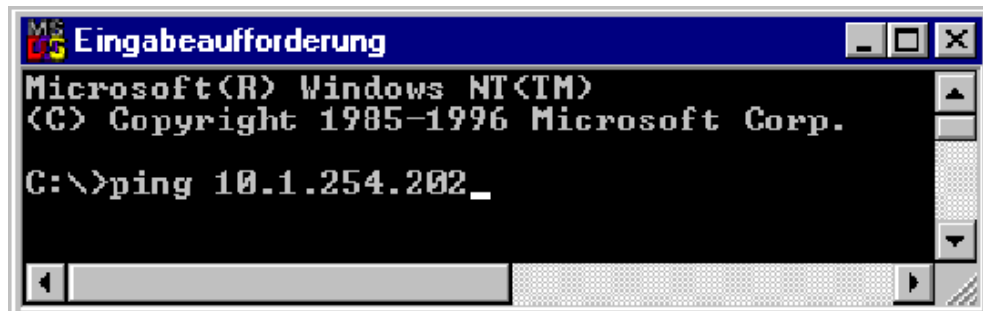


Fig. 4.4 Example for the function test of a fieldbus node

3. When the Return key has been pressed, your PC will receive a response from the coupler, which will then be displayed in the DOS prompt. If the error message: "Timeout" appears instead, please compare your entries again to the allocated IP address.
4. When the test has been performed successfully, you can close the DOS prompt. The network node has now been prepared for communication.

4.1.6 Reading out the information as HTML pages

The information saved in the fieldbus coupler can be read as an HTML page using a web browser.

- Information on the fieldbus node (Terminal Status):
 - Number of digital, analog or complex modules
 - Representation of the process image
- Information on the fieldbus coupler (Coupler and Network Details):
 - Order number
 - Firmware version
 - MAC-ID
 - IP address
 - Gateway address (if applicable)
 - Subnet mask
 - Number of transmitted and received packets
- Diagnostic information on the fieldbus coupler (Coupler Status):
 - Error code
 - Error argument
 - Error description

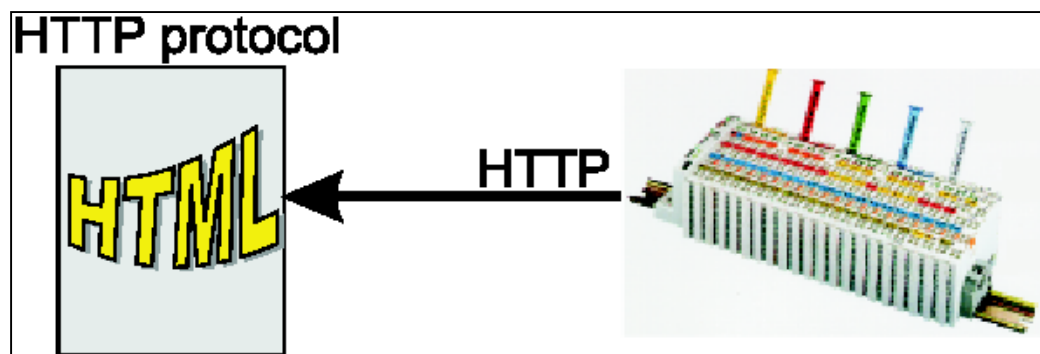


Fig. 4.5: Reading out the information via the HTTP protocol

1. Open a web browser such as Microsoft Internet-Explorer, Netscape Navigator, ...
2. Simply enter the IP address of your fieldbus node in the address field of the browser and press the Return key. The first HTML page with the information on your fieldbus coupler will be displayed in the browser window. Use the hyperlinks to find out more information.

Attention

If the pages are not displayed after local access to the fieldbus node, then define in your web browser that, as an exception, no proxyserver is to be used for the IP address of the node.

4.2 Accessing Wago modules with LabMap®

Assemble the Wago modules as described in Fig. 5.1, with the Bus coupler first followed by the I/O modules and then the end module. The statements that follow show a typical definition of how a I/O module is defined for LapMap as appears in the registry.

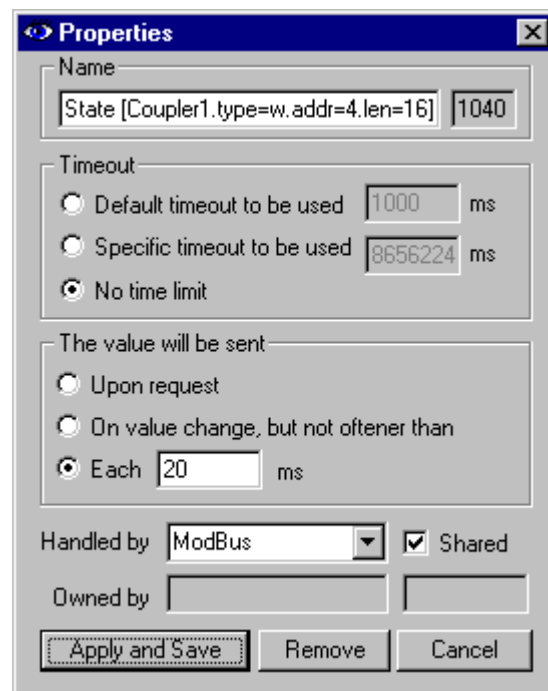
```
>ModBus>[::]!O:I:-1:20:5:317:State [Coupler1.type=w.addr=4.len=16]
```

<i>ModBus</i>	:	Interface name for ModBus
<i>!O</i>	:	sharing direction for output
<i>I</i>	:	data type integer
<i>-1</i>	:	no timeout
<i>20</i>	:	The value is to be sent periodically all 20 ms
<i>5</i>	:	X-coordinate of Handle window
<i>317</i>	:	Y-coordinate of Handle window
<i>State</i>	:	text string that defines the register name
<i>Coupler1</i>	:	Ethernet coupler Hostname
<i>type</i>	:	Integer
<i>addr</i>	:	indicates the channel number of the module type
<i>len</i>	:	data length 16 Bit



The direction and data type can be defined at creation time with the “New register” dialog.

All other values can be defined in the “Properties” dialog.



For details refer to section 5.1 in LabMap® Handbook

4.2.1 Analogue input modules:

Channel 1:

```
>ModBus>[::]!O:I:-1:-1:-1:-1:Wago Analog Input 1[Coupler1.type=int.addr=0.len=16]
```

Channel 2:

```
>ModBus>[::]!O:I:-1:-1:-1:-1:Wago Analog Input 2[Coupler1.type=int.addr=1.len=16]
```

Channel n:

```
>ModBus>[::]!O:I:-1:-1:-1:-1:Wago Analog Input n[Coupler1.type=int.addr=n-1.len=16]
```

4.2.2 Analogue output modules:

Channel 1:

```
>ModBus>[::]!!I:I:-1:-1:-1:-1:Wago Analog Output 1[Coupler1.type=int.addr=0.len=16]
```

Channel 2:

```
>ModBus>[::]!!I:I:-1:-1:-1:-1:Wago Analog Output 2[Coupler1.type=int.addr=1.len=16]
```

Channel n:

```
>ModBus>[::]!!I:I:-1:-1:-1:-1:Wago Analog Output n[Coupler1.type=int.addr=n-1.len=16]
```

4.2.3 Digital input modules:

Channel 1

```
>ModBus>[::]!O:I:-1:-1:-1:-1:Wago Digital Input 1[Coupler1.type=b.addr=0.len=1]
```

Channel 2

```
>ModBus>[::]!O:I:-1:-1:-1:-1:Wago Digital Input 2[Coupler1.type=b.addr=1.len=1]
```

Channel n

```
>ModBus>[::]!O:I:-1:-1:-1:-1:Wago Digital Input n[Coupler1.type=b.addr=n-1.len=1]
```

4.2.4 Digital output modules:

Channel 1

```
>ModBus>[::]!!I:I:-1:-1:-1:-1:Wago Digital Output 1[Coupler1.type=b.addr=0.len=1]
```

Channel 2

```
>ModBus>[::]!!I:I:-1:-1:-1:-1:Wago Digital Output 2[Coupler1.type=b.addr=1.len=1]
```

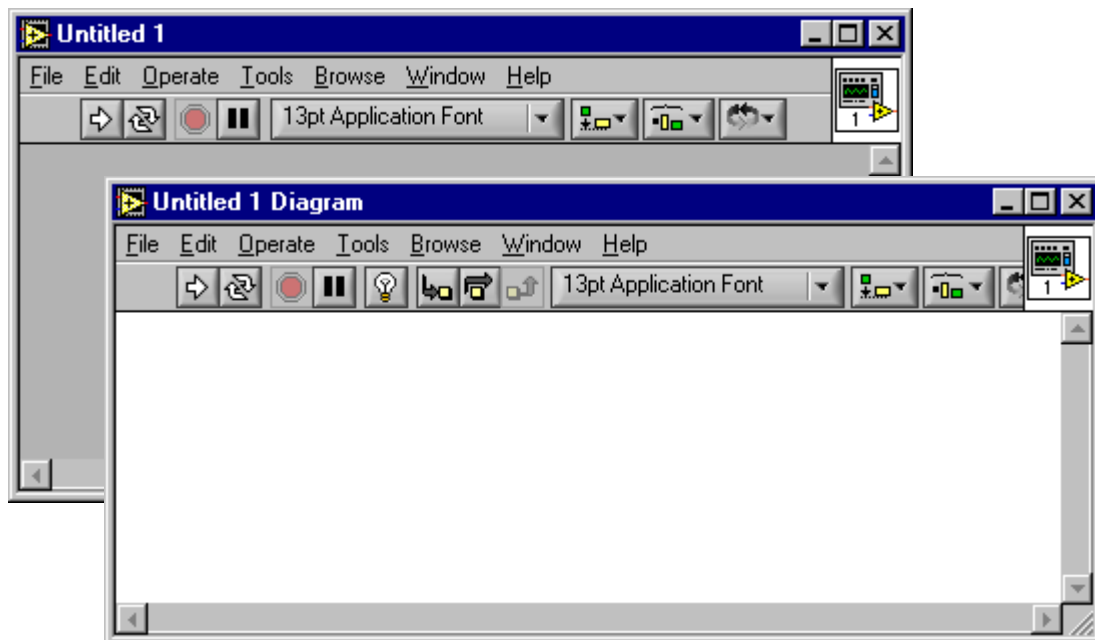
Channel n

```
>ModBus>[::]!!I:I:-1:-1:-1:-1:Wago Digital Output n[Coupler1.type=b.addr=n-1.len=1]
```

5. Access Library-Functions from LabMap[®] in LabView[®]

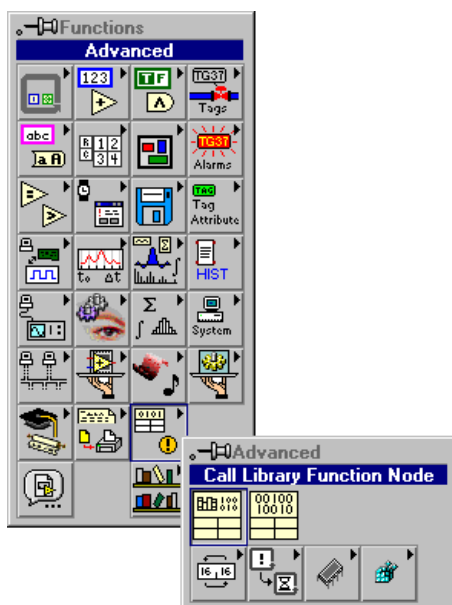
Requirements: LabMap[®] and LabView[®] are installed.

1. Start LabView[®]



You see an empty Virtual-Interface and the appending Diagram.

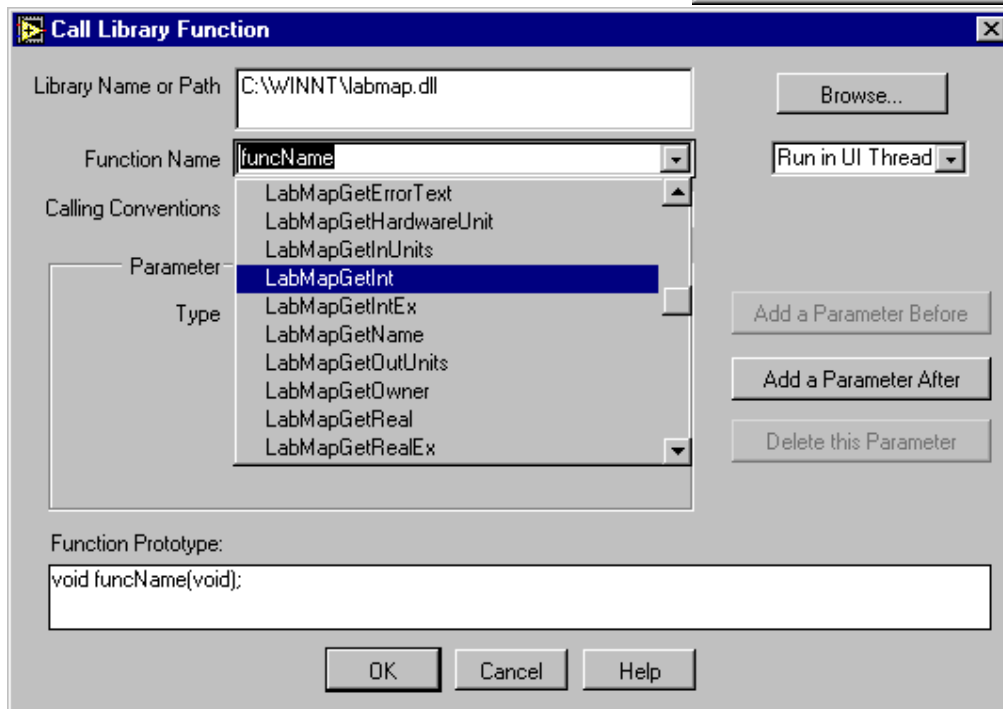
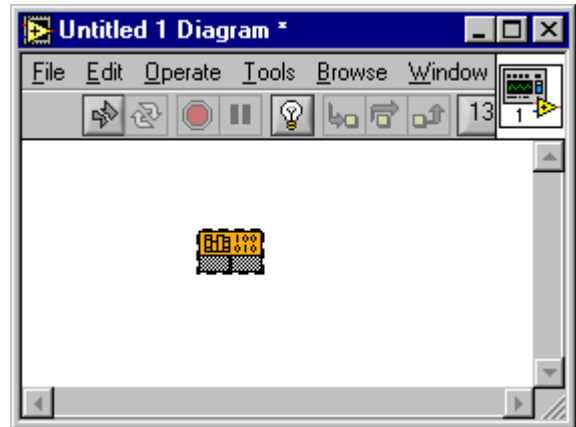
2. Select in LabView[®] the *Call Library Node*



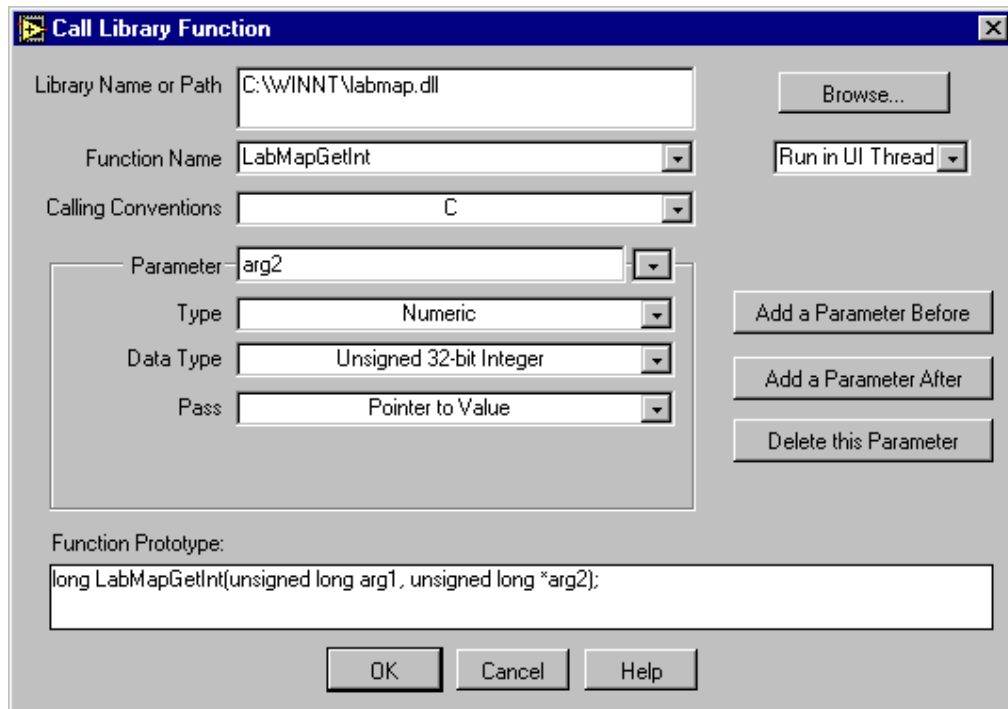
Right Click with the mouse in the empty Diagram and select under *Advanced-Functions* the *Call Library Function Node*.

