Rexroth Bosch Group

Rexroth IndraLogic 04VRS Modbus TCP

R911326406
Edition 02

Application Description

Title
Rexroth IndraLogic 04VRS
Modbus TCP

Type of Documentation
Application Description

Document Typecode
DOK-IL*1G*-MOD*TCP*V04-AW02-EN-P

Internal File Reference
RS-831ddf1aacc77bd10a6846a001e494ac-2-en-US-4

Purpose of Documentation
This documentation describes how to establish a communication via a Modbus TCP.

Record of Revision

Edition	Release Date	Notes
120-3450-B301-01/EN	10.2008	First edition
120-3450-B301-02/EN	12.2008	Modifications implemented

Published by
Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2 ■ 97816 Lohr a. Main, Germany
Phone +49 (0)93 52/ 40-0 ■ Fax +49 (0)93 52/ 40-48 85
http://www.boschrexroth.com/
Dept. BRC/EAM (StFo/HaBu) (SyMu/MePe)

Note
This document has been printed on chlorine-free bleached paper.

Table of Contents

1 Modbus TCP

1.1 Introduction and Overview

The Modbus TCP is used for the data exchange between Ethernet devices supporting the Modbus protocol. This documentation focuses on HMI devices. The Modbus slave (server) is the control used for the Modbus masters (clients) to connect. A Modbus TCP server is implemented in each of the "**IL_ModBusTCPServer**" and "**IL_ModBusTCPServerType02** " FBs. The RIL_ModBusTCP.library contains these FBs.

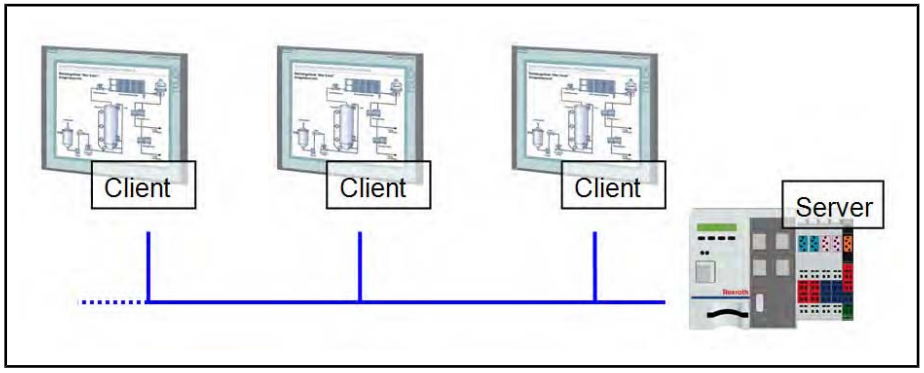

Fig.1-1: Connecting several Modbus clients to a Modbus server

The functions blocks IL_ModBusTCPServer and IL_ModBusTCPServerType02 differ in the following:

In contrast to the Modbus specification, the data areas for "Coil Data" and "Discrete Input Data" are summarized under "BitData", the "Holding Register" and "Input Register" are summarized under "RegisterData" in the "IL_ModBusTCPServer" FB.

1.2 The IL_ModbusTCPServer FB

Brief description

A Modbus TCP slave is implemented in the IL_ModbusTCPServer FB. Discrete Inputs, Coils, Input Registers and Holding Registers are supported with read and write access. Arrays are used for the data exchange.

Interface description

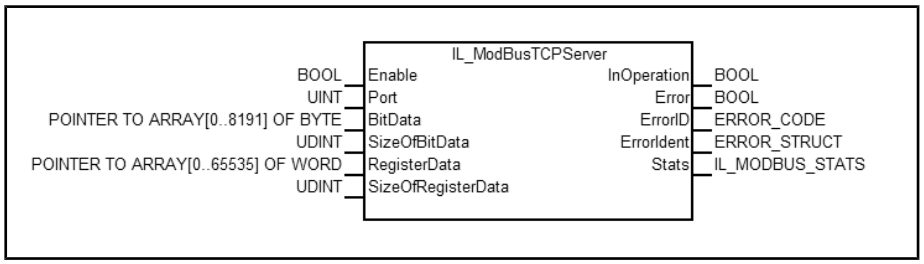

Fig.1-2: The IL_ModbusTCPServer FB

Modbus TCP

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Function block processing enable (level-controlled).
	Port	UINT	The TCP port at which incoming connections are expected.
	BitData	POINTER TO ARRAY[0..8191] OF BYTE	Pointer on bit data.
	SizeOfBitData	UDINT	Number of bytes in BitData (bit number / 8).
	RegisterData	POINTER TO ARRAY[0..65535] OF WORD	Pointer on register data in word width.
	SizeOfRegisterData	UDINT	Number of bytes in the register data (register number * 2).
VAR OUTPUT	InOperation	BOOL	The Modbus server runs and accepts incoming connections.
	Error	BOOL	An error while processing the FB.
	ErrorID	ERROR_ CODE	When the "Error" output is set, this output contains a broad error classification.
	ErrorIdent	ERROR_ STRUCT	In case of a set "Error" output, this output contains detailed error information See also "Troubleshooting" on page 4.
	Stats	IL_MODBUS_STATS	Contains counter for data packages and errors.

Fig. 1-3: IL_ModbusTCPServer interface

Name	Type	Min. value	Max. value	Default value	Transfer
Enable	BOOL			FALSE	Continuous
Port	UINT	0	65535	502	Rising edge at "Enable"
BitData	POINTER	0	n.def.	0	Rising edge at "Enable"
SizeOfBitData	UDINT	0	8191	0	Rising edge at "Enable"
RegisterData	POINTER	0	n.def.	0	Rising edge at "Enable"
SizeOfRegisterData	UDINT	0	131070	0	Rising edge at "Enable"

Fig. 1-4: IL_ModbusTCPServer input behaviour

Name	Type	Default value	Description
NumReadBits	UDINT	0	Number of read access on bits
NumReadRegisters	UDINT	0	Number of read access on register
NumWriteBit	UDINT	0	Number of single write access on bits
NumWriteRegister	UDINT	0	Number of single write access on register
NumWriteBits	UDINT	0	Number of multiple write access on bits
NumWriteRegisters	UDINT	0	Number of multiple write access on register

Name	Type	Default value	Description
NumReadWriteRegs	UDINT	0	Number of combined read / write access on register
NumExceptions	UDINT	0	Number of Modbus errors
ConnectedClients	ARRAY[0..15] OF String(16)		IP addresses of the clients connected

Fig.1-5: IL_MODBUS_STATS data structure

Functional Description

The IL_ModbusTCPServer FB implements a Modbus slave that can connect up to 16 Modbus masters. The FB is passive during the data exchange. Being passive means that the Modbus master specifies which data is read and which data is written. Discrete Inputs, Coils, Input Registers and Holding Registers are supported.

☞ In contrast to the Modbus specification, the respective Discrete Inputs, Coils as well as Input Registers and Holding Registers are imaged on the same data range.

In case more than 16 participants try to connect simultaneously, the connection to the most previous data exchange is aborted before a new connection is accepted. This behavior complies with the Modbus TCP specification.
This should be avoided in practice. Therefore, it is useful to open a second Modbus TCP server on a different port.

Enable, InOperation

"Enable" allows the processing of the FB. The FB is active as long as "Enable" is TRUE. In case of a rising edge at "Enable", all inputs are latched and the FB is initialized. After the initialization is completed, the output is set to "InOperation" to indicate that the incoming Modbus TCP connections are accepted and can now be processed. At a falling edge at "Enable", the existing connections are closed and the port is released.

Error, ErrorID, ErrorIdent

If an error occurs while processing the FB, the output "Error" is set. "ErrorID" and "ErrorIdent" contain detailed information on the error. The processing of the FB is paused. At a falling edge at "Enable", the "Error" output is reset.

The error outputs are only set if errors occur while processing the FB. Errors caused by the Modbus TCP protocol do not generate any output at the error outputs. The "NumExceptions" element in the "Stats" output is used for that purpose.

Port

Generally, the Modbus TCP uses the port 502, but a Modbus TCP server can be optionally opened on a different port as well. Therefore, the port number to be used has to be written on the "Port" input before the FB is enabled.

BitData, SizeOfBitData

The "BitData" input corresponds to the Discrete Inputs and Coils defined in the Modbus. Coils form a bit array with a length of 65536 bit max. Bits can be read and written. The data exchange is executed via the array assigned at the "BitData" input. Each array byte represents 8 bits.