3. Select the **LabMap**[®]-Function.

Make a double Click on the *Call Library Function Node Symbol* in the Diagram.



Browse to the labmap.dll. Normally these DLL is located in the folder C:\Winnt . Select the Function name.



Insert appropriated Parameters for the selected function.

Close the *Call Library Function Node* Dialog with *OK*.

You can find information about the required parameters in the [LabMap[®]-Handbook](#) under the topic *Programmers-Interface*.

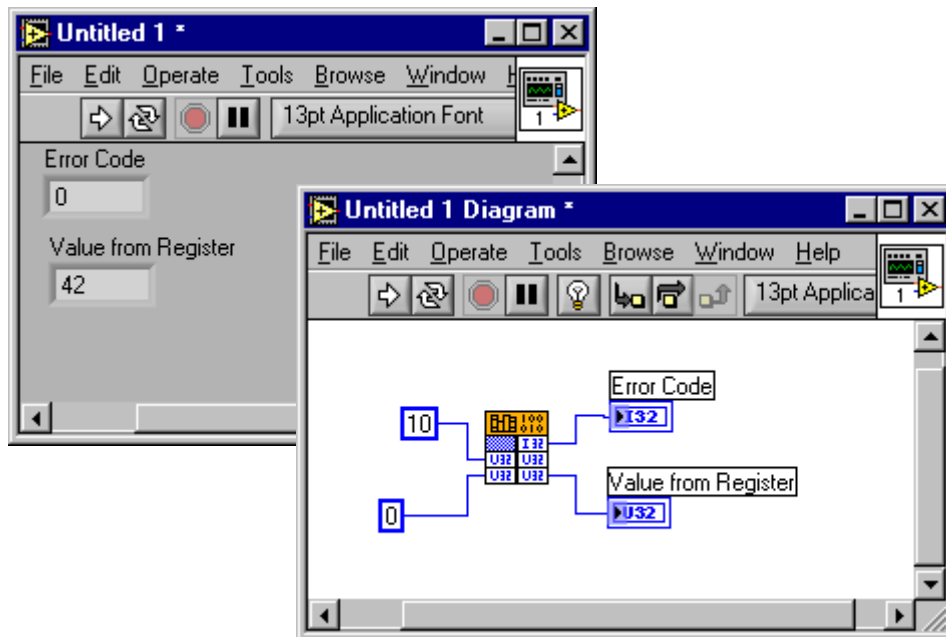
For the selected Function `LabMapGetInt()` you can find the following Information in the [LabMap[®]-Handbook](#):

4.19. LabMapGetInt

```
int LabMapGetInt
(
    unsigned HandleNo,
    int      * Value
);
```

These functions get the current value of the integer register specified by the parameter **HandleNo**. It does not start any operation on the register, just the last value is returned (see also [LabRRRRequest](#)). Values of both input and output registers may be requested.

4. Connect the Input- and Output Parameters to Indicators and Controls/Constants and run the example.



The Parameter for the Handle Number has the value 10. The constant 0 is a placeholder for the Indicator *Value from Register* because we use here *Pointer to Value* as Pass. *Error Code* indicates if the function returns with or without errors. (LabMap-Errorcodes are described in the [LabMap[®]](#)-Handbook)

6. LabMap® Interface for WinFACT®/Boris

6.1 Installation of the LabMap®/WinFACT® Interface DLL's

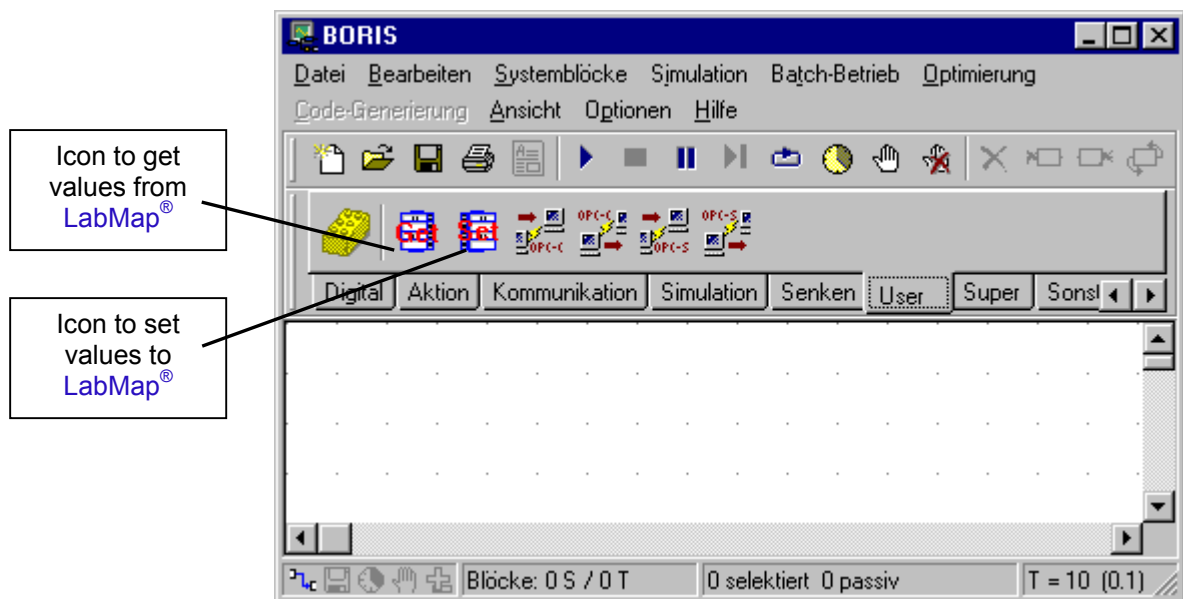
Copy these files:

- Lm_Get.dll
- Lm_Send.dll
- Lm_Get_p.bmp
- Lm_Get_t.bmp
- Lm_Send_p.bmp
- Lm_Send_t.bmp

Into the program location from WinFACT® in the folder /UserDlls

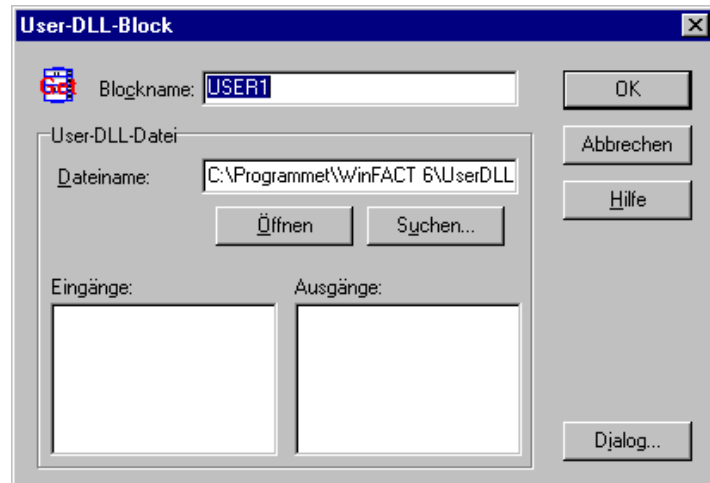
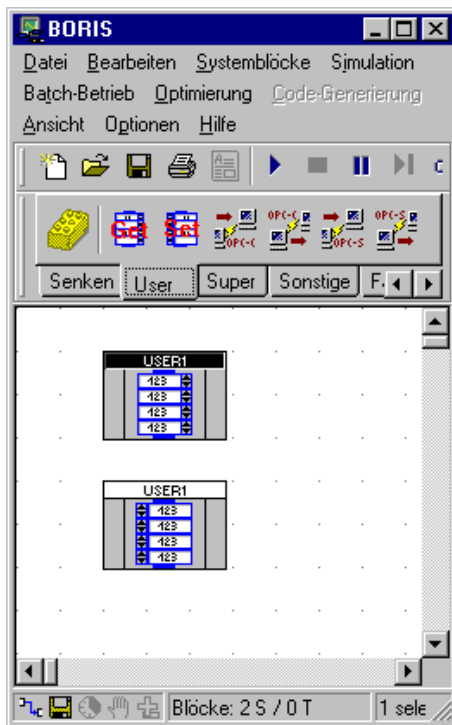
6.2 Using the LabMap® / WinFACT® Interface

Start WinFACT®/Boris.

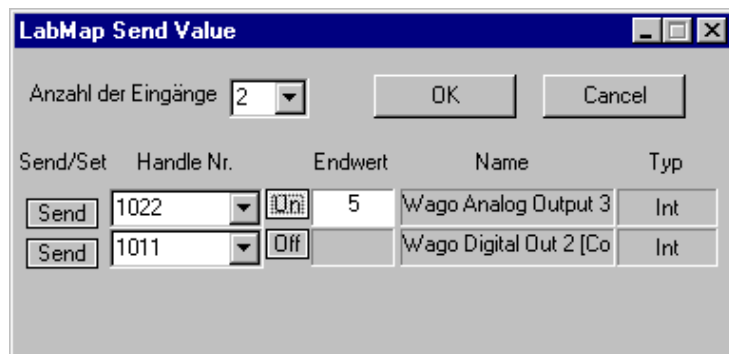
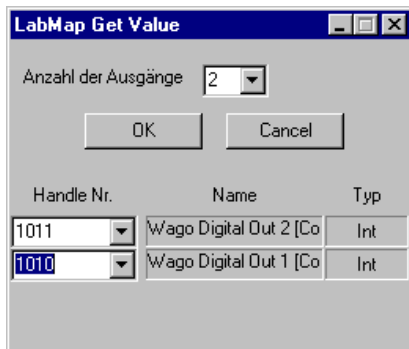


A mouse click on the icon *Get* insert the block *Get Value* and mouse click on the icon *Set* insert the block *Set Value* into the sheet.

A double click of a block opens the *User-Dll-Block* dialog.



The Dialog... button pops up the property dialog for the selected interface block.



Select the appropriated registers and properties and click OK.

7. Examples

In this section I will explain some examples to show special features of the software bus LabMap®.

7.1 Virtual Channels to limit input values

This configuration shows a way in which virtual channels can be used to limit an input range from +/- 10V of a Wago 2-Channel Analog Output Module (Item No. 750-556).

We need three Handles with the following properties.

1. A User Handle as Input register.

Hdl. No. 1000 >User>[::]!R:-1:0:592:127: User Input Handle

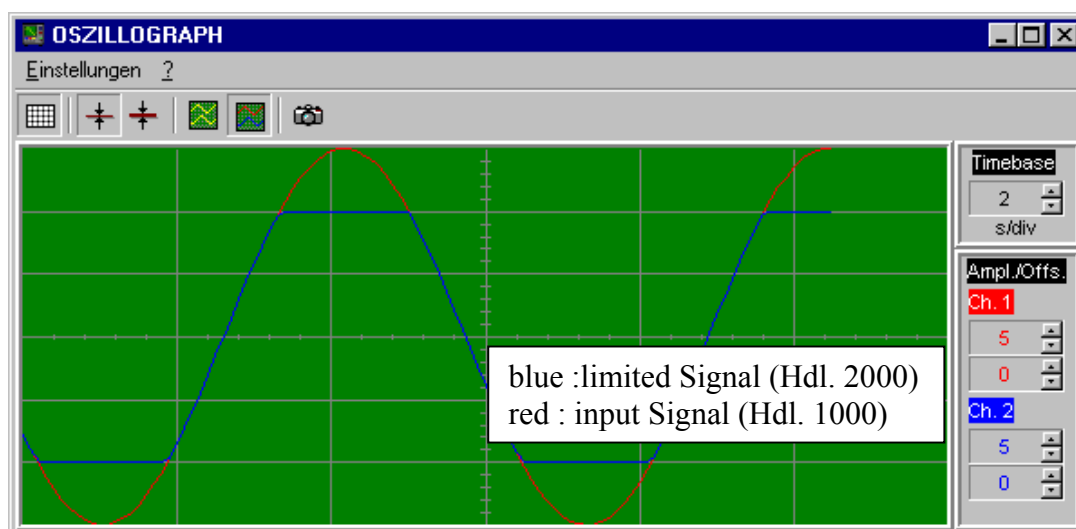
2. The virtual handle

Hdl. No. 2003 >Virt>[::]!O:R:0:-50:592:248:Virtual [-10 max \$1000 min 10 -> 2000]

3. The limited ModBus handle

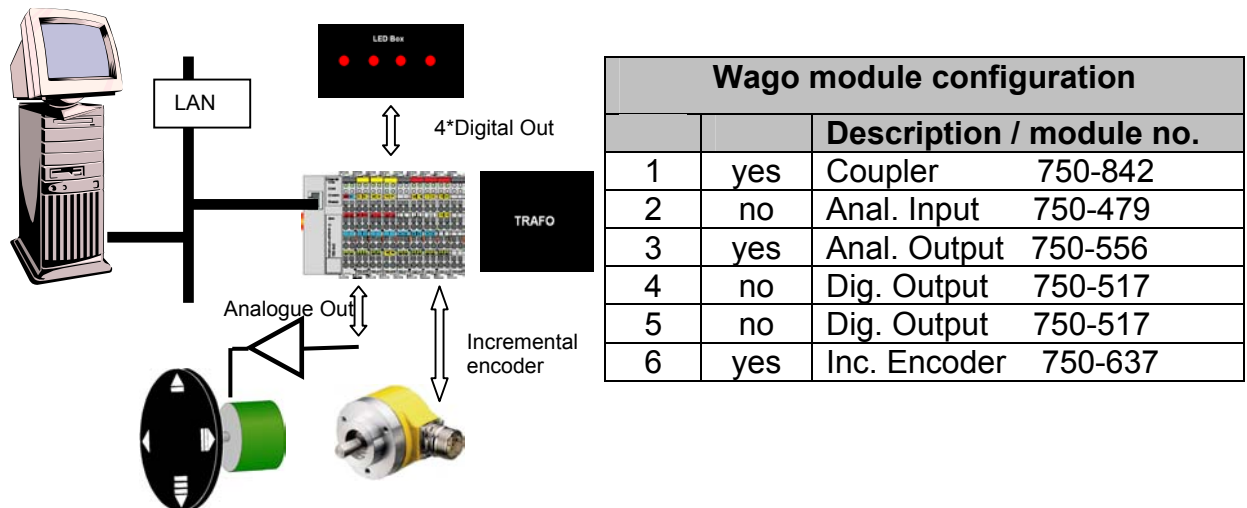
Hdl. No. 2000 >ModBus>[V:V*3.0518509476e-4:V]!R:-1:0:590:371:
 Calculated Analog Output 1
 [Coupler1.type=i.addr=1.len=16]

The Boris Simulation shows the limited Signal.



7.2 Feedback System with Boris, Wago modules and LabMap®

The following example shows a feedback system to control motor rotation speed. Here is the hardware system layout for the example.

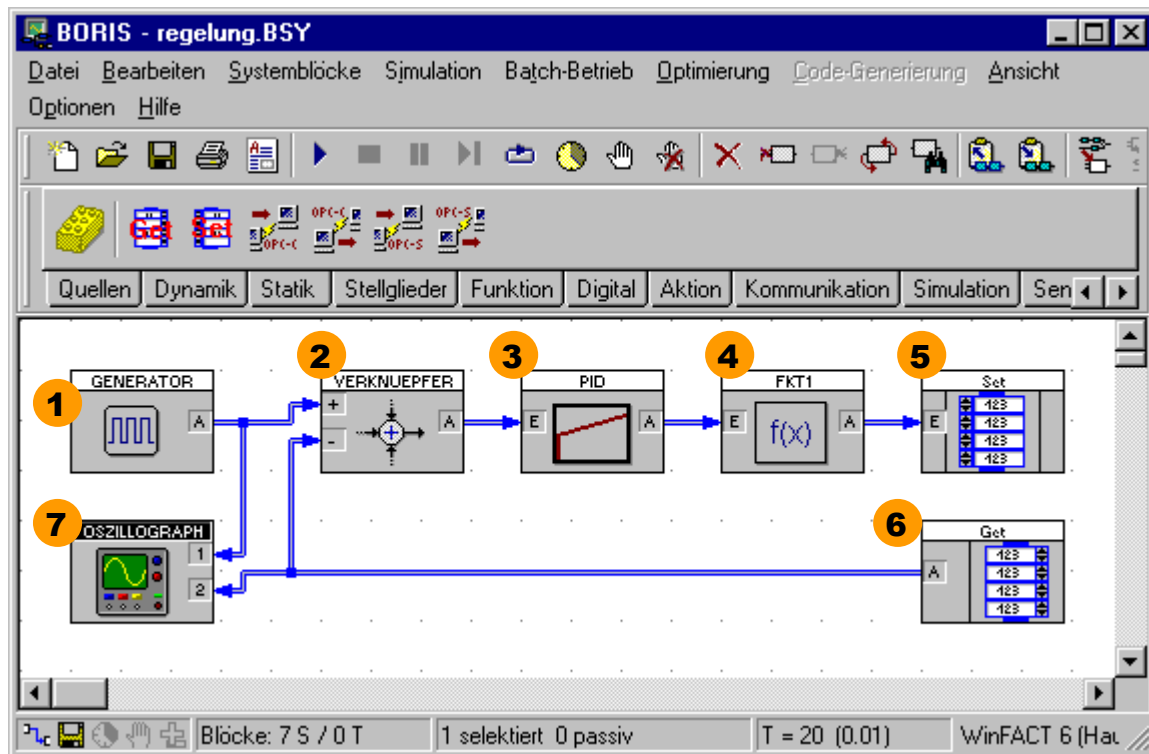


On the computer runs **WinFACT®/Boris** and **LabMap®**. Boris contains the software model for the feedback system. **LabMap®** connects the Wago I/O modules over Ethernet with the feedback simulation model. The motor gets the controller output from a Wago analogue output module transferred over an amplifier. An incremental encoder puts the rotation speed in a Wago analogue input module.

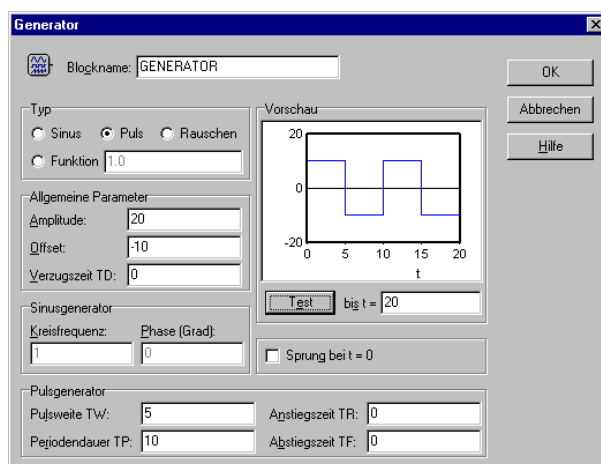
The **WinFACT®/Boris** feedback simulation model gets and sends the values for the actual speed and the desired speed from the softwarebus **LabMap®** about specific Boris/**LabMap®** interfaces.

7.2.1 WinFACT[®]/Boris feedback model

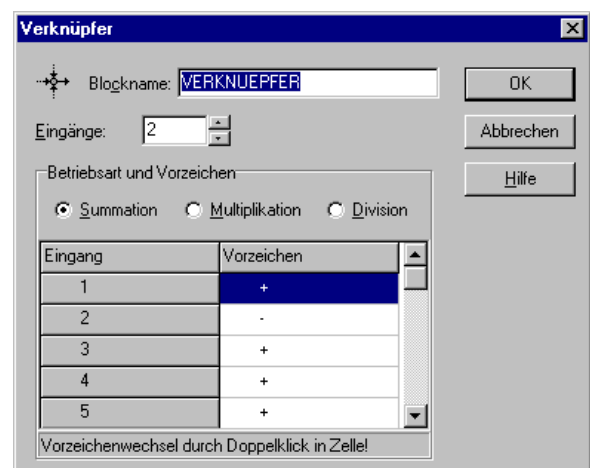
The following diagram shows the WinFACT[®]/Boris feedback simulation model and the parameter dialogs for the used blocks:



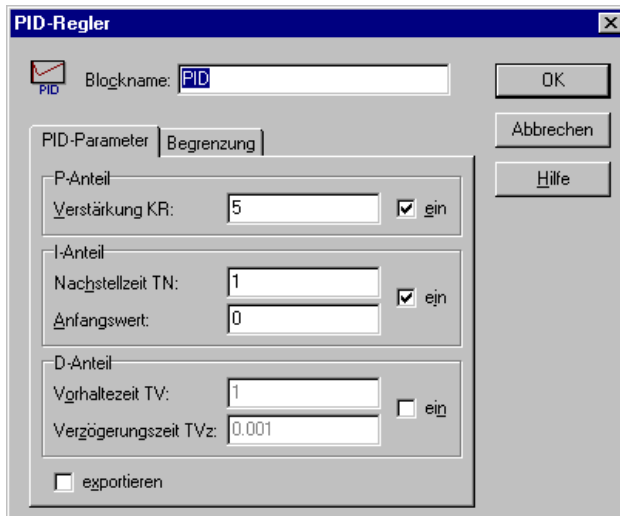
1) Signal-Generator



2) Linker



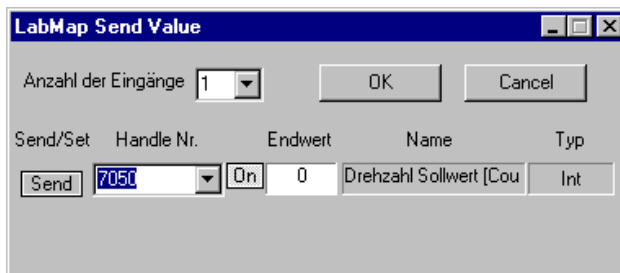
3) Controller



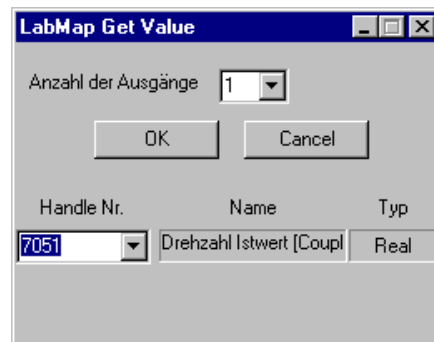
4) Function



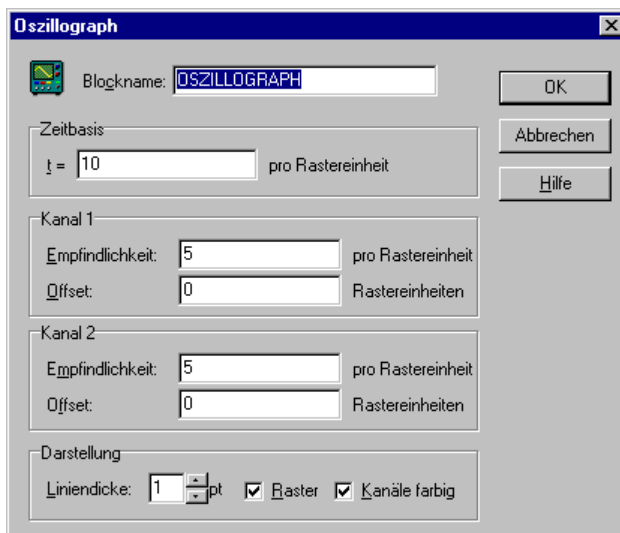
5) Boris/LabMap®-Send interface



6) Boris/LabMap®-Get interface

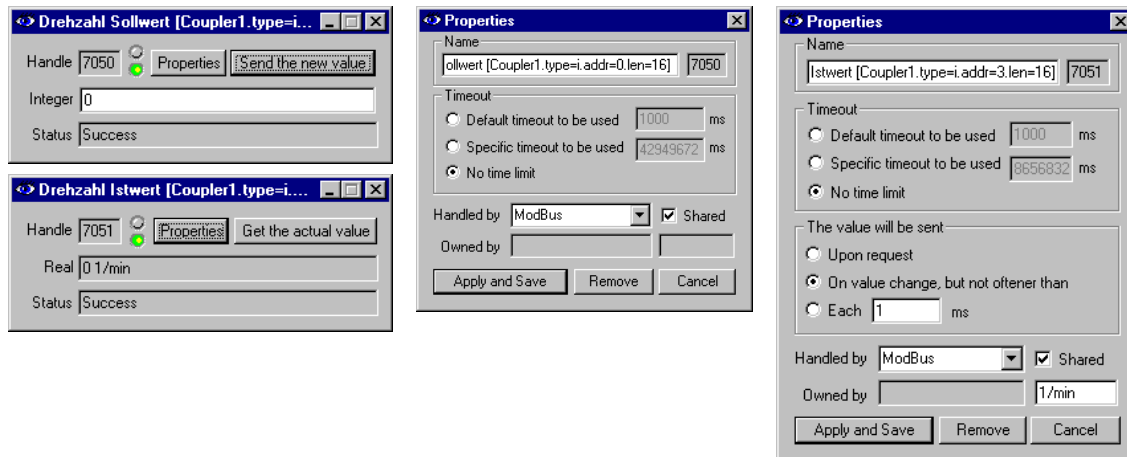


7) Oscilloscope



7.2.2 LabMap[®] register configuration:

Here you see the LabMap[®] register user interface and the appendant properties dialog for the configured registers. The first one is the controller output (register 7050) and the second one is the actual speed (register 7051).

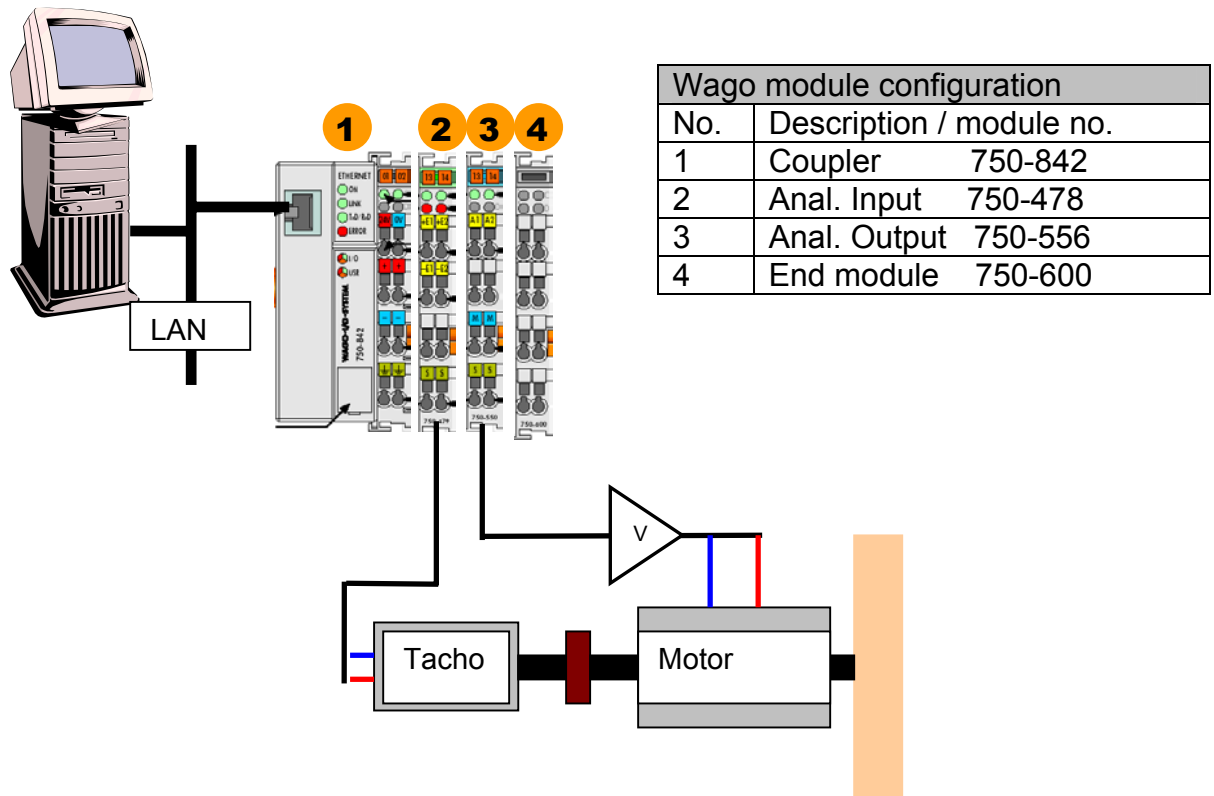


Register Number	Registration string	Description
7000	>ModBus>[::] !!:!:~1:0:-1:-1: Encoder Control C1 und C2 [Coupler1.type=w.addr=6.len=16]	ModBus input register to set speed measure state for the incremental encoder. Send the value 2 to the Wago output module on output channel 6 before starting simulation
7050	>ModBus>[::] !!:!:~1:0:-1:-1: Drehzahl Sollwert [Coupler1.type=i.addr=0.len=16]	ModBus input register to set the voltage (+/- 10V) to the Wago analogue output module on output channel 0 -> motor voltage
7051	>ModBus> [1/s:0.0694444444*1/s:1/min] !O:R:-1:1:-1:-1: Drehzahl Istwert [Coupler1.type=i.addr=3.len=16]	ModBus output register that shows the actual speed generated by the incremental encoder when working in speed measure state. This is the output channel 3

7.3 Filtering a signal with Virtual-Registers

This example shows a filter realisation with virtual registers to reduce the noise in a signal from a tacho-machine.

Hardware configuration:



On the computer runs [LabMap®](#) with configured registers for a Wago analogue output module and an analogue input module to set a defined voltage over the amplifier to the motor and get the voltage generated by the tacho machine.

Supplemental configured register is a virtual register that works as filter for a user register that contains the filter time constant and a user register to output the filtered signal.