The array size in byte is to be assigned at the "SizeOfBitData" input. The SIZEOF() operator can be used for this. The FB checks if the incoming requests are within the assigned array using the "SizeOfBitData".

☞ It is useful to define an own data structure to access the bits. The pragma {bitaccess } allows an access on the individual bits.

RegisterData, SizeOfRegisterData

The "RegisterData" input corresponds to the Input Registers and Holding Registers defined in the Modbus. The registers form an array of words (16 bit values) with a length of 65536 words max. The data exchange is executed via the array assigned at the "RegisterData" input.

The array size in byte is to be assigned at the "SizeOfRegisterData" input. The SIZEOF() operator can be used for this. The FB checks if the incoming requests are within the assigned array using the "SizeOfRegisterData".

☞ It is useful to define an own data structure to access the registers. The alignment of the data (16 bit) has to fit into words. Refer to "Example of a data structure for RegisterData" on page 11.

Stats

The "Stats" output contains a data structure indicating the status and statistics information on the running communication and the IP addresses of the connected clients. Read and write access is counted according to the elements. Protocol errors are only counted in the "NumExceptions" element. If the maximum number of the counter is exceeded, it restarts with 0. The maximum value of a counter is 2^{32}-1.

Supported Modbus function codes

Function codes are used by the Modbus master to access the Modbus slave. Knowledge about the codes supported helps to find the errors. It is not required for the normal operating mode though. The following function codes are supported:

Function code	Name	Description
1	Read Coils	Reads a number of bits from "BitData" starting from an initial address
2	Read Discrete Inputs	Reads a number of bits from "BitData" starting from an initial address
3	Read Holding Registers	Reads a number of registers from "RegisterData" starting from an initial address
4	Read Input Registers	Reads a number of registers from "RegisterData" starting from an initial address
5	Write Single Coil	Writes a bit into "BitData" starting from an initial address
6	Write Single Register	Writes a register into "RegisterData" starting from an initial address
15	Write Multiple Coils	Writes a number of bits into "BitData" starting from an initial address
16	Write Multiple Registers	Writes a number of registers into "RegisterData" starting from an initial address
23	Read/Write Multiple Registers	Executes a combined read / write operation on "RegisterData"
43/14	Read Device Identification	Reads the default device information of the Modbus TCP server: • VendorName:"Bosch Rexroth" • ProductCode: "IL_ModbusTCPServer" • MajorMinorRevision: "01Vxx"

Fig. 1-6: Supported Modbus function codes

☞ The function codes 1, 2, 5, 15 access a memory area assigned at the "BitData" input.

☞ The function codes 3, 4, 6, 16 access a memory area assigned at the "RegisterData" input.

Troubleshooting

The FB uses the F_RELATED_TABLE, 16#170x error table. It can generate the following error messages in Additional1 and Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR (16#0001)	16#00001800	16#00000000	The "BitData" input is a zero pointer.
INPUT_INVALID_ERROR (16#0001)	16#00001801	16#00000000	The "RegisterData" input is a zero pointer.
INPUT_INVALID_ERROR (16#0001)	16#00001802	16#00000000	The "SizeOfBitData" input is 0.
INPUT_INVALID_ERROR (16#0001)	16#00001803	16#00000000	The "SizeOfRegisterData" input is 0.
INPUT_INVALID_ERROR (16#0001)	16#00001804	16#00000000	The "SizeOfRegisterData" input is odd.
RESOURCE_ERROR (16#0003)	16#00001805	16#00000000	Could not open socket.
RESOURCE_ERROR (16#0003)	16#00001806	16#00000000	Could not accept incoming connection.
RESOURCE_ERROR (16#0003)	16#00001807	16#00000000	The port is assigned.
RESOURCE_ERROR (16#0003)	16#00001808	16#00000000	Port listen failed.

Fig.1-7: IL_ModbusTCPServer error codes

1.3 The IL_ModbusTCPServerType02 FB

Brief description

A Modbus TCP slave is implemented in the IL_ModbusTCPServerType02 FB. Discrete Inputs, Coils, Input Registers and Holding Registers are supported with read and write access. The data exchange is executed via 4 separate arrays for the individual data ranges.

Interface description

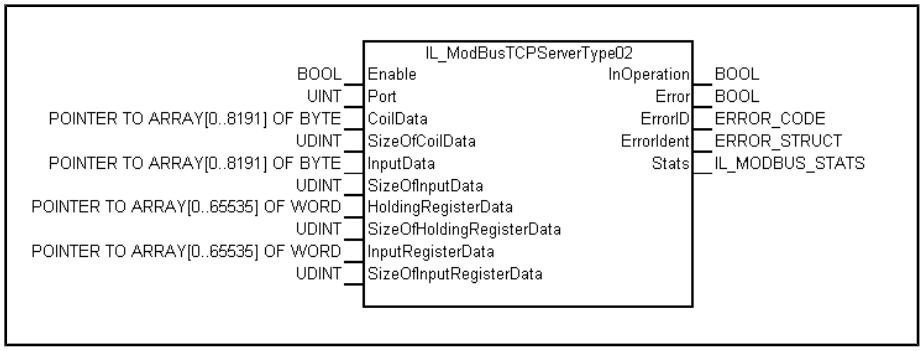

Fig.1-8: The IL_ModbusTCPServerType02 FB

Modbus TCP

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Function block processing enable (level-controlled).
	Port	UINT	The TCP port at which incoming connections are expected.
	CoilData	POINTER TO ARRAY[0..8191] OF BYTE	Pointer on coil data (bit data range is neither read-protected nor write-protected).
	SizeOfCoilData	UDINT	Number of bytes in CoilData (bit number / 8).
	InputData	POINTER TO ARRAY[0..8191] OF BYTE	Pointer on discrete input data (bit data range is not read-protected).
	SizeOfInputData	UDINT	Number of bytes in InputData (bit number / 8).
	HoldingRegisterData	POINTER TO ARRAY[0..65535] OF WORD	Pointer on holding register data in word width (register data range is neither read-protected nor write-protected).
	SizeOfHoldingRegisterData	UDINT	Number of bytes in the holding register data (register number * 2).
	InputRegisterData	POINTER TO ARRAY[0..65535] OF WORD	Pointer on input register data in word width (register data range is not read-protected).
	SizeOfHoldingRegisterData	UDINT	Number of bytes in the input register data (register number * 2).
VAR OUTPUT	InOperation	BOOL	The Modbus server runs and accepts incoming connections.
	Error	BOOL	An error while processing the FB.
	ErrorID	ERROR_ CODE	When the "Error" output is set, this output contains a broad error classification.
	ErrorIdent	ERROR_ STRUCT	In case of a set "Error" output, this output contains detailed error information See also "Troubleshooting" on page 9.
	Stats	IL_MODBUS_STATS	Contains counter for data packages and errors.

Fig.1-9: IL_ModbusTCPServerType02 interface

Name	Type	Min. value	Max. value	Default value	Transfer
Enable	BOOL			FALSE	Continuous
Port	UINT	0	65535	502	Rising edge at "Enable"
CoilData	POINTER	0	n.def.	0	Rising edge at "Enable"
SizeOfCoilData	UDINT	0	8191	0	Rising edge at "Enable"
InputData	POINTER	0	n.def.	0	Rising edge at "Enable"
SizeOfInputData	UDINT	0	8191	0	Rising edge at "Enable"
HoldingRegisterData	POINTER	0	n.def.	0	Rising edge at "Enable"
SizeOfHoldingRegisterData	UDINT	0	131070	0	Rising edge at "Enable"

Name	Type	Min. value	Max. value	Default value	Transfer
InputRegisterData	POINTER	0	n.def.	0	Rising edge at "Enable"
SizeOfInputRegister-Data	UDINT	0	131070	0	Rising edge at "Enable"

Fig.1-10: IL_ModbusTCPServerType02 input behaviour

Name	Type	Default value	Description
NumReadBits	UDINT	0	Number of read access on bits
NumReadRegisters	UDINT	0	Number of read access on register
NumWriteBit	UDINT	0	Number of single write access on bits
NumWriteRegister	UDINT	0	Number of single write access on register
NumWriteBits	UDINT	0	Number of multiple write access on bits
NumWriteRegisters	UDINT	0	Number of multiple write access on register
NumReadWriteRegs	UDINT	0	Number of combined read / write access on register
NumExceptions	UDINT	0	Number of Modbus errors
ConnectedClients	ARRAY[0..15] OF String(16)		IP addresses of the clients connected

Fig.1-11: IL_MODBUS_STATS data structure

Functional Description

The IL_ModbusTCPServerType02 FB implements a Modbus slave that can connect up to 16 Modbus masters. The FB is passive during the data exchange. Being passive means that the Modbus master (client) specifies which data is read and which data is written. Discrete Inputs, Coils, Input Registers and Holding Registers are supported. As in the Modbus specification, the Discrete Inputs, Coils as well as Input Registers and Holding Registers are imaged on separate data ranges.