7.3.1 LabMap[®] register configuration:

Register Number	Registration string		Description
10	>ModBus> [V:V*3.0518509476e-4:V] !!:R:-1:0:-1:-1: Analog Output 1 [Coupler1.type=i.addr=0.len=16]		ModBus input register to set a defined voltage (+/- 10V) to the Wago analogue output module on output channel 0 -> motor voltage
20	>ModBus> [V:V*0.0003052:V] !O:R:-1:5:-1:-1: Analog Input 1 [Coupler1.type=int.addr=0.len=16]	U_k	ModBus output register to get the noised voltage from the Wago analogue input module on input channel 0 <- tachometer voltage
30	>User> [s:s:ms] !!:R:0:0:-1:-1: T-constant	T	User register for the filter time constant. The time constant must be set before testing the filter. The time constant can be modified at runtime.
40	>Virt> [::] !O:!: -1:-1:-1:-1: Filter [[(\$20*dT+\$50*\$30)/(dT+\$30)]->50]		Virtual output register with filter algorithmic. The calculated value is sent to the result register number 50
50	>User> [V:V:V] !!:R:0:0:-1:-1: Result	Y_{k-1}	User register that shows the filtered voltage signal. Before testing the filter the register must be set with a defined value (e.g. 0)

The used filter is a Low pass first order and has a PT1 characteristic.

The digital algorithm is: $Y_k = q_0 * U_k + p_1 * Y_{k-1}$

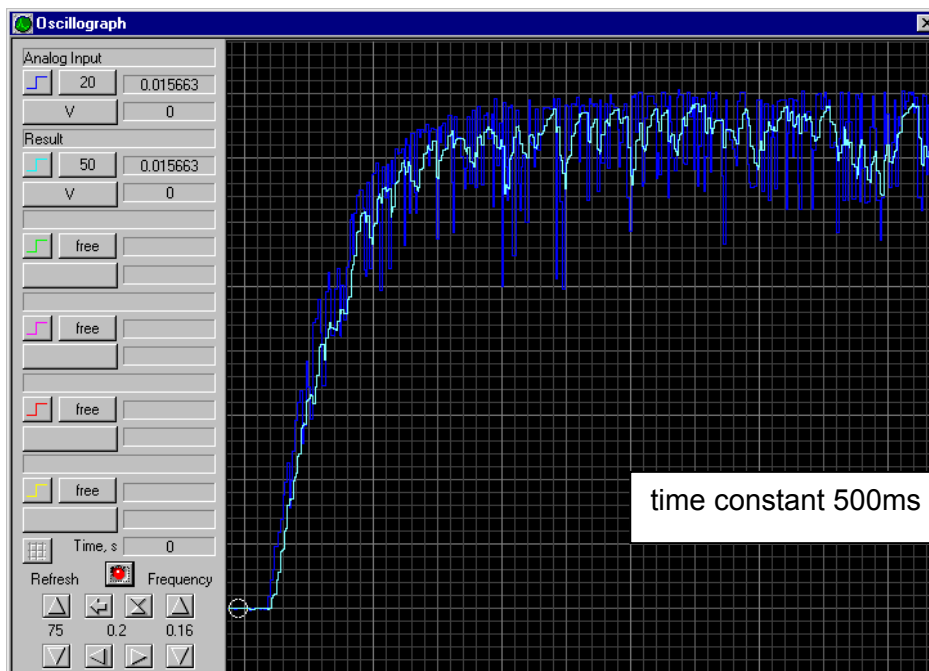
7.3.2 Using the filter

To use the filter send "0" to the result register 50 to initialize the feedback filter algorithm. Then send a value for the time constant to the register 30.

To start now a filter test you can send for example a jump from 0V to 5V with the register user interface to the register number 10 and monitor the results with the LabMap[®] tool "Oscilloscope"

7.3.3 Example filter test

The following pictures are monitored with the LabMap[®] tool “Oscillograph”. They show the voltage from the tacho machine (register 20) and the filtered voltage (register 50) at a jump from 0V to 5V on register 10 with different time constants represented by the register 30.



7.4 Monitoring data from LabMap[®] with LabView[®]

In this example I will show how you can access data from the software bus LabMap[®] in an easier way than shown under the topic 5. .





We use the same hardware configuration as in section 7.2. . The goal of this example is to monitor the voltages from the tacho machine and the filtered signal as shown in section 7.3 now with LabView[®] .

Requirements: LabMap[®] and LabView[®] are installed and following files are available:

- *LabLabView.dll* : wrapper dll for LabMap[®]
- *LMGetReal.vi* : LabView[®] VI to get real values from LabMap[®]
- *LMGetInt.vi* : LabView[®] VI to get integer values from LabMap[®]
- *LMSendReal.vi* : LabView[®] VI to send real values to LabMap[®]
- *LMSendInt.vi* : LabView[®] VI to send integer values to LabMap[®]

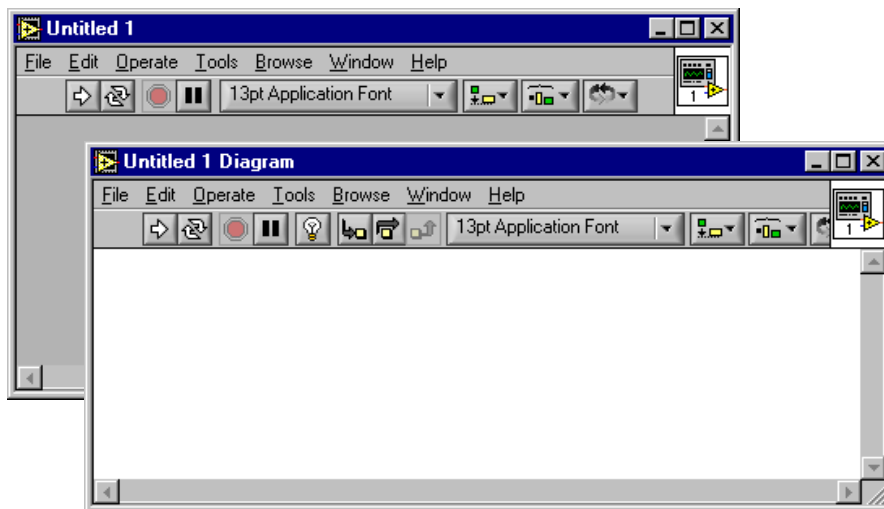
The example vi is *LM_LabView_test.vi*.

The *vi files* gives you an easier access to LabMap[®] because they show you symbolic which LabMap[®]-function is implemented by the selected block.

Symbol	Function
	Get the integer value associated with handle number
	Get the real value associated with handle number
	Send the integer value to the specified handle number
	Send the real value to the specified handle number

7.4.1 Getting access to LabMap[®] with the LabView[®] VI's

1.) Start LabView[®]



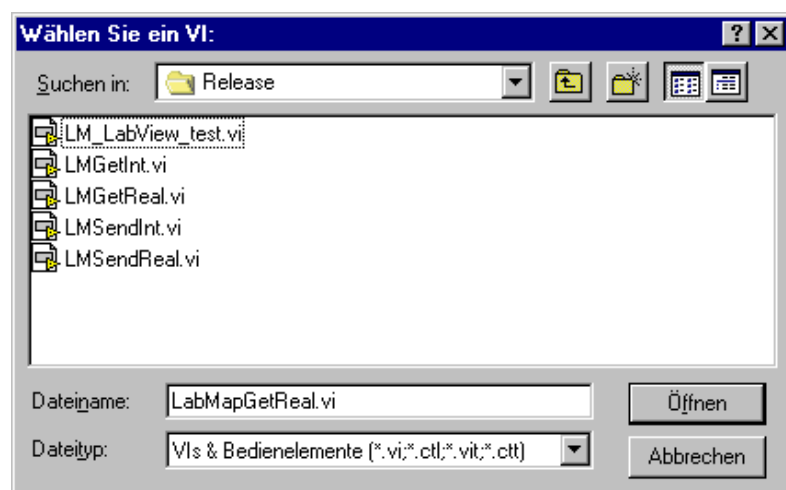
You see an empty Virtual-Interface and the appending Diagram.

2.) Select in LabView[®] the *Call Library Node*

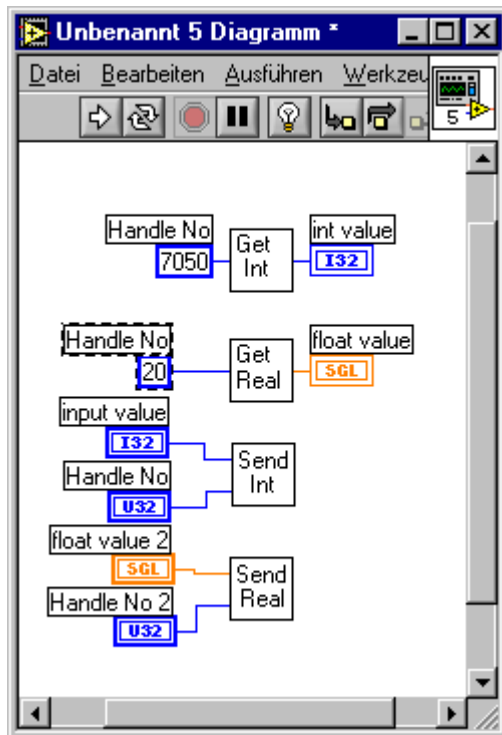
Right Click with the mouse in the empty Diagram and select under Functions the *Select a VI Function Node*.



Browse to the *LM vi files* and select this one you want.

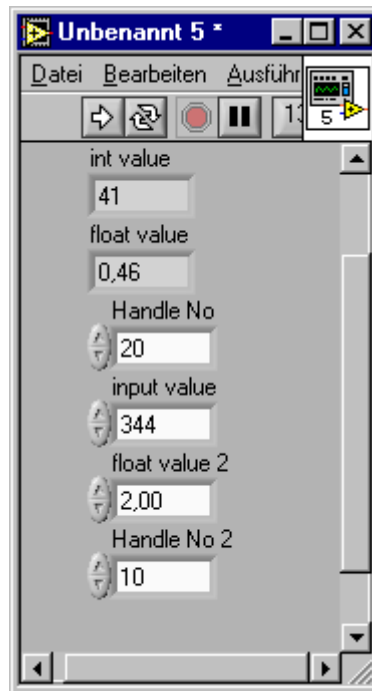


3.) Place the selected vi into the diagram



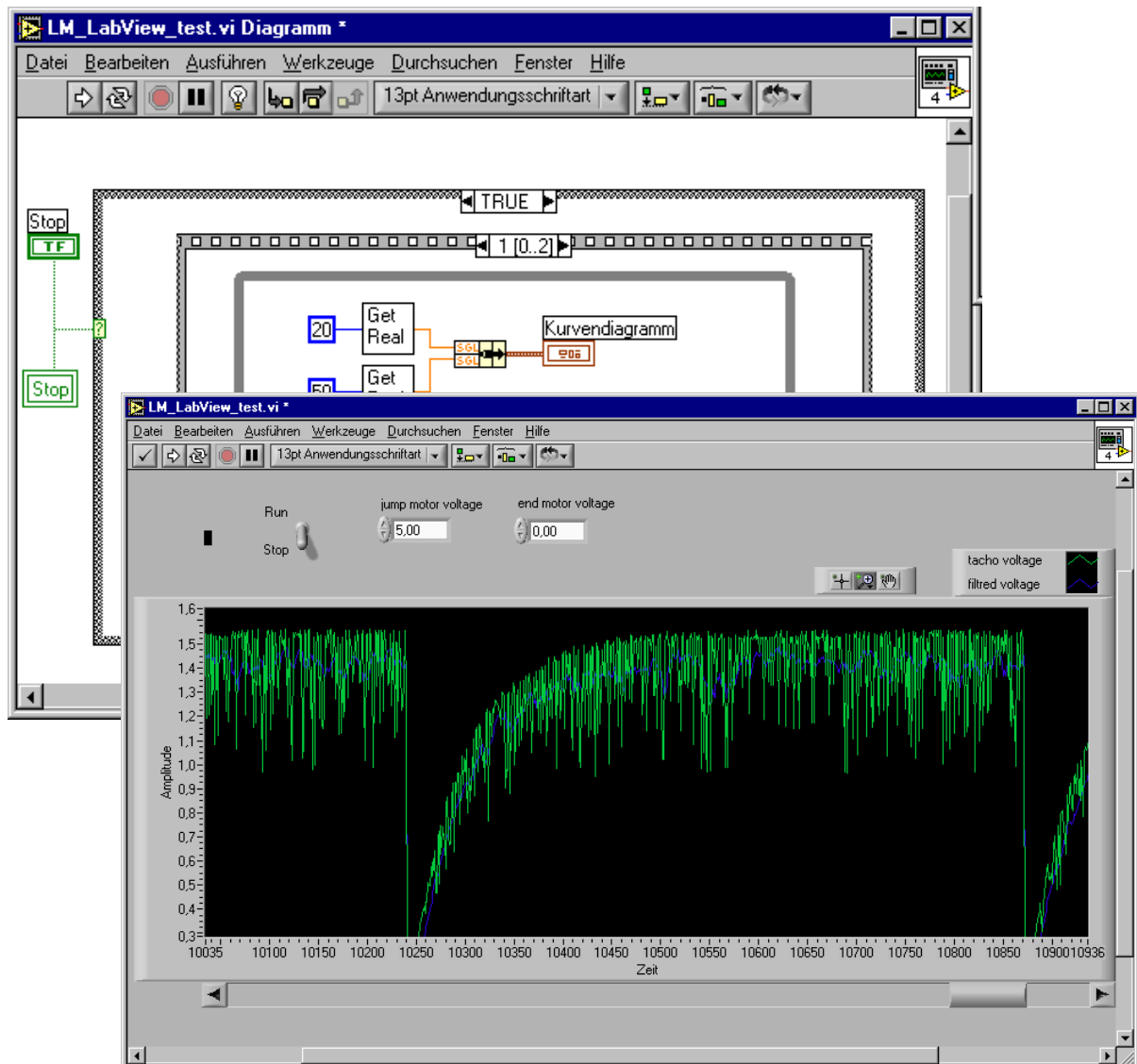
Connect the inputs and outputs.

Now you have read and write access to LabMap registers.



7.4.2 Using the LM_LabView_test.vi

Start LabView® and open the LM_LabView_test.vi and run the example.



Here you can see the same data in LabView® as under section 7.3.3 with the LabMap® tool Oscillograph.

7.5 Monitoring data from LabMap[®] with Matlab[®]

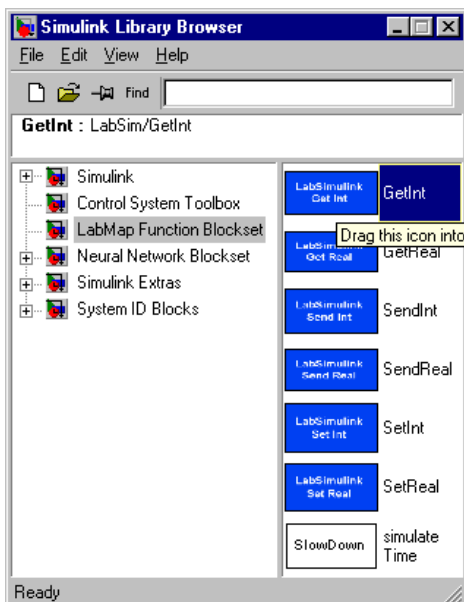
This example is similar to example 7.4 (same hardware configuration and same LabMap[®] configuration). The different is that we use Matlab[®] to access the data from LabMap[®].

Requirements: LabMap[®] and Matlab[®] are installed and following files are available and located *under root\MatLab\toolbox\LabMap*:

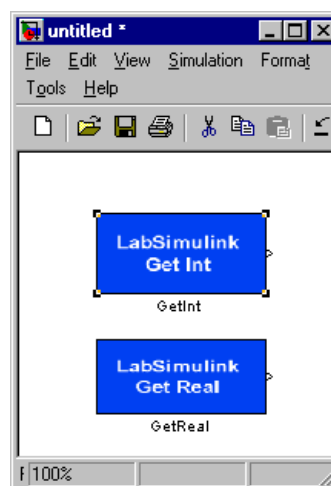
- *setpara.m* : m-file to set block parameter
- *siblocks.m* : m-file for simulink library display
- *UpdateParams.m* : m-file to update block parameters
- *LabMapReg.dll* : dll for block dialog (register selection)
- *LabSimulink.dll* : dll to access LabMap functions (GetInt, ...)
- *SlowDown.dll* : dll to reduce Matlab calculation speed
- *GetInt.tif* : picture for GetInt block
- *GetReal.tif* : picture for Getreal block
- *SendInt.tif* : picture for SendInt block
- *SendReal.tif* : picture for SendReal block
- *SetInt.tif* : picture for SetInt block
- *SetReal.tif* : picture for SetReal block

The example Matlab/Simulink[®] model *LM_Matlab_test*.

When all files are available under the above-specified folder and you start Matlab/Simulink[®] you can find the blocks to have read and write access to the software bus LabMap[®] in the Simulink[®] Library Browser.



You can now drag and drop these blocks in your model sheet and build your own simulation model.



7.5.1 Working with LabMap®/Matlab® blocks

When you have to insert a LabMap® block in your model sheet you must select the register, which the block should use.



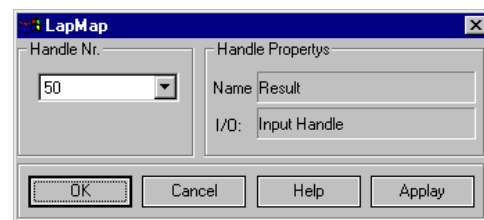
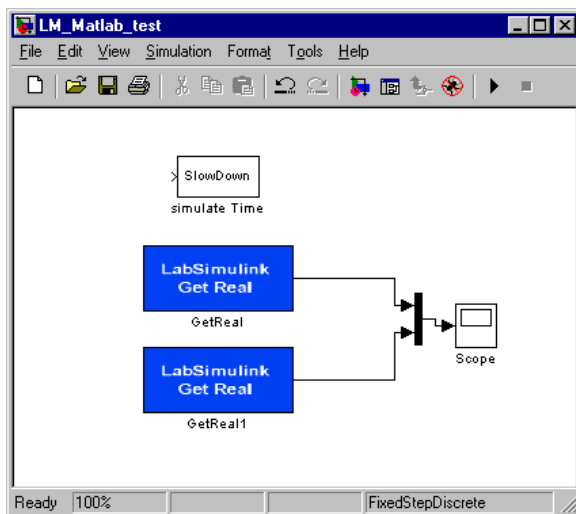
To do this double click on the block. The block property dialog pops up.

Select the register and press OK.

The properties dialog shows the selected register name and the I/O direction.

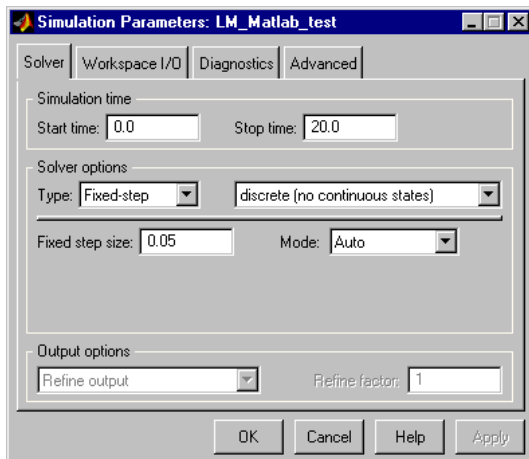
7.5.2 Using the LM_Matlab_test.mdl

Start Matlab/Simulink® and open the model file LM_LabView_test.mdl.



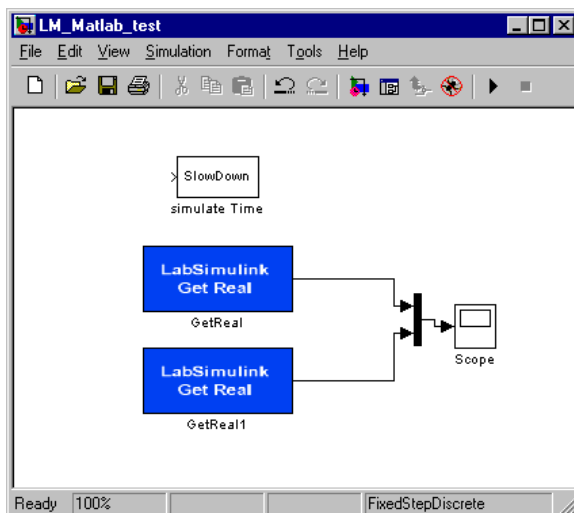
Check the select register and close the properties dialog with OK.

Check under *Simulation->Simulation Parameters ...* the following values:



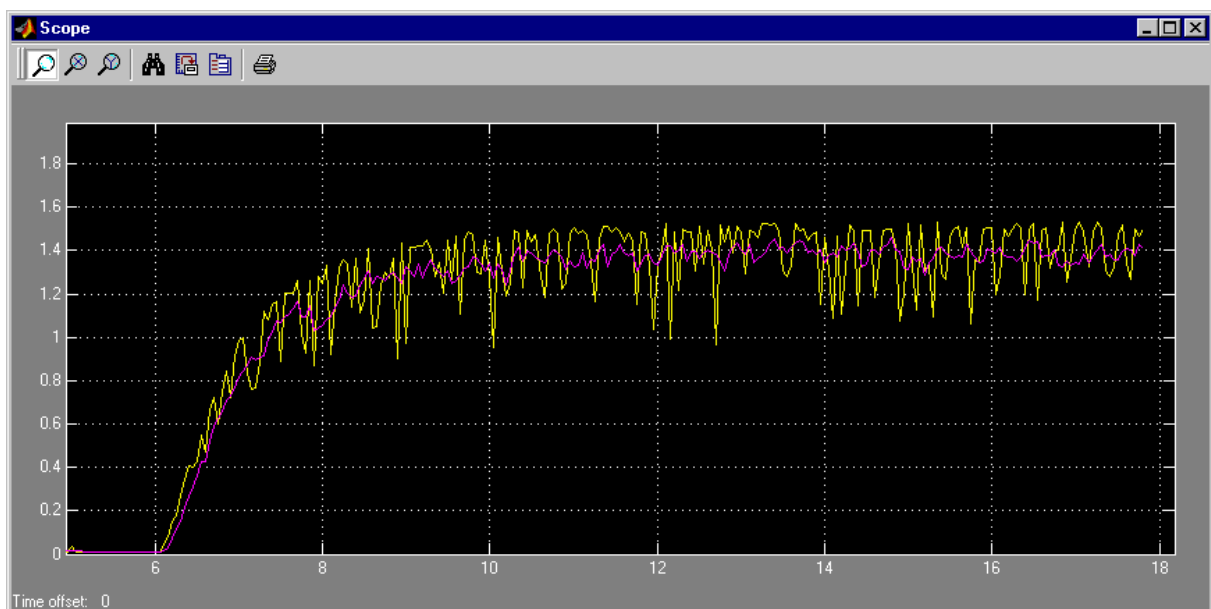
It is important that the *Solver options Type* is set to *Fixed-step* and the *Fixed step size* is set to a discrete time.

Now you can start the simulation with the play button.

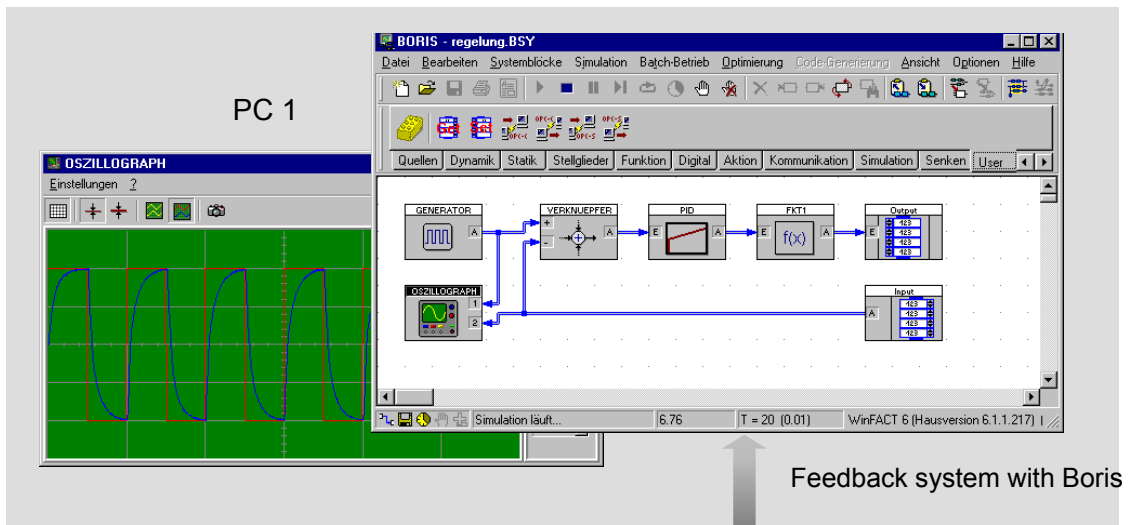


With this simple example model you can see the two values from the [LabMap®](#) registers in the Matlab® scope.

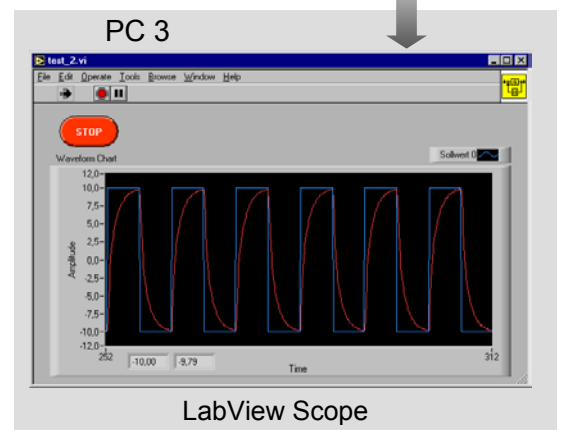
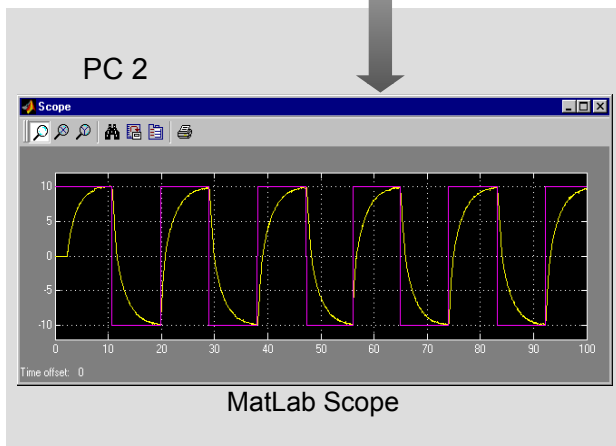
The SlowDown block in the model slows down the calculation speed from Matlab® to simulate "real time".



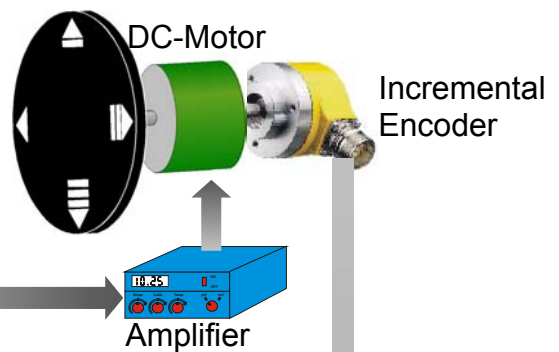
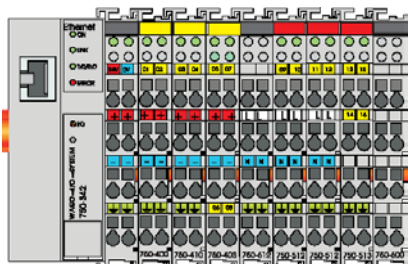
8. Distributing data over the network with LabMap[®]



LabMap[®]



Labortechnik
Laborautomation



To distribute data with LabMap® over Network a client and server pc must be configured.

The remote access interface is provided by the LabNet.dll.

- *Server configuration.* The server part is responsible to accept connections from clients. It is controlled by the registry keys MaxConnections, HostsAllow, HostsDeny and Port. MaxConnections limits the number of simultaneously connected clients. The server part is not started when MaxConnections is 0. Note that this is the default. HostsAllow and HostsDeny are used to prevent unsanctioned access. The key Port specifies the TCP/IP port used to accept the connections from the clients.
- *Client configuration.* The client part may maintain more than one connection to remote servers. It is also possible to establish several connections to the same server. Each connection to a remote server is indicated by the server's **nickname**. The nickname is a case sensitive text string indicating the remote server. For each nickname (server) the following registry keys should be defined: Server.<Host>, Port.<Host>, where <Host> indicates the nickname of the remote server. The key Server.<Host> contains the address of the remote server to connect to. The key Port.<Host> specifies the TCP/IP port used for the connection.
- *Remote register configuration.* Remote registers are handled by the LabNet.dll (>Net>). The request policy of a remote register has the following meaning. **Upon request** and input registers are updated from the server side. The frequency at which the updates are sent to the client is limited by the RemotePollLimit registry key. **On change** registers are updated not more often than specified. **Each** registers are updated as specified no matter whether their state is changed.

The name of a remote register name may end with additional properties specification put in square brackets. For instance, register 2810 could be defined as:

>Net>[m/s:m/s:km/h]O:R:0:100:-1:-1:Speed [2810.RRR]

Here the additional properties specification is [2810.RRR]. It has the following format:

[<Register>.]<Host>

<Register> specifies the register number on the server side. This allows to map a remote register to any desired register of the remote server. In the given example both registers on the client and server sides have same number, so '2810.' could be omitted. <Host> defines the nickname of the remote server. In our example it is 'RRR'. When this part is missing the nickname is assumed to be empty. Each unique nickname indicates a separate connection to some remote server. For each such name a virtual input string register named 'Host [<Host>]' shall be declared (host name register). In our example it should have the name Host [RRR]. This register is initialized with the contents of the registry key Server.<Host>. It always contains the name of a remote server. A send operation over this register causes reconnection to the designated remote server. An off-line error on the register indicates that the remote server is currently off-line. Note that there is no need to manually reconnect to the remote server on an error, because LabNet.dll does it automatically. So the connection is restored as soon as possible. The remote host registers can be browsed from the client side using the host number register (see).

8.1 Example Distribute data over the network

This example shows the client and server configuration for the example in 7.3 “Filtering a signal with Virtual-Registers” to distribute the values on register 50 (filtered voltage) and the register 20 (unfiltered voltage).