In case more than 16 participants try to connect simultaneously, the connection to the most previous data exchange is aborted before a new connection is accepted. This behavior complies with the Modbus TCP specification.
This should be avoided in practice. Therefore, it is useful to open a second Modbus TCP server on a different port.

Enable, InOperation

"Enable" allows the processing of the FB. The FB is active as long as "Enable" is TRUE. In case of a rising edge at "Enable", all inputs are latched and the FB is initialized. After the initialization is completed, the output is set to "InOperation" to indicate that the incoming Modbus TCP connections are accepted and can now be processed. At a falling edge at "Enable", the existing connections are closed and the port is released.

Error, ErrorID, ErrorIdent

If an error occurs while processing the FB, the output "Error" is set. "ErrorID" and "ErrorIdent" contain detailed information on the error. The processing of the FB is paused. At a falling edge at "Enable", the "Error" output is reset.

The error outputs are only set if errors occur while processing the FB. Errors caused by the Modbus TCP protocol do not generate any output at the error outputs. The "NumExceptions" element in the "Stats" output is used for that purpose.

Modbus TCP

Port

Generally, the Modbus TCP uses the port 502, but a Modbus TCP server can be optionally opened on a different port as well. Therefore, the port number to be used has to be written on the "Port" input before the FB is enabled.

CoilData, SizeOfCoilData

The "CoilData" input corresponds to the coils defined in the Modbus. Coils form a bit array with a length of 65536 bit max. Bits can be read and written. The data exchange is executed via the array assigned at the "CoilData" input. Each array byte represents 8 bits.

The array size in byte is to be assigned at the "SizeOfCoilData" input. The SIZEOF() operator can be used for this. The FB checks if the incoming requests are within the assigned array using the "SizeOfCoilData".

☞ It is useful to define an own data structure to access the bits. The pragma {bitaccess } allows an access on the individual bits.

InputData, SizeOfInputData

The "InputData" input corresponds to the discrete inputs defined in the Modbus. Discrete inputs form a bit array with a length of 65536 bit max. The bits can only be read. The data exchange is executed via the array assigned at the "InputData" input. Each array byte represents 8 bits.

The array size in byte is to be assigned at the "SizeOfInputData" input. The SIZEOF() operator can be used for this. The FB checks if the incoming requests are within the assigned array using the "SizeOfInputData".

☞ It is useful to define an own data structure to access the bits. The pragma {bitaccess } allows an access on the individual bits.

HoldingRegisterData, SizeOfHoldingRegisterData

The "HoldingRegisterData" input corresponds to the holding register defined in the Modbus. The registers form an array of words (16 bit values) with a length of 65536 words max. The register can be read and written. The data exchange is executed via the array assigned at the "HoldingRegisterData" input.

The array size in byte is to be assigned at the "SizeOfHoldingRegisterData" input. The SIZEOF() operator can be used for this. The FB checks if the incoming requests are within the assigned array using the "SizeOfHoldingRegisterData".

☞ It is useful to define an own data structure to access the registers. The alignment of the data (16 bit) has to fit into words. Refer to "Example of a data structure for RegisterData" on page 11.

InputRegisterData, SizeOfInputRegisterData

The "InputRegisterData" input corresponds to the input register defined in the Modbus. The registers form an array of words (16 bit values) with a length of 65536 words max. The registers can only be read. The data exchange is executed via the array assigned at the "InputRegisterData" input.

The array size in byte is to be assigned at the "SizeOfInputRegisterData" input. The SIZEOF() operator can be used for this. The FB checks if the incoming requests are within the assigned array using the "SizeOfInputRegisterData".

☞ It is useful to define an own data structure to access the registers. The alignment of the data (16 bit) has to fit into words. Refer to "Example of a data structure for RegisterData" on page 11.

Stats

The "Stats" output contains a data structure indicating the status and statistics information on the running communication and the IP addresses of the connected clients. Read and write access is counted according to the elements. Protocol errors are only counted in the "NumExceptions" element. If the maximum number of the counter is exceeded, it restarts with 0. The maximum value of a counter is 2^{32}-1.

Supported Modbus function codes

Function codes are used by the Modbus master (client) to access the Modbus slave (server). Knowledge about the codes supported helps to find the errors. It is not required for the normal operating mode though. The following function codes are supported:

Function code	Name	Description
1	Read Coils	Reads a number of bits from "CoilData" starting from an initial address
2	Read Discrete Inputs	Reads a number of bits from "InputData" starting from an initial address
3	Read Holding Registers	Reads a number of registers from "HoldingRegisterData" starting from an initial address
4	Read Input Registers	Reads a number of registers from "InputRegisterData" starting from an initial address
5	Write Single Coil	Writes a bit into "CoilData" starting from an initial address
6	Write Single Register	Writes a register into "HoldingRegisterData" starting from an initial address
15	Write Multiple Coils	Writes a number of bits into "CoilData" starting from an initial address
16	Write Multiple Registers	Writes a number of registers into "HoldingRegisterData" starting from an initial address
23	Read/Write Multiple Registers	Executes a combined read / write operation on "HoldingRegisterData"
43/14	Read Device Identification	Reads the default device information of the Modbus TCP server: • VendorName:"Bosch Rexroth" • ProductCode: "IL_ModbusTCPServerType02" • MajorMinorRevision: "01Vxx"

Fig.1-12: Supported Modbus function codes

☞ The function codes 1, 5, 15 access a memory area assigned at the "CoilData" input.

☞ The function code 2 accesses a memory area assigned at the "InputData" input.

☞ The function codes 3, 6, 16 access a memory area assigned at the "HoldingRegisterData" input.

☞ The function code 4 accesses a memory area assigned at the "InputRegisterData" input.

Troubleshooting

The FB uses the F_RELATED_TABLE, 16#170x error table. It can generate the following error messages in Additional1 and Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR (16#0001)	16#00001800	16#00000000	The "CoilData" input is a zero pointer.
		16#00000001	The "InputData" input is a zero pointer.

Modbus TCP

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR (16#0001)	16#00001801	16#00000000	The "HoldingRegisterData" input is a zero pointer.
		16#00000001	The "InputRegisterData" input is a zero pointer.
INPUT_INVALID_ERROR (16#0001)	16#00001802	16#00000000	The "SizeOfCoilData" input is 0.
		16#00000001	The "SizeOfInputData" input is 0.
INPUT_INVALID_ERROR (16#0001)	16#00001803	16#00000000	The "SizeOfHoldingRegisterData" input is 0.
		16#00000001	The "SizeOfInputRegisterData" input is 0.
INPUT_INVALID_ERROR (16#0001)	16#00001804	16#00000000	The "SizeOfHoldingRegisterData" input is odd.
		16#00000001	The "SizeOfInputRegisterData" input is odd.
RESOURCE_ERROR (16#0003)	16#00001805	16#00000000	Could not open socket.
RESOURCE_ERROR (16#0003)	16#00001806	16#00000000	Could not accept incoming connection.
RESOURCE_ERROR (16#0003)	16#00001807	16#00000000	The port is assigned.
RESOURCE_ERROR (16#0003)	16#00001808	16#00000000	Port listen failed.

Fig.1-13: IL_ModbusTCPType02 error codes

1.4 Example Applications

1.4.1 General

It is useful for the configuration to execute first tests with the Modbus test clients. Test client examples available in the internet for free (date 09.2008):

- IPC@CHIP Modbus Client V1.4
- Modbus TCP/IP Request Tool Pro

The following chapters describe the configuration of a TCP connection with

- **Siemens WinCC-flexible 2007** (chapter 1.4.2 "Configuring with Siemens WinCC-flexible 2007" on page 12) and
- **Pro-face GP-Pro EX2.2** (chapter 1.4.3 "Configuring using Pro-face GP-Pro EX 2.2" on page 15) in detail.

Addressing data from a PLC project

In the PLC project, data ranges for the register and bit data are transferred to the FB instance IL_ModbusTCPServer using the pointer.

The data ranges of the Modbus TCP server are accessed via the following addressing by the Modbus client

Register data range:

Addressing the "Holding Register" data using **%MW** (read and write access) or the "Input Register" data using **%IW** (only read access).

Bit data range:

Addressing the "Coil" data using **%M** (read and write access) or the "Discrete Input" data using **%I** (only read access).

Bit data

In the test project, the bit data range was assigned as array of bytes from where every single bit can be addressed continuously.

☞ No ARRAY OF BOOL can be used for the bit data area since 8 bits of memory are assigned to a variable of BOOL in the IndraLogic.

The individual bits of a byte can be copied on variables of type BOOL (and vice versa)

The address of a variable from a bit data area corresponds to the bit number from the byte array used for bit data.

Example:

Bit data (%M)

%M0 address corresponds to the **BitData[0].0** variable

%M7 address corresponds to the **BitData[0]0.7** variable

%M8 address corresponds to the **BitData[1].0** variable

%M12 address corresponds to the **BitData[1].4** variable

Register data

The register data range can consist of up to 65536 words and can be organized by the user without any restrictions.

For a better overview on the register data, a structure containing arrays of different PLC data types was used for the register data area in the test project. This structure was created in such a way that the individual data type areas are always provided with a length of 1000 words and do therefore begin with a defined word address.

Example of a data structure for RegisterData

Program:

```
TYPE REGISTER_DATA :
STRUCT
  arWord:   ARRAY[0..999]OF WORD;     (* %MW0000 - %MW0999 *)
  arInt:    ARRAY[0..999]OF INT;      (* %MW1000 - %MW1999 *)
  arUint:   ARRAY[0..999]OF UINT;     (* %MW2000 - %MW2999 *)
  arDint:   ARRAY[0..499]OF DINT;     (* %MW3000 - %MW3999 *)
  arUdint:  ARRAY[0..499]OF UDINT;    (* %MW4000 - %MW4999 *)
  arReal:   ARRAY[0..499]OF REAL;     (* %MW5000 - %MW5999 *)
  arString: ARRAY[0..99]OF STRING(19);(* %MW6000 - %MW6999 *)
(* String(19): because 19 byte string + 1 byte end sign  *)
(* together 7000 WORDS *)
END_STRUCT
END_TYPE
```

Text string

Texts (strings) can also be transferred via the register data area.

☞ In the IndraLogic, the length of a string (in byte) is always the given length + 1 byte of end ID

To define the strings with their total word length, the length is always to be specified with an odd number of bits.

(e.g. string(19) => 19 byte string + 1 byte End ID = 20 byte = 10 words)

The address of a variable from the register data range results from the design of the structure used for the register data.

The following results if the previously shown "Example of a data structure for RegisterData" on page 11 is taken into consideration:

Modbus TCP

Example:

Register data (%MW)

%MW0 address corresponds to the **RegisterData.arWord[0]** variable
%MW1000 address corresponds to the **RegisterData.arInt[0]** variable
%MW1001 address corresponds to the **RegisterData.arInt[1]** variable
%MW5000 address corresponds to the **RegisterData.arReal[0]** variable
%MW5002 address corresponds to the **RegisterData.arReal[1]** variable
%MW6000 address corresponds to the **RegisterData.arString[0]** variable
%MW6010 address corresponds to the **RegisterData.arString[1]** variable

1.4.2 Configuring with Siemens WinCC-flexible 2007

Communication settings

To configure the connection in WinCC-flexible, the following settings are required:

1. Open **Communication** ▸ **Connections** in the project tree menu of the operating device
2. Enter the name of the connection (comment is optional)
   Select the communication driver **"Modicon MODBUS TCP/IP"**
3. Subsequently, the communication parameters are to be defined in the dialog

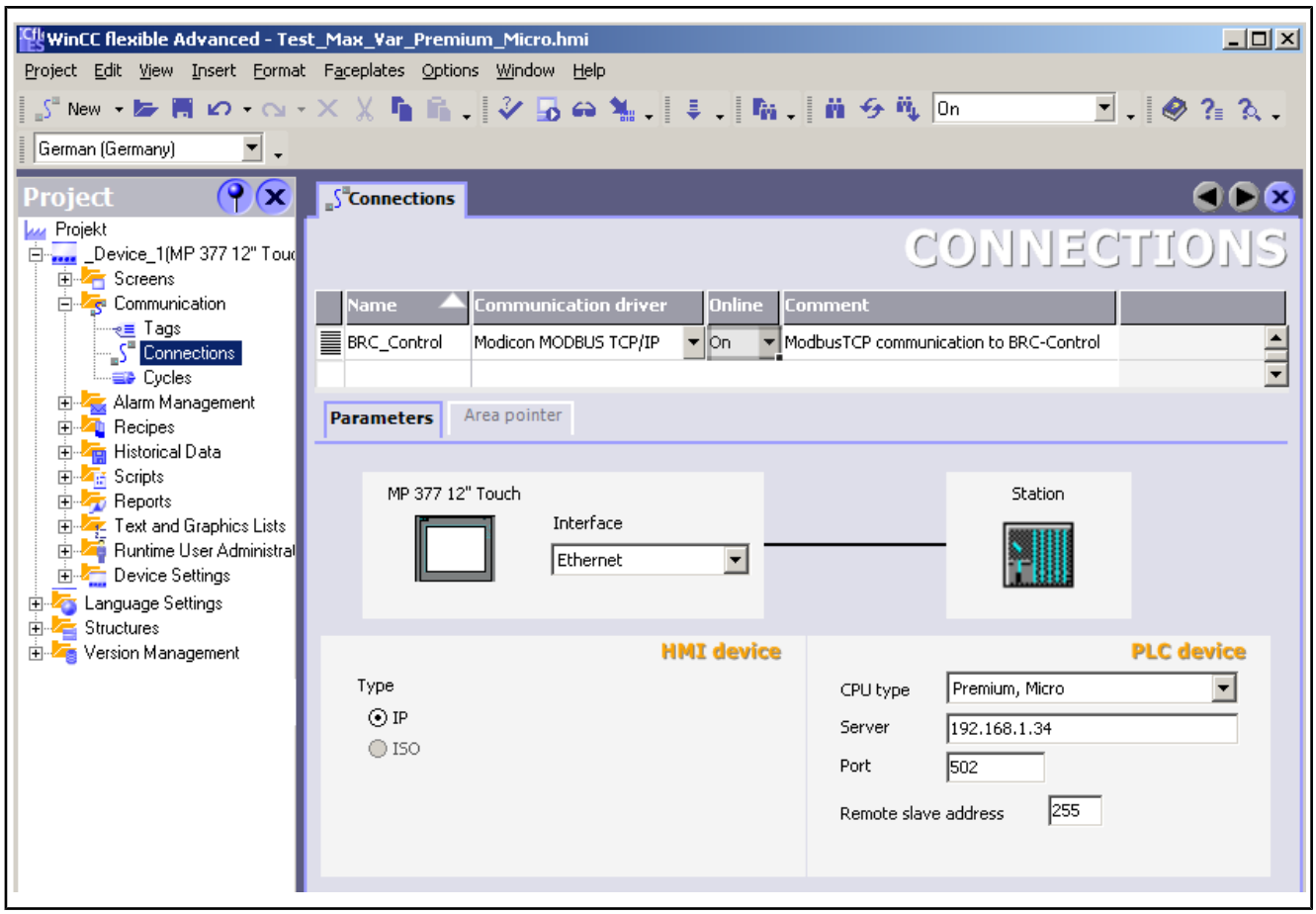

Fig. 1-14: WinCC-flexible Modbus TCP communication parameters

Section	Name	Value	Description
HMI device	Interface	Ethernet	Selection of communication interface
	Type	IP	Only IP is supported
PLC device	CPU type	Premium, Micro	Schneider Modicon control type operating without address offset and normal bit counting (refer to the Siemens user manual "WinCC-flexible 2007 Communication Volume 2 - Chapter 6"
	Server	"192.168.1.34"	IP address of the control
	Port	"502"	Port of the communication connection (port of the IL_ModbusTCPServer FB instance)
	Remote slave address	"255"	Maintain default setting (255) ! not relevant for the Modbus TCP - refer to "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" on www.Modbus-IDA.org

Fig. 1-15: WinCC-flexible Modbus TCP communication parameters

Variable declaration

After configuring the connection, the variables can be declared.

Therefore, the **Communication / Tags** menu is to be opened in the project tree of the operating device. Subsequently, a dialog appears and the variables required can be assigned.