The following registry keys are required

Server configuration:

"Net"="LabNet.dll" : network interface
 "Port"=dword:00000420 : port number
 "HostsDenied"="" : no IP denied
 "HostsAllow"="" : all IP's allowed
 "MaxConnections"=dword:00000010 : max. connections = 10

Client configuration:

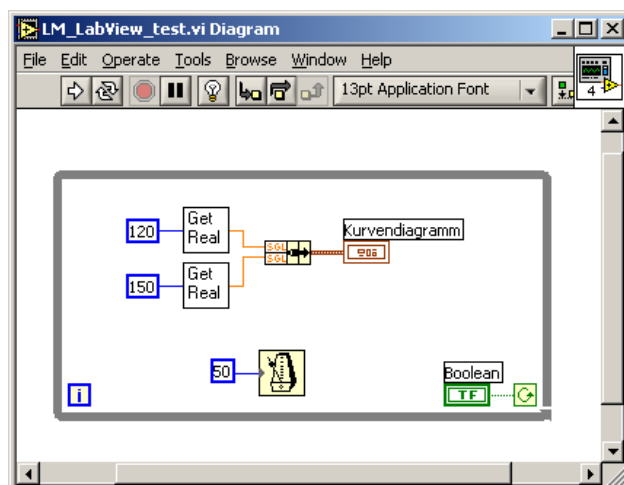
"Net"="LabNet.dll" : network interface
 "RemotePollLimit"=dword:00000064 : refresh rate (100ms)
 "Server.hongshoui"="192.168.192.33 (sync)" : Server IP
 "Port.hongshoui"=dword:00000420 : Server Port number

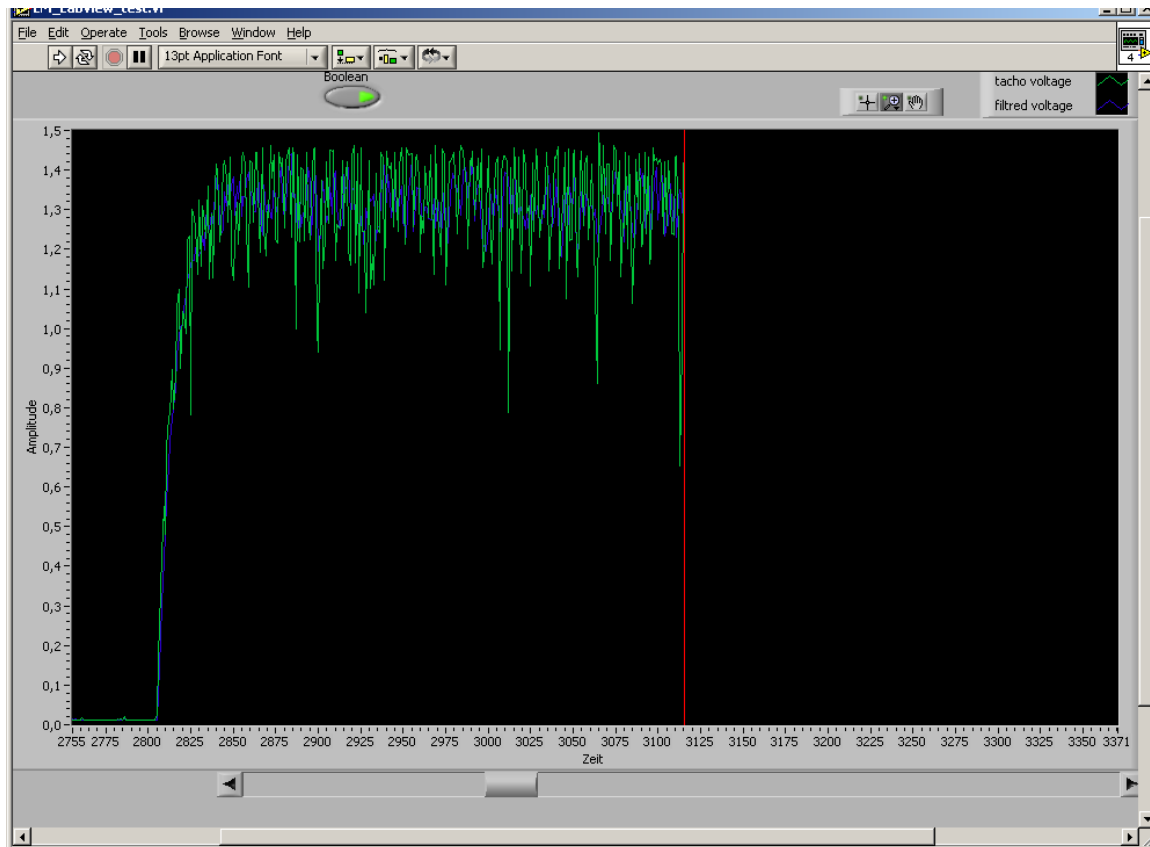
Client register handles:

"1000"=">Net>[::]!S:0:0:229:530:Host[hongshoui]"
 "120"=">Net>[V:V:V]!O:R:0:-1:381:146:Analog Input 1[20.hongshoui]"
 "130"=">Net>[ms:ms:ms]!R:0:0:531:522:T-constant [30.hongshoui]"
 "150"=">Net>[V:V:V]!R:0:0:657:262:Result[50.hongshoui]"

With this configuration you have access from a client PC to the register 20, 30 and 50 on the server. The server register values are here mapped to the client PC register 110, 120, 150.

We can now run for example LabView® on the client PC to show the signals distributed by the server PC.

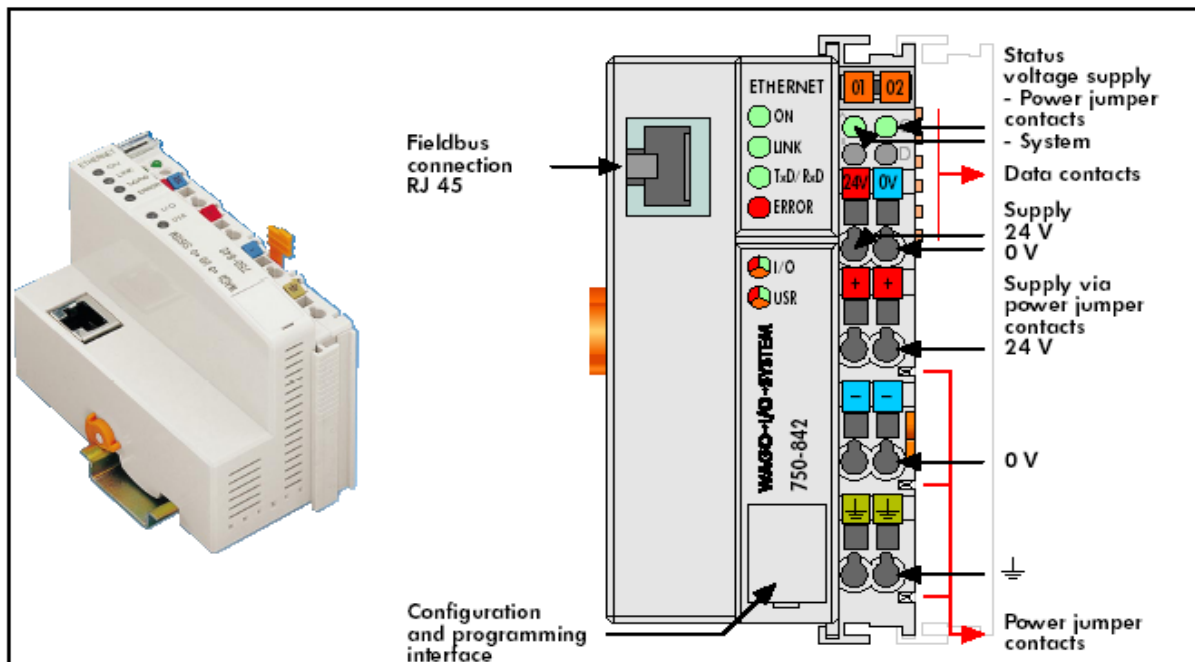




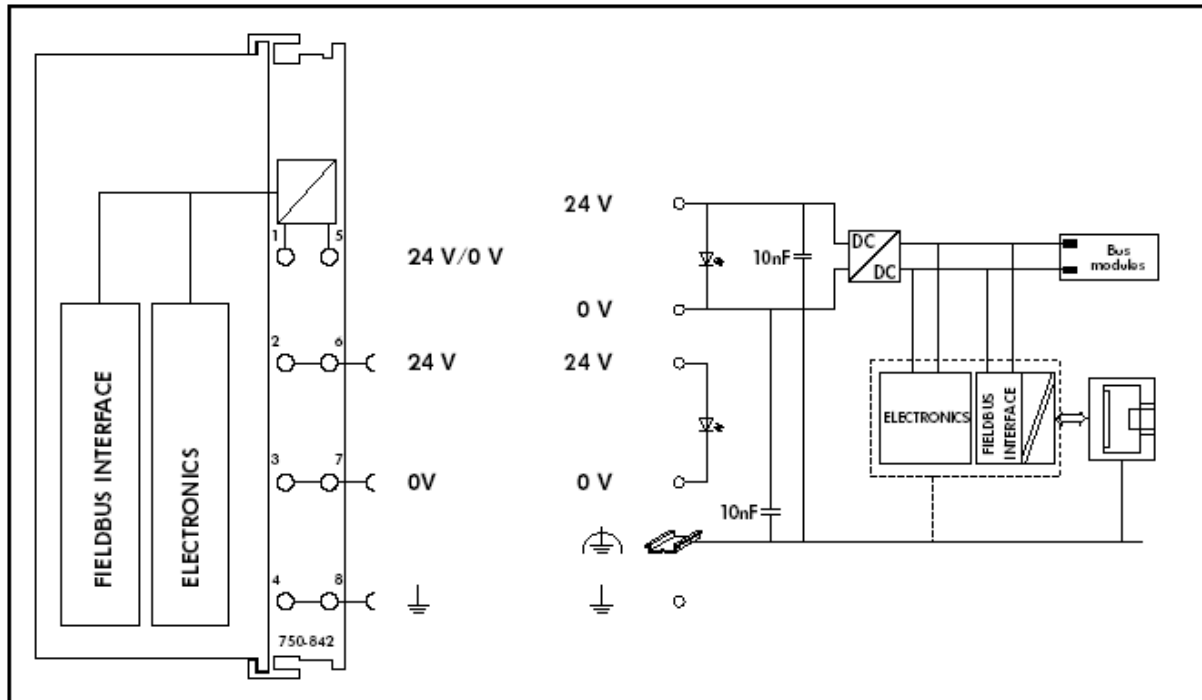
Appendix: Wago Modul description

ETHERNET TCP/IP

Programmable Fieldbus Controller; 10 Mbits/s; digital and analog signals



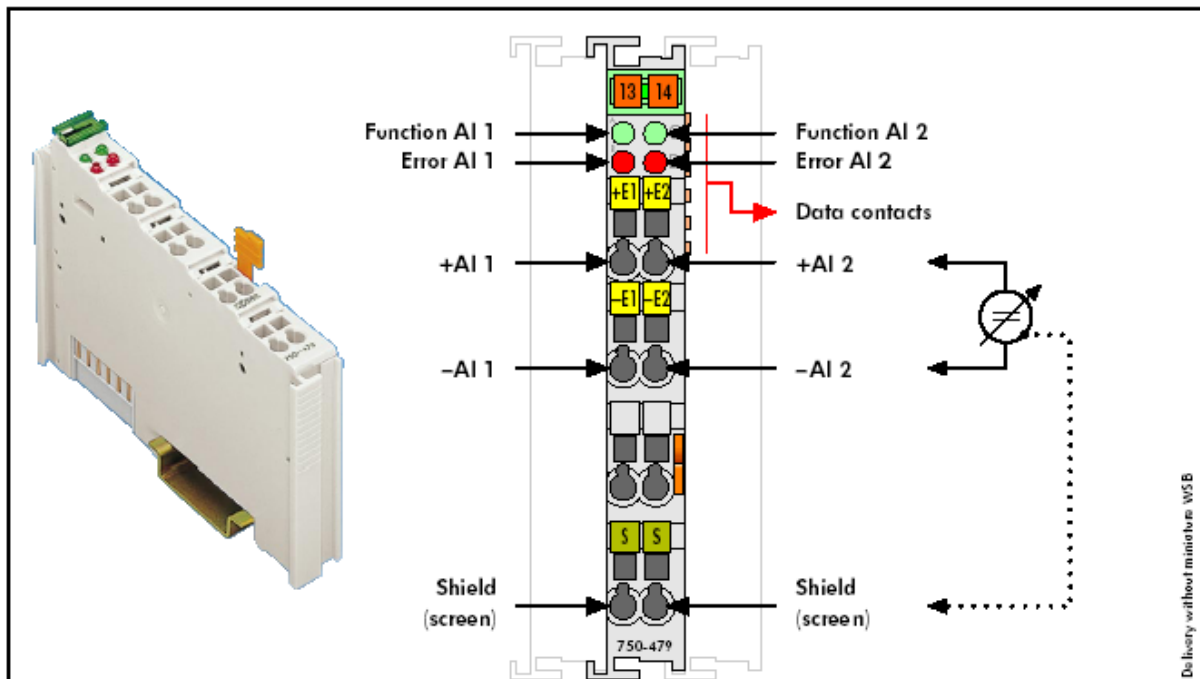
Description	Item-No.	Pack.-unit pcs																								
Contr. ETHERNET TCP/IP 10MBit	750-842	1																								
<p>The programmable fieldbus controller for ETHERNET combines the functionality of the ETHERNET fieldbus coupler with the functionality of a Programmable Logic Control (PLC).</p> <p>Programming of the application is done with WAGO-I/O-PRO 32 in accordance with IEC 61131-3. Via the API socket the user can program all clients and servers for the transport protocols (TCP, UDP,...) using function modules.</p> <p>Characteristics and use:</p> <ul style="list-style-type: none"> • The use of decentralized control can better support a PLC or PC • Signal pre-processing reduces fieldbus transmissions • Complex applications can be divided into multiple tasks • Tasks can be prioritized • Peripheral equipment can be controlled directly, resulting in faster system response times • Programmable response in the event of a fieldbus failure • Simple, self-sufficient control 	<p>System Data</p> <table border="1"> <tr> <td>Max. no. of nodes</td> <td>limited by ETHERNET specification</td> </tr> <tr> <td>Transmission medium</td> <td>Twisted Pair S-UTP 100Ω cat. 5</td> </tr> <tr> <td>Buscoupler connection</td> <td>RJ45</td> </tr> <tr> <td>Max. length of fieldbus segment</td> <td>100m between hub station and 750-842</td> </tr> <tr> <td></td> <td>max. length of network limited by ETHERNET specification</td> </tr> <tr> <td>Baud rate</td> <td>10Mbits/s</td> </tr> <tr> <td>Protocols</td> <td>MODBUS /TCP, HTTP, BootP MODBUS / UDP</td> </tr> <tr> <td>Programming</td> <td>WAGO-I/O-PRO</td> </tr> <tr> <td>IEC 61131-3</td> <td>IL, LD, FBD, ST, FC</td> </tr> </table> <p>Approvals</p> <table border="1"> <tr> <td>UL</td> <td>E175199, UL508</td> </tr> <tr> <td>Conformity marking</td> <td>CE</td> </tr> </table> <p>Accessories</p> <table border="1"> <tr> <td>Miniature WSB quick marking system</td> <td>pages 1.166/167</td> </tr> </table>	Max. no. of nodes	limited by ETHERNET specification	Transmission medium	Twisted Pair S-UTP 100Ω cat. 5	Buscoupler connection	RJ45	Max. length of fieldbus segment	100m between hub station and 750-842		max. length of network limited by ETHERNET specification	Baud rate	10Mbits/s	Protocols	MODBUS /TCP, HTTP, BootP MODBUS / UDP	Programming	WAGO-I/O-PRO	IEC 61131-3	IL, LD, FBD, ST, FC	UL	E175199, UL508	Conformity marking	CE	Miniature WSB quick marking system	pages 1.166/167	
	Max. no. of nodes	limited by ETHERNET specification																								
	Transmission medium	Twisted Pair S-UTP 100Ω cat. 5																								
	Buscoupler connection	RJ45																								
	Max. length of fieldbus segment	100m between hub station and 750-842																								
		max. length of network limited by ETHERNET specification																								
	Baud rate	10Mbits/s																								
	Protocols	MODBUS /TCP, HTTP, BootP MODBUS / UDP																								
	Programming	WAGO-I/O-PRO																								
	IEC 61131-3	IL, LD, FBD, ST, FC																								
UL	E175199, UL508																									
Conformity marking	CE																									
Miniature WSB quick marking system	pages 1.166/167																									