Example of a variable declaration in WinCC-flexible 2007:

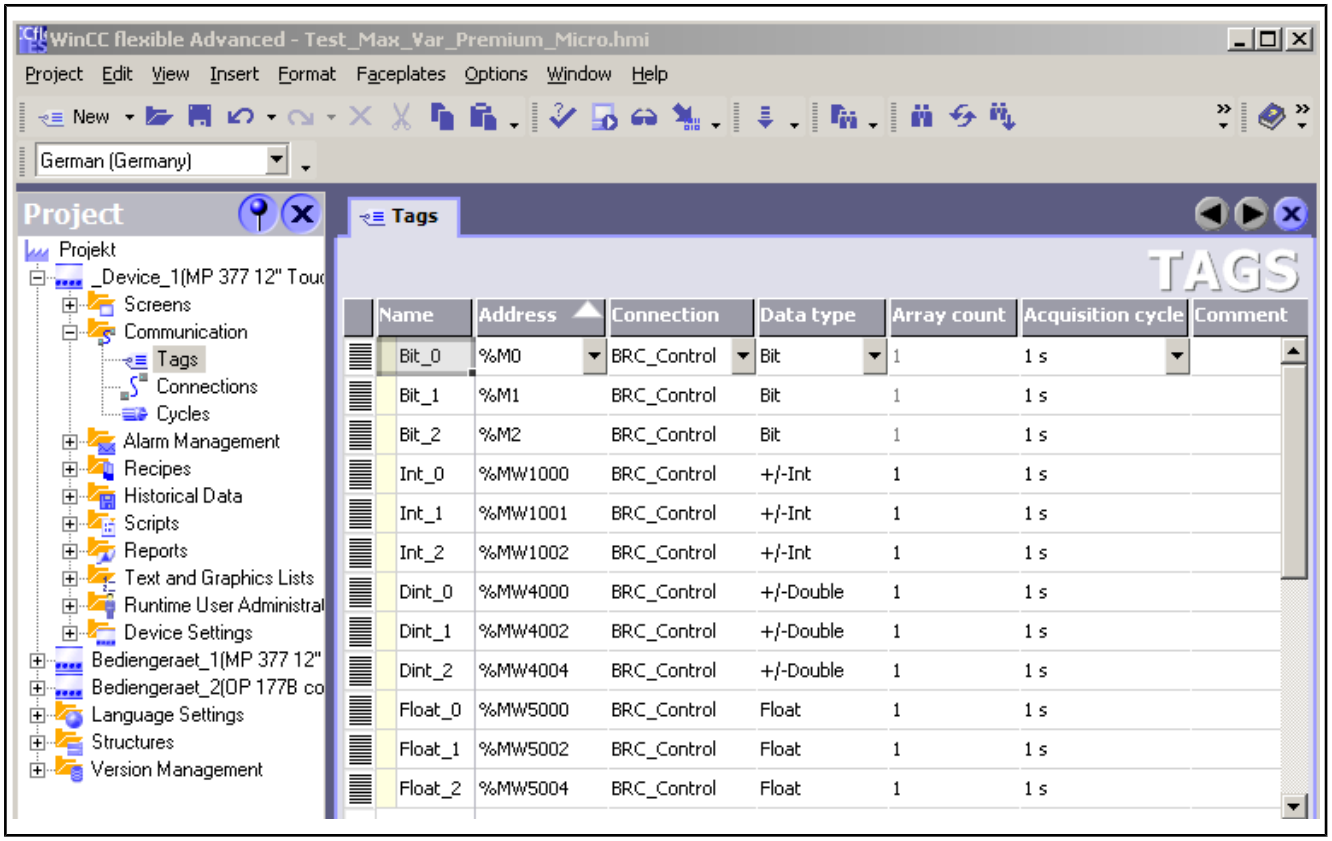

Fig. 1-16: WinCC-flexible variable declaration - Example

WinCC - flexible variable declaration

Input field	Example	Description
Name	"Bit_0"	Variable name
Address	"%M0"	Modbus address of the variable ("Addressing data from a PLC project" on page 10)
Connection	"BRC_Control"	Connection name
Data Type	"Bit"	WinCC-flexible variable data type (fig. 1-18 "WinCC-flexible data types / IndraLogic data types" on page 14)
Array count	"1"	Number of array elements if variable is to be assigned as array
Acquisition cycle	"1s"	Update rate in which the variable values are to be read from the control

Fig.1-17: WinCC - flexible variable declaration

In WinCC-flexible 2007, the following data types are available. The corresponding IndraLogic data types are listed below.

WinCC - flexible data types

WinCC - flexible	IndraLogic	Description
Int	UINT	Unsigned integer length 16 bit Value range: 0 to 65535
+/-int	INT	Signed integer length 16 bit Value range: -32768 to 32767
Double	UDINT	Unsigned double integer length 32 bit Value range: 0 to 4294967295
+/-double	DINT	Signed double integer length 32 bit Value range: -2147483648 to 2147483647
Float	REAL	Floating point length 32 bit Value range: 1.175494351e-38F to 3.402823466e+38F
Bit	Single bit of a byte	Single bit length 1 bit ("Bit data" on page 11)
16 bit group	WORD	WORD length 16 bits **Note:** For 16 bit groups, the bit counting in WinCC-flexible is vice versa to the one in IndraLogic
ASCII	STRING	The string length (in byte) is to be configured in WinCC-flexible via the Tag Properties. ("Text string" on page 11)

Fig.1-18: WinCC-flexible data types / IndraLogic data types

The declared variables can now be used for different WinCC-flexible functionalities.

Example:

Displaying and entering different variables

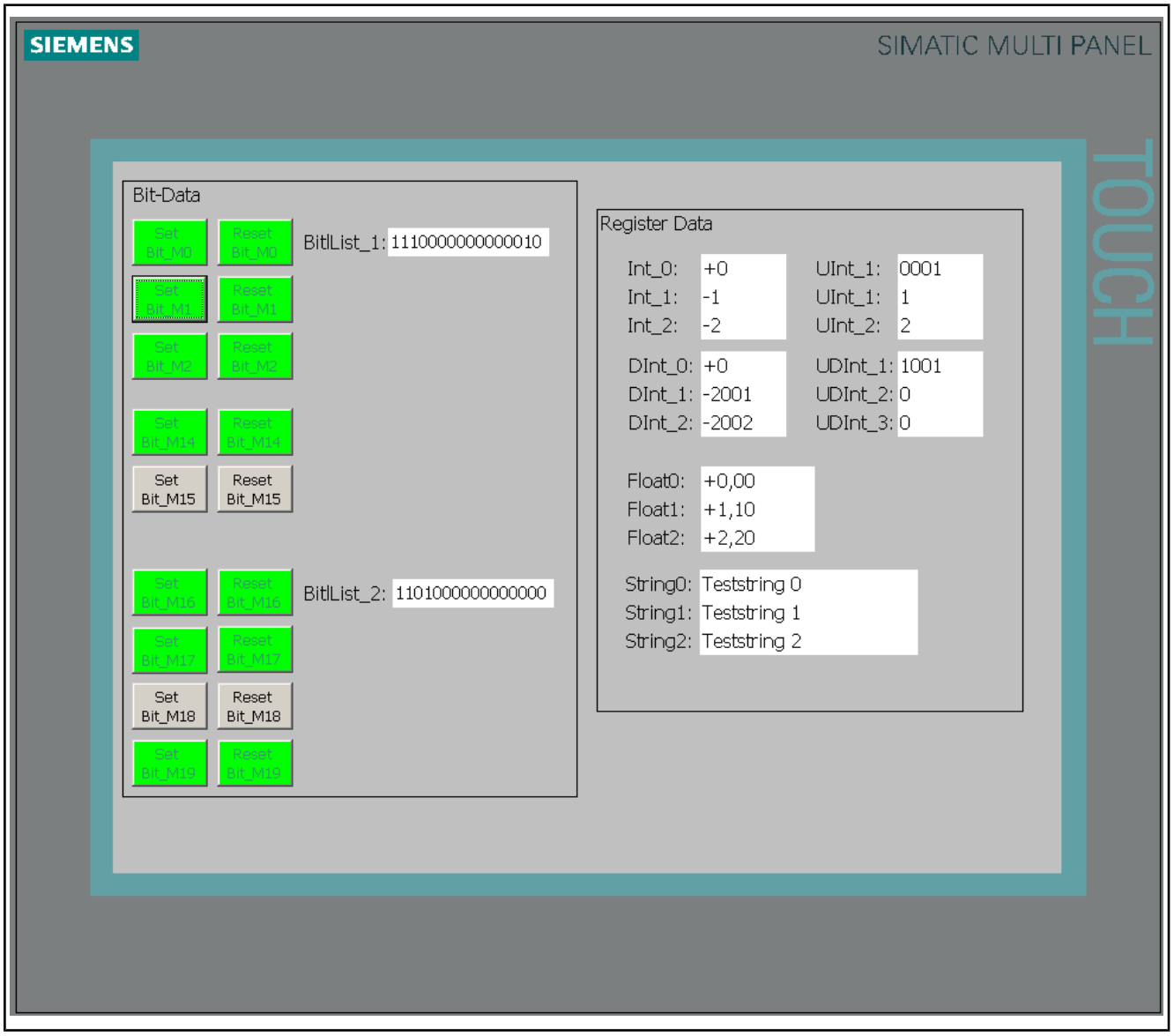

Fig. 1-19: WinCC-flexible - Visualization example

1.4.3 Configuring using Pro-face GP-Pro EX 2.2

Communication settings

The following settings are required to configure the connection in Pro-face GP-Pro EX 2.2:

1. Select the **"Device/PLC"** entry in the **System Settings** menu.
2. Execute the **"Add Device/PLC"** link in the working area of the window
3. Subsequently, the following settings are to be carried out in the "Add Device/PLC" dialog:

Modbus TCP

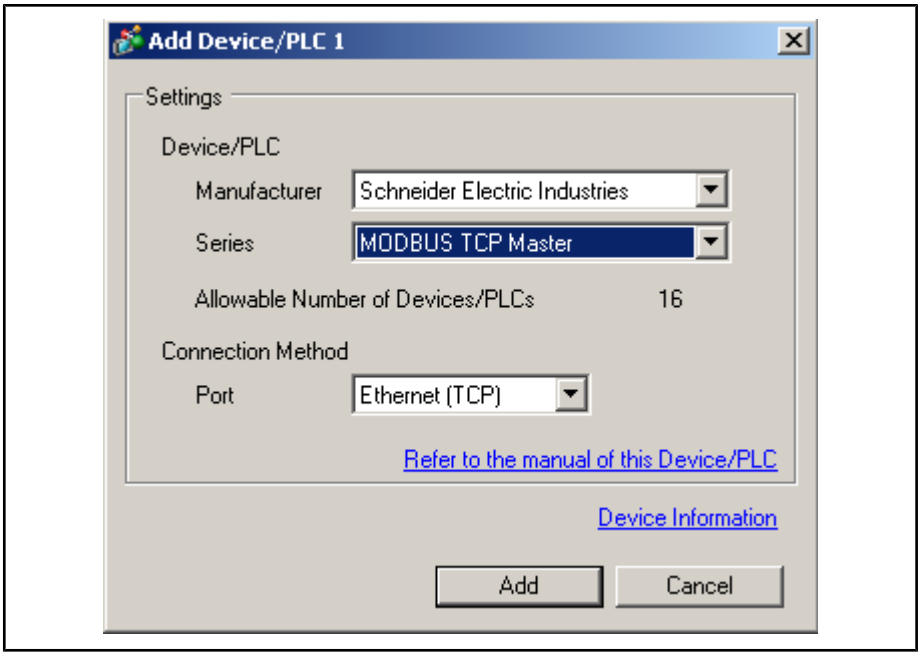

Fig. 1-20: GP-Pro EX communication settings - Device selection

Drop-down menu	Selection
Manufacturer	Schneider Electric Industries
Series	MOCBUS TCP master
Port	Ethernet (TCP)

Fig. 1-21: Communication settings - Device selection

Confirm selection with <Add>

4. Enter the connection name in the extended window of the working range under "Device Name" and click on **"Settings"**.

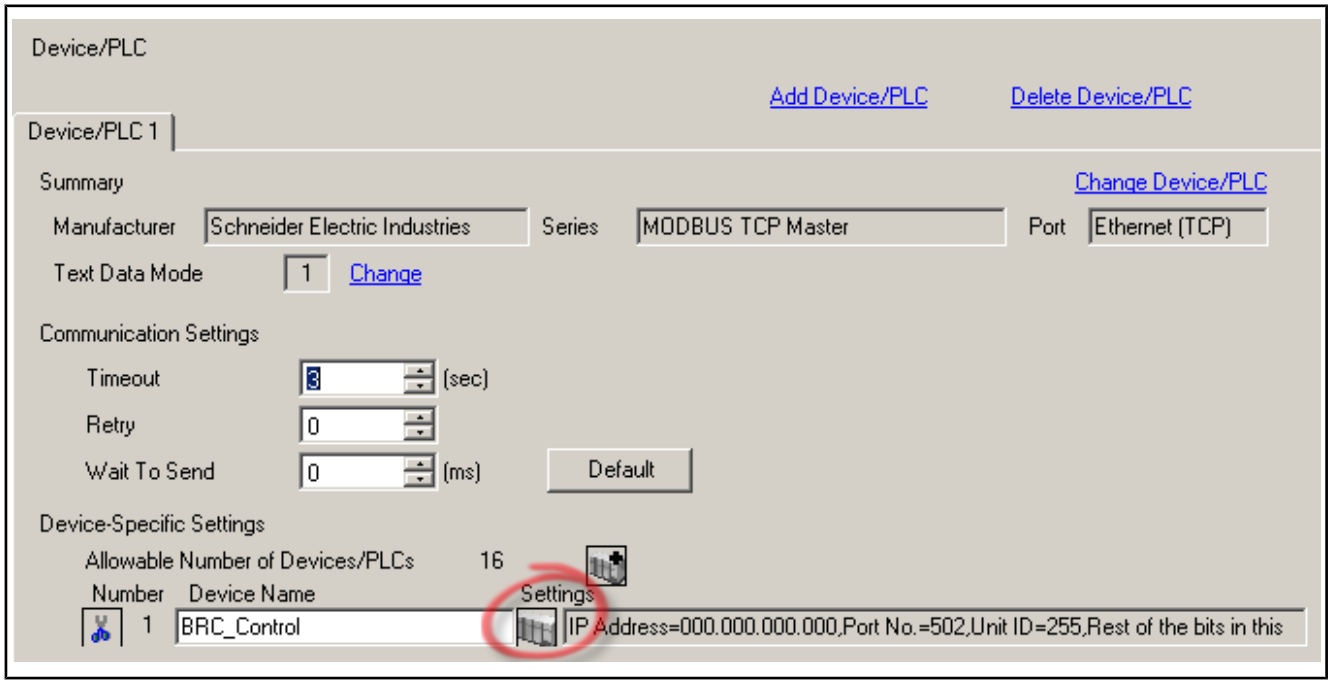

Fig. 1-22: GP-Pro EX communication settings - Working range

5. Subsequently, the following settings are to be carried out in the "Equipment Configuration" register in the "Individual Device Settings" dialog:

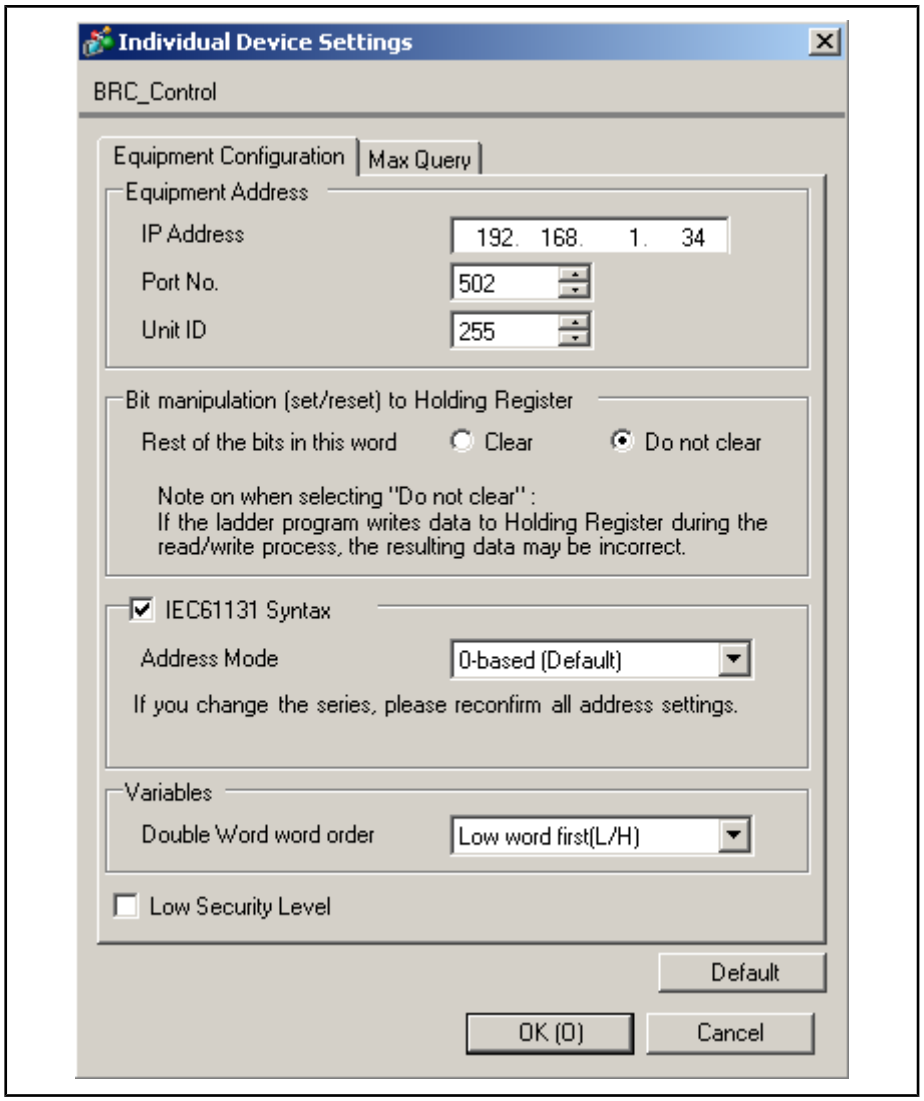

Fig.1-23: GP-Pro EX communication settings - Device parameters

Section	Name	Value	Description
Equipment Address	IP Address	"192.168.1.34"	IP address of the control
	Type	"502"	Port of the communication connection (port of the IL_ModbusTCPServer FB instance)
	Unit ID	"255"	Maintain default setting (255) ! not relevant for the Modbus TCP - refer to "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" on www.Modbus-IDA.org
Bit manipulation to Holding Register	Reset of the bits in this word	"Do not clear"	Maintain default setting (Do not clear)

Modbus TCP

Section	Name	Value	Description
IEC61131 Syntax	Address Mode	"0-based (default)"	Activate IEC61131 Syntax and maintain Address Mode for the default setting (0-based)
Variables	Double Word word order	"Low word first (L/H)"	Word order within a double word

Fig.1-24: GP-Pro EX communication settings - Device parameters

The "MaxQuery" register defines the maximum number of data that can be packed in the Modbus request. The default settings remain.

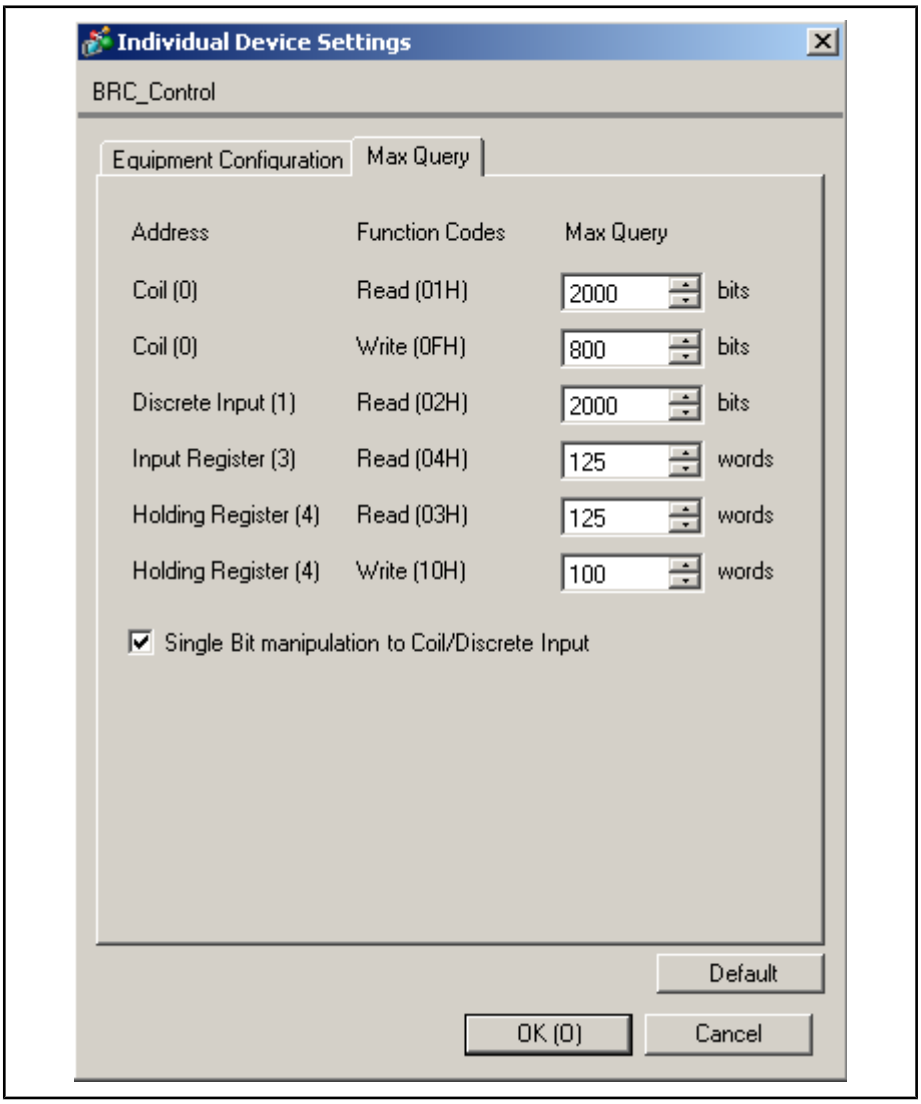

Fig.1-25: GP-Pro EX communication settings - Device parameters

Confirm selection with <OK>

6. The "Text Data Mode" is still to be adapted to allow interpretation of texts. Therefore, click on the **"Change"** link in the working area window.

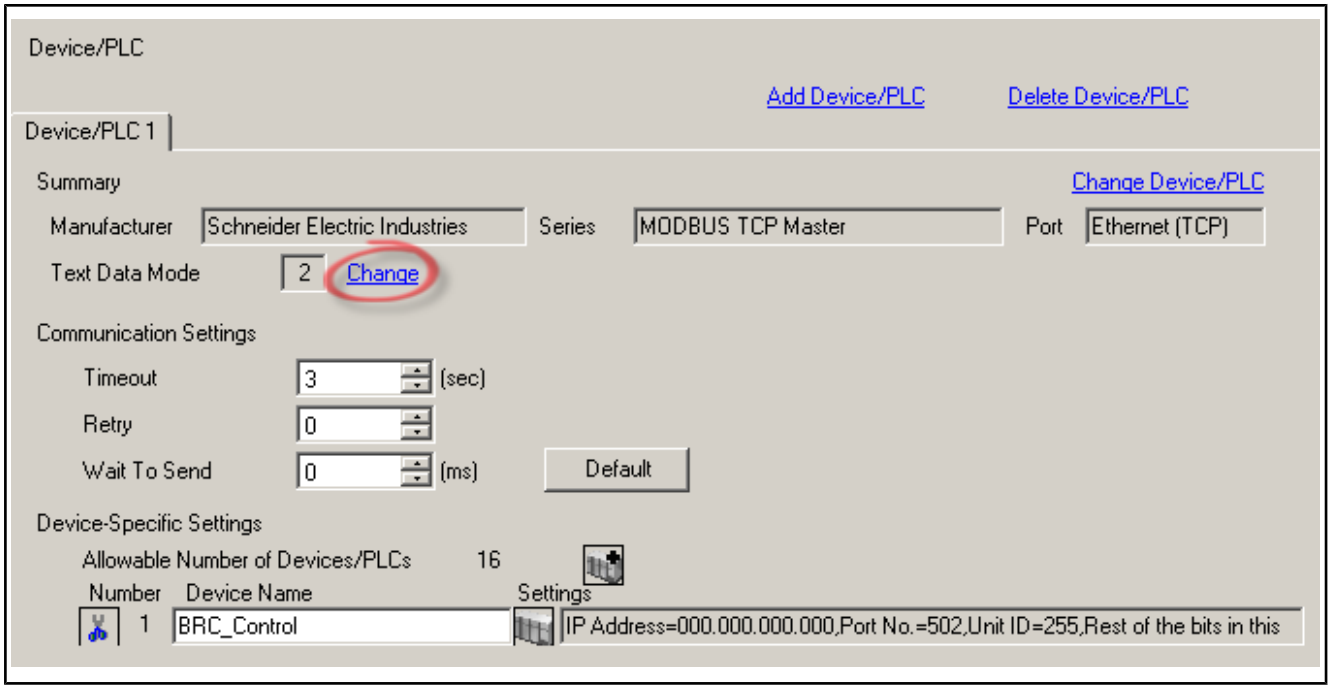

Fig. 1-26: GP-Pro EX communication settings - Working range

Subsequently, select 2 in the "Change Text Data Mode" dialog and confirm with the <Change> button.