Technical Data	
Max. no. of I/O modules	64
Fieldbus	
Input process image	max. 512 bytes
Output process image	max. 512 bytes
Input variables	max. 512 bytes
Output variables	max. 512 bytes
Configuration	automatic
Program memory	128 kbytes
Data memory	64 kbytes
Non-volatile memory	8 kbytes
Cycle time	< 3 ms for 1,000 statements/256 dig. I/Os
Voltage supply	DC 24 V (-15% .. +20%)
Input current max.	500 mA at 24 V
Efficiency of the power supply	87%
Total current for I/O modules	1800 mA at 5 V
Internal current consumption	200 mA at 5 V
Isolation	500 V system / supply
Voltage via power jumper contacts max.	DC 24 V (-15% ... +20%)
Current via power jumper contacts max.	DC 10 A
Operating temperature	0 °C ... +55 °C
Wire connection	CAGE CLAMP®
	0.08 mm ² ... 2.5 mm ²
	AWG 28 ... 14
	8 ... 9 mm / 0.33 in stripped length
Dimensions (mm) W x H x L	51 x 65* x 100
	* from upper edge of DIN 35 rail
Weight	ca 195 g
Storage temperature	-25 °C .. +85 °C
Relative air humidity	95% no condensation
Vibration and shock resistance	acc. to IEC 60068-2-6 acc. to IEC 60068-2-27
Degree of protection	IP 20
EMC	
Immunity to interference	acc. to EN 50082-2 (96)
Emission of interference	acc. to EN 50081-2 (94)
Availability date on request	

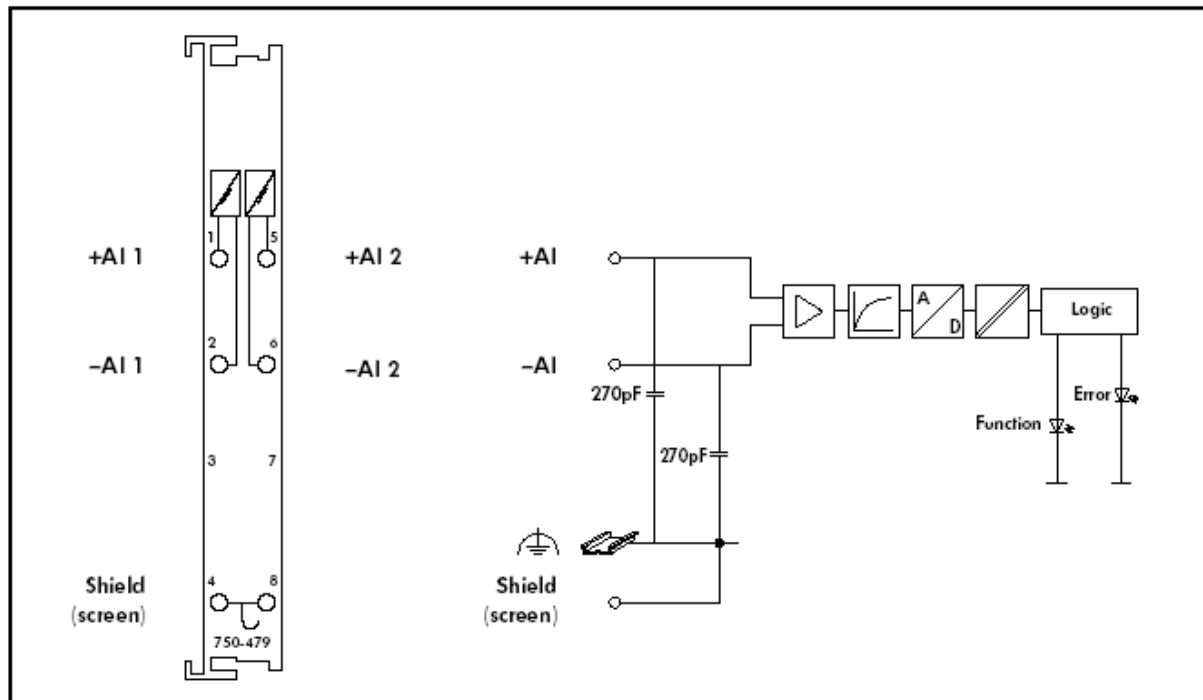
2-Channel Analog Input Module $\pm 10\text{ V}$

differential measurement input



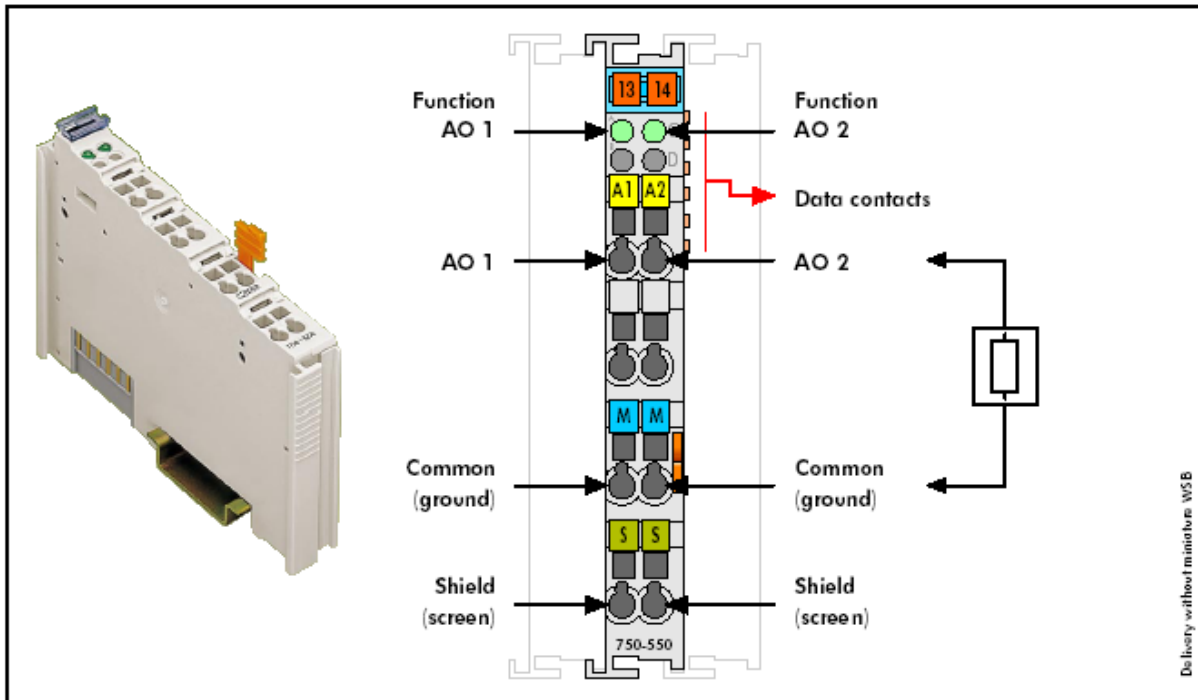
Do delivery without miniature: W5B

Description	Item-No.	Pack.-unit pcs																																																						
2AI $\pm 10\text{V}$ DC Diff. Measur. Inp.	750-479	1																																																						
<p>The analog input module receives differential signals of values $\pm 10\text{ V}$ DC. The input signal of each channel is electrically isolated and will be transmitted with a resolution of 13 bits.</p> <p>The system supply (via the data bus contacts) is used for the power supply of the module.</p> <p>The shield (screen) is directly connected to the DIN rail.</p>																																																								
<p>Technical Data</p> <table border="1"> <tbody> <tr> <td>No. of inputs</td> <td>2, electrically isolated from each other</td> </tr> <tr> <td>Measured-value acquisition</td> <td>time synchronous (both inputs)</td> </tr> <tr> <td>Voltage supply</td> <td>via system voltage DC/DC</td> </tr> <tr> <td>Current consumption (internal)</td> <td>$\leq 100\text{ mA}$</td> </tr> <tr> <td>Signal voltage</td> <td>$\pm 10\text{ V}$</td> </tr> <tr> <td>Overrange/ measuring range underflow</td> <td>status byte and LED</td> </tr> <tr> <td>Input filter</td> <td>low pass first order, $f_G = 5\text{ kHz}$</td> </tr> <tr> <td>Internal resistance</td> <td>$1\text{ M}\Omega$</td> </tr> <tr> <td>Resolution of the A/D converter</td> <td>14 bits</td> </tr> <tr> <td>Monotonicity without missing codes</td> <td>yes</td> </tr> <tr> <td>Resolution of the measured value</td> <td>13 bits + sign bit</td> </tr> <tr> <td>Value of a LSB (Least Significant Bit)</td> <td>1.2 mV</td> </tr> <tr> <td>Measuring error 25°C</td> <td>$\leq \pm 0.05\%$ of the full scale value</td> </tr> <tr> <td>Temperature coefficient</td> <td>$< \pm 0.01\%$ / K of the full scale value</td> </tr> <tr> <td>Measuring error</td> <td>$\leq 0.4\%$ over whole temperature range $\leq 0.1\%$ of upper range value (non-linearity)</td> </tr> <tr> <td>Crosstalk attenuation</td> <td>$\geq 80\text{ dB}$</td> </tr> <tr> <td>Sampling time of repetition</td> <td>1 ms</td> </tr> <tr> <td>Sampling delay (module)</td> <td>1 ms</td> </tr> <tr> <td>Sampling delay (channel/channel)</td> <td>$\leq 1\ \mu\text{s}$</td> </tr> <tr> <td>Sampling duration</td> <td>$\leq 5\ \mu\text{s}$</td> </tr> <tr> <td>Method of conversion</td> <td>SAR (Successive Approximation Register)</td> </tr> <tr> <td>Operating mode</td> <td>continuously sampling (preset)</td> </tr> <tr> <td>Protection</td> <td>RC circuit</td> </tr> <tr> <td>Admissible continuous overload</td> <td>230 V</td> </tr> <tr> <td>Voltage resistance</td> <td>DC 500 V channel/channel or channel/system</td> </tr> <tr> <td>Bit width</td> <td>2×16 bits data 2×8 bits control/status (option)</td> </tr> <tr> <td>Operating temperature</td> <td>$0^\circ\text{C} \dots +55^\circ\text{C}$</td> </tr> </tbody> </table>			No. of inputs	2, electrically isolated from each other	Measured-value acquisition	time synchronous (both inputs)	Voltage supply	via system voltage DC/DC	Current consumption (internal)	$\leq 100\text{ mA}$	Signal voltage	$\pm 10\text{ V}$	Overrange/ measuring range underflow	status byte and LED	Input filter	low pass first order, $f_G = 5\text{ kHz}$	Internal resistance	$1\text{ M}\Omega$	Resolution of the A/D converter	14 bits	Monotonicity without missing codes	yes	Resolution of the measured value	13 bits + sign bit	Value of a LSB (Least Significant Bit)	1.2 mV	Measuring error 25°C	$\leq \pm 0.05\%$ of the full scale value	Temperature coefficient	$< \pm 0.01\%$ / K of the full scale value	Measuring error	$\leq 0.4\%$ over whole temperature range $\leq 0.1\%$ of upper range value (non-linearity)	Crosstalk attenuation	$\geq 80\text{ dB}$	Sampling time of repetition	1 ms	Sampling delay (module)	1 ms	Sampling delay (channel/channel)	$\leq 1\ \mu\text{s}$	Sampling duration	$\leq 5\ \mu\text{s}$	Method of conversion	SAR (Successive Approximation Register)	Operating mode	continuously sampling (preset)	Protection	RC circuit	Admissible continuous overload	230 V	Voltage resistance	DC 500 V channel/channel or channel/system	Bit width	2×16 bits data 2×8 bits control/status (option)	Operating temperature	$0^\circ\text{C} \dots +55^\circ\text{C}$
No. of inputs	2, electrically isolated from each other																																																							
Measured-value acquisition	time synchronous (both inputs)																																																							
Voltage supply	via system voltage DC/DC																																																							
Current consumption (internal)	$\leq 100\text{ mA}$																																																							
Signal voltage	$\pm 10\text{ V}$																																																							
Overrange/ measuring range underflow	status byte and LED																																																							
Input filter	low pass first order, $f_G = 5\text{ kHz}$																																																							
Internal resistance	$1\text{ M}\Omega$																																																							
Resolution of the A/D converter	14 bits																																																							
Monotonicity without missing codes	yes																																																							
Resolution of the measured value	13 bits + sign bit																																																							
Value of a LSB (Least Significant Bit)	1.2 mV																																																							
Measuring error 25°C	$\leq \pm 0.05\%$ of the full scale value																																																							
Temperature coefficient	$< \pm 0.01\%$ / K of the full scale value																																																							
Measuring error	$\leq 0.4\%$ over whole temperature range $\leq 0.1\%$ of upper range value (non-linearity)																																																							
Crosstalk attenuation	$\geq 80\text{ dB}$																																																							
Sampling time of repetition	1 ms																																																							
Sampling delay (module)	1 ms																																																							
Sampling delay (channel/channel)	$\leq 1\ \mu\text{s}$																																																							
Sampling duration	$\leq 5\ \mu\text{s}$																																																							
Method of conversion	SAR (Successive Approximation Register)																																																							
Operating mode	continuously sampling (preset)																																																							
Protection	RC circuit																																																							
Admissible continuous overload	230 V																																																							
Voltage resistance	DC 500 V channel/channel or channel/system																																																							
Bit width	2×16 bits data 2×8 bits control/status (option)																																																							
Operating temperature	$0^\circ\text{C} \dots +55^\circ\text{C}$																																																							



Technical Data		Variations	Item-No.	Pack.-unit pcs
Wire connection	CAGE CLAMP® 0.08 mm² ... 2.5 mm² AWG 28 ... 14 8 ... 9 mm/ 0.33 in stripped length	2AI ±10V DC Diff. Measur. Inp. Synchronous Measured-value acquisition Overrange/ measuring range underflow	750-479/000-001	1
Dimensions (mm) W x H x L	12 x 64* x 100 * from upper edge of 35 DIN rail			
Weight	ca 55 g	Sampling delay (instruction/ conversion)	≤ 50 µs	
Storage temperature	-25 °C ... +85 °C	Operating mode	triggered	
Relative air humidity	95 % no condensation			
Vibration and shock resistance	acc. to IEC 60068-2-6 acc. to IEC 60068-2-27			
Degree of protection	IP 20			
EMC		¹) In connection with synchronized sampling of the slave (fieldbus coupler 750-303 (as from version 0101))		
Immunity to interference	acc. to EN 50082-2 (96)	²) Minimum/ maximum limiting values can also be set according to customers' specifications!		
Emission of interference	acc. to EN 50081-2 (94)			
Approvals				
UL	E175199, UL508 (applied for)			
Conformity marking	CE			
Accessories				
Miniature WSB quick marking system	pages 1.166/167			

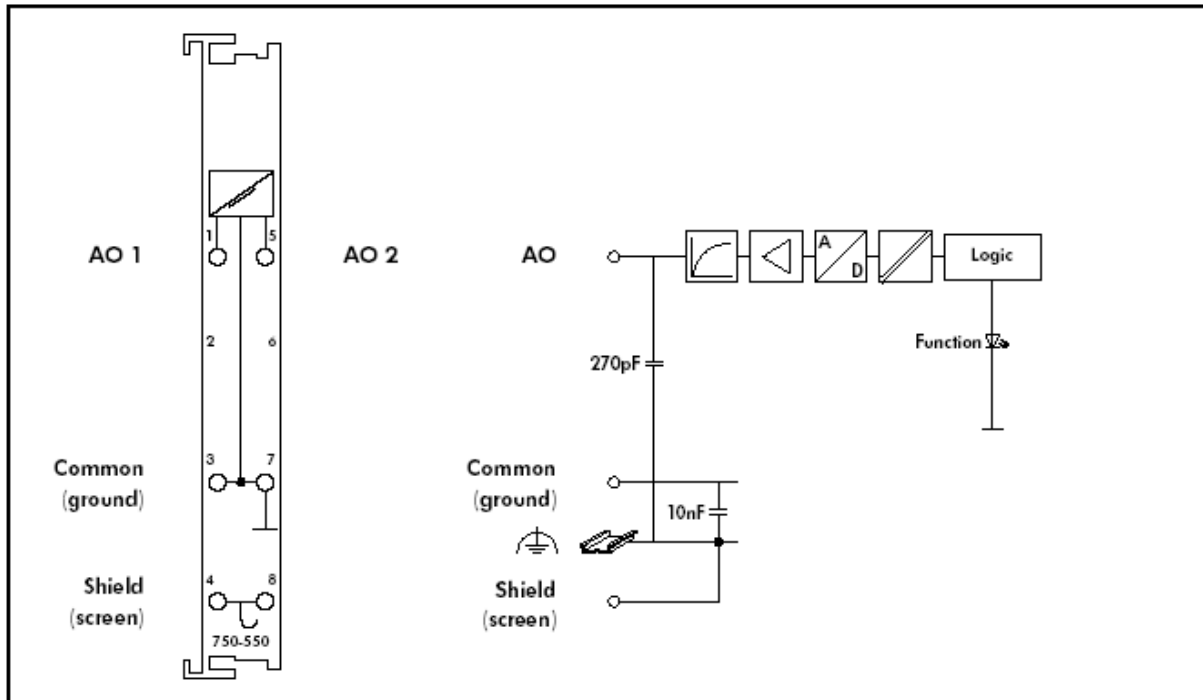
2-Channel Analog Output Module 0-10 V/ ±10 V



Delivery without miniature WSB

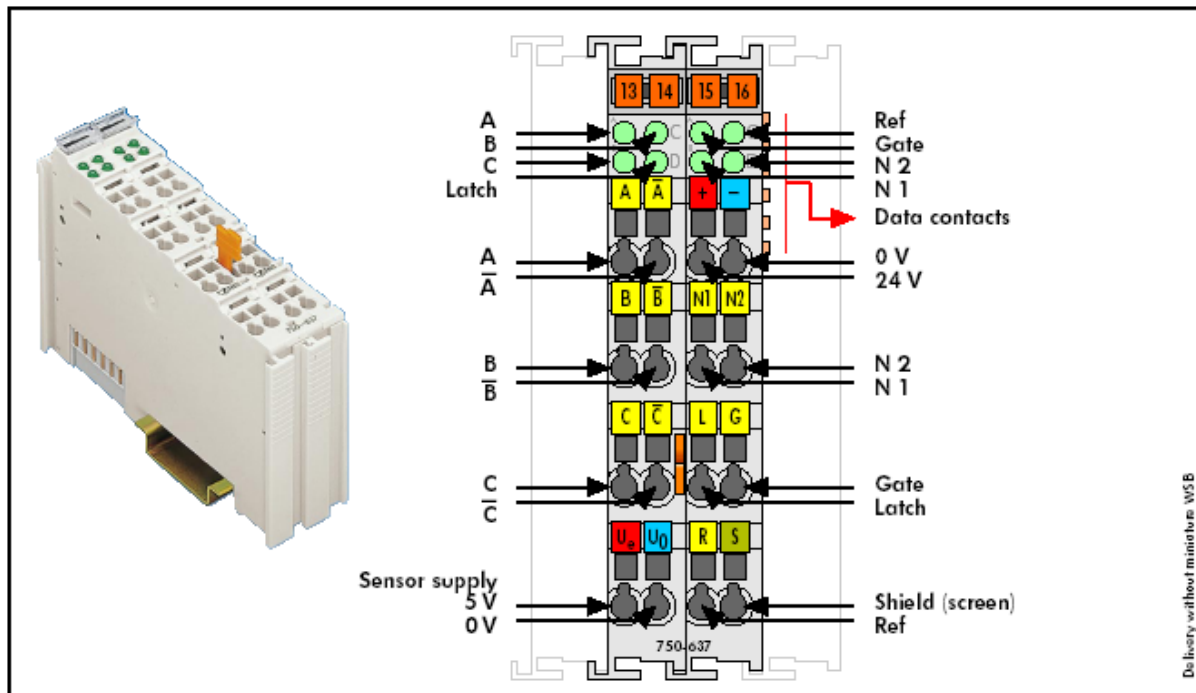
Description	Item-No.	Pack.-unit pcs
2AO 0-10 V DC	750-550	10
2AO ±10 V DC	750-556	10

Description	Technical Data	
	Item-No.	Pack.-unit pcs
The analog output module creates a standardized signal of 0-10 V or ±10 V.		
The output signal is electrically isolated and will be transmitted with a resolution of 12 bits.		
The internal system supply is used for the power supply of the module.		
The output channels have one common ground potential.		
No. of outputs	2	
Current consumption (internal)	65 mA	
Voltage supply	via system voltage DC/DC	
Signal voltage	0 V ... 10 V (750-550) ±10 V (750-556)	
Load impedance	> 5kΩ	
Resolution	12 bits	
Conversion time	ca 2 ms	
Measuring error 25°C	< ± 0.1% of the full scale value	
Temperature coefficient	< ± 0.01%/K of the full scale value	
Isolation	500 V system/supply	
Bit width	2 x 16 bits data 2 x 8 bits control/status (option)	
Operating temperature	0°C ... +55°C	
Wire connection	CAGE CLAMP® 0.08 mm ² ... 2.5 mm ² AWG 28 ... 14	
Dimensions (mm) W x H x L	8 ... 9 mm / 0.33 in stripped length 12 x 64* x 100 * from upper edge of 35 DIN rail	
Weight	ca 55 g	
Storage temperature	-25°C ... +85°C	
Relative air humidity	95% no condensation	
Vibration and shock resistance	acc. to IEC 60068-2-6 acc. to IEC 60068-2-27	
Degree of protection	IP 20	
EMC		
Immunity to interference	acc. to EN 50082-2 (96)	
Emission of interference	acc. to EN 50081-2 (94)	



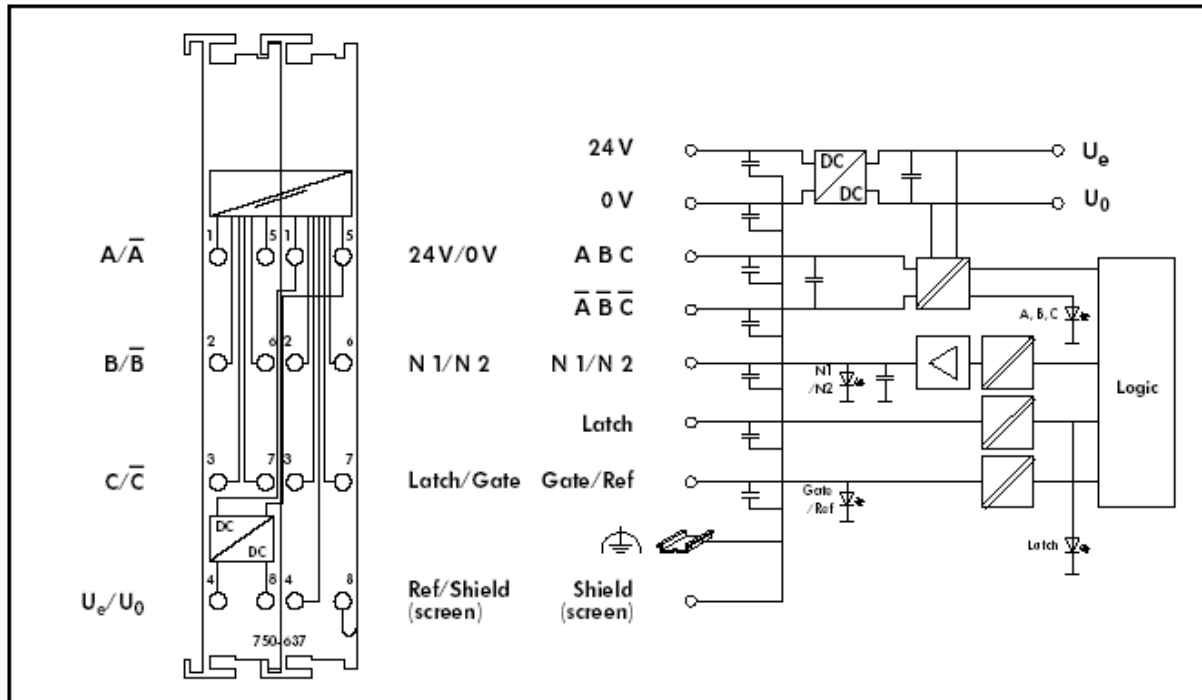
Technical Data	Variations	Item-No.	Pack.-unit pcs
	2AO 0-10V DC SS ¹⁾	750-550/000-200	1
	2AO ±10V DC SS ¹⁾	750-556/000-200	1
	¹⁾ Data format for SS control with FB 251.		
Approvals			
UL	E175199, UL508		
UL	E198726, UL1604		
	Class I Div2 ABCD T4A		
KEMA	01ATEX 1024 X		
Conformity marking	EEc nA II T4		
	CE		
Accessories			
Miniature WSB quick marking system pages 1.166/167			

Incremental Encoder Interface



Delivery without miniature VSB

Description	Item-No.	Pack.-unit																																																										
Incremental Encoder	750-637	2 pcs																																																										
<p>This module is an interface for any incremental encoder with a RS 422 connection. A counter with quadrature decoder as well as a latch for the zero impulse can be read or enabled by the control. The control can set the counter or transmit the counter value to the Latch. As an alternative this can also be done using input "C" or "Latch".</p> <p>The frequency data is automatically acquired and can also be transmitted to the control.</p> <p>A counter lock-out is possible using input G. Input "Ref" can be used to activate the initial point "C" function.</p> <p>The outputs N1 and N2 indicate whether the counter value is within a defined range of values. The range can be adjusted.</p> <p>The module must be powered using an external 24 V DC power supply. It is then possible to supply the encoder with 24 V DC, or alternatively with 5 V DC derived internally from the terminations (V_e/V₀).</p> <p>The shield (screen) is directly connected to the carrier rail.</p>																																																												
<p>Technical Data</p> <table border="1"> <tbody> <tr><td>Sensor connection</td><td>A, A', B, B', C, C'</td></tr> <tr><td>Current consumption (internal)</td><td>110 mA</td></tr> <tr><td>Counter</td><td>32 bits binary</td></tr> <tr><td>Max. operating frequency</td><td>250 kHz</td></tr> <tr><td>Quadrature decoder</td><td>4-fold report</td></tr> <tr><td>Zero impulse latch</td><td>32 bits</td></tr> <tr><td>Commands</td><td>read, set, enable</td></tr> <tr><td>Voltage supply</td><td>DC 24 V (-15% ... +20%)</td></tr> <tr><td>Current supply typ.</td><td>35 mA without load</td></tr> <tr><td>Operating voltage of sensor</td><td>DC 5 V</td></tr> <tr><td>Current output (sensor) max.</td><td>300 mA</td></tr> <tr><td>Internal bit width</td><td>1 x 32 bits data</td></tr> <tr><td></td><td>2 x 8 bits control/status</td></tr> <tr><td>Operating temperature</td><td>0 °C ... +55 °C</td></tr> <tr><td>Wire connection</td><td>CAGE CLAMP®</td></tr> <tr><td></td><td>0.08 mm² ... 2.5 mm²</td></tr> <tr><td></td><td>AWG 28 ... 14</td></tr> <tr><td>Dimensions (mm) W x H x L</td><td>8 ... 9 mm / 0.33 in stripped length</td></tr> <tr><td></td><td>12 x 64* x 100</td></tr> <tr><td></td><td>* from upper edge of DIN 35 rail</td></tr> <tr><td>Weight</td><td>ca 105 g</td></tr> <tr><td>Storage temperature</td><td>-25 °C ... +85 °C</td></tr> <tr><td>Relative air humidity</td><td>95% no condensation</td></tr> <tr><td>Vibration and shock resistance</td><td>acc. to IEC 60068-2-6</td></tr> <tr><td></td><td>acc. to IEC 60068-2-27</td></tr> <tr><td>Degree of protection</td><td>IP 20</td></tr> <tr><td>EMC</td><td></td></tr> <tr><td>Immunity to interference</td><td>acc. to EN 50082-2 (96)</td></tr> <tr><td>Emission of interference</td><td>acc. to EN 50081-2 (94)</td></tr> </tbody> </table>			Sensor connection	A, A', B, B', C, C'	Current consumption (internal)	110 mA	Counter	32 bits binary	Max. operating frequency	250 kHz	Quadrature decoder	4-fold report	Zero impulse latch	32 bits	Commands	read, set, enable	Voltage supply	DC 24 V (-15% ... +20%)	Current supply typ.	35 mA without load	Operating voltage of sensor	DC 5 V	Current output (sensor) max.	300 mA	Internal bit width	1 x 32 bits data		2 x 8 bits control/status	Operating temperature	0 °C ... +55 °C	Wire connection	CAGE CLAMP®		0.08 mm ² ... 2.5 mm ²		AWG 28 ... 14	Dimensions (mm) W x H x L	8 ... 9 mm / 0.33 in stripped length		12 x 64* x 100		* from upper edge of DIN 35 rail	Weight	ca 105 g	Storage temperature	-25 °C ... +85 °C	Relative air humidity	95% no condensation	Vibration and shock resistance	acc. to IEC 60068-2-6		acc. to IEC 60068-2-27	Degree of protection	IP 20	EMC		Immunity to interference	acc. to EN 50082-2 (96)	Emission of interference	acc. to EN 50081-2 (94)
Sensor connection	A, A', B, B', C, C'																																																											
Current consumption (internal)	110 mA																																																											
Counter	32 bits binary																																																											
Max. operating frequency	250 kHz																																																											
Quadrature decoder	4-fold report																																																											
Zero impulse latch	32 bits																																																											
Commands	read, set, enable																																																											
Voltage supply	DC 24 V (-15% ... +20%)																																																											
Current supply typ.	35 mA without load																																																											
Operating voltage of sensor	DC 5 V																																																											
Current output (sensor) max.	300 mA																																																											
Internal bit width	1 x 32 bits data																																																											
	2 x 8 bits control/status																																																											
Operating temperature	0 °C ... +55 °C																																																											
Wire connection	CAGE CLAMP®																																																											
	0.08 mm ² ... 2.5 mm ²																																																											
	AWG 28 ... 14																																																											
Dimensions (mm) W x H x L	8 ... 9 mm / 0.33 in stripped length																																																											
	12 x 64* x 100																																																											
	* from upper edge of DIN 35 rail																																																											
Weight	ca 105 g																																																											
Storage temperature	-25 °C ... +85 °C																																																											
Relative air humidity	95% no condensation																																																											
Vibration and shock resistance	acc. to IEC 60068-2-6																																																											
	acc. to IEC 60068-2-27																																																											
Degree of protection	IP 20																																																											
EMC																																																												
Immunity to interference	acc. to EN 50082-2 (96)																																																											
Emission of interference	acc. to EN 50081-2 (94)																																																											



Technical Data	
Digital outputs (N1, N2)	
Output voltage	DC 24V
Output current max.	0.5A short-circuit proof
Digital inputs (Latch, Gate, Ref)	
Signal voltage (0)	DC -3V to 5V
Signal voltage (1)	DC 15V to 30V
Input current I _p .	
Latch	5 mA
Gate	7 mA
Ref.	7 mA
A/A	24V DC
B/B	Input
C/C	squarewave
Approvals	
UL	E175199, UL508
Conformity marking	CE
Accessories	
Miniature WSB quick marking system	pages 1.166/167

Bibliography

cbb software GmbH [LabMap](#)[®] Handbook
Wago ModBus_HandBook_e.pdf
Wago Ethernet TCP IP HandBook_e.pdf
Wago Analog_In_750-479_e.pdf
Wago Analog_Out_750-556_e.pdf
Wago Digital_Out_750-517_e.pdf
Wago Inkremental Enc 750-637_e.pdf