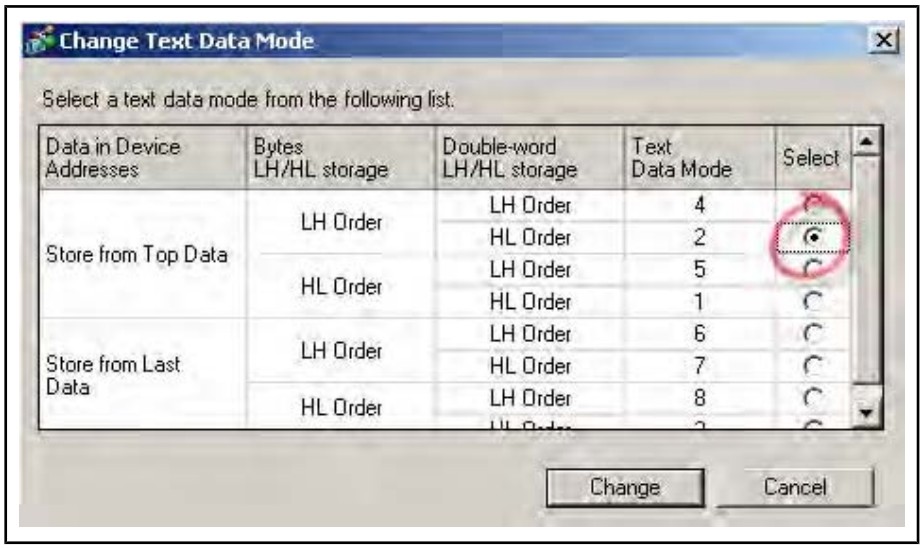

Fig. 1-27: GP-Pro EX communication settings - Text data mode

Using addresses

After configuring the connection, the addresses can be used directly in different GP-Pro EX functionalities, refer to "Addressing data from a PLC project" on page 10.

Modbus TCP

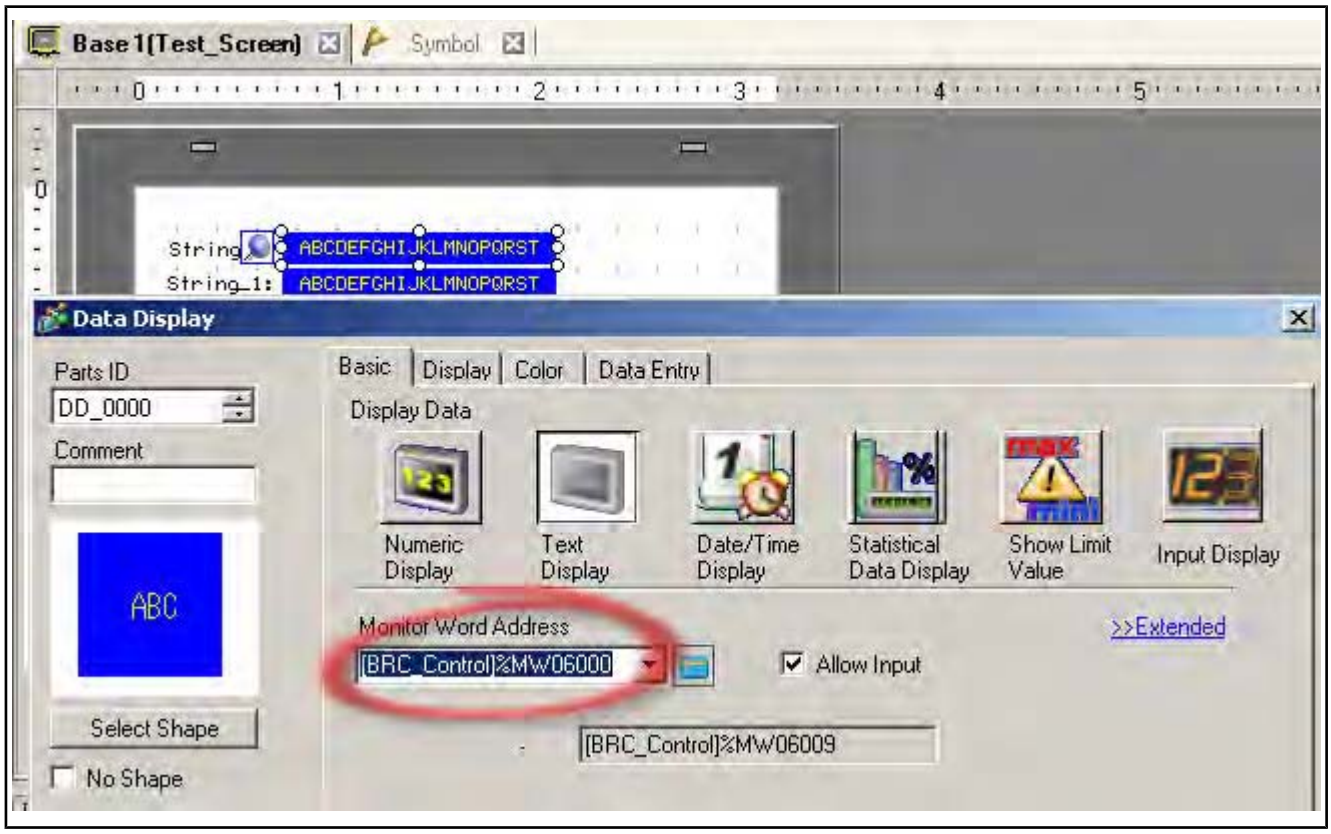

Fig.1-28: GP-Pro EX - Direct address use

Variable declaration

Symbol variables of type bit address and word address can also be declared. For details, please refer to "Addressing data from a PLC project" on page 10. The address content can be used in different GP-Pro EX functionalities via these symbol variables.

To declare the symbol variables in the working area of the window, open **Common Settings ▸ Symbol Variables** in the menu.

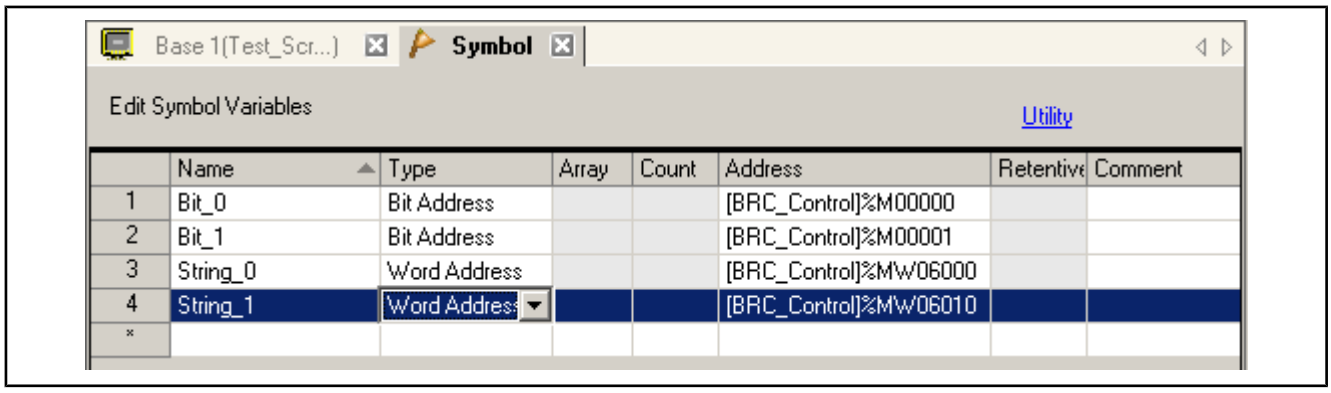

Fig.1-29: GP-Pro EX - Create a symbol variable

After creating the symbol variable, it can be used as "placeholder" for the address.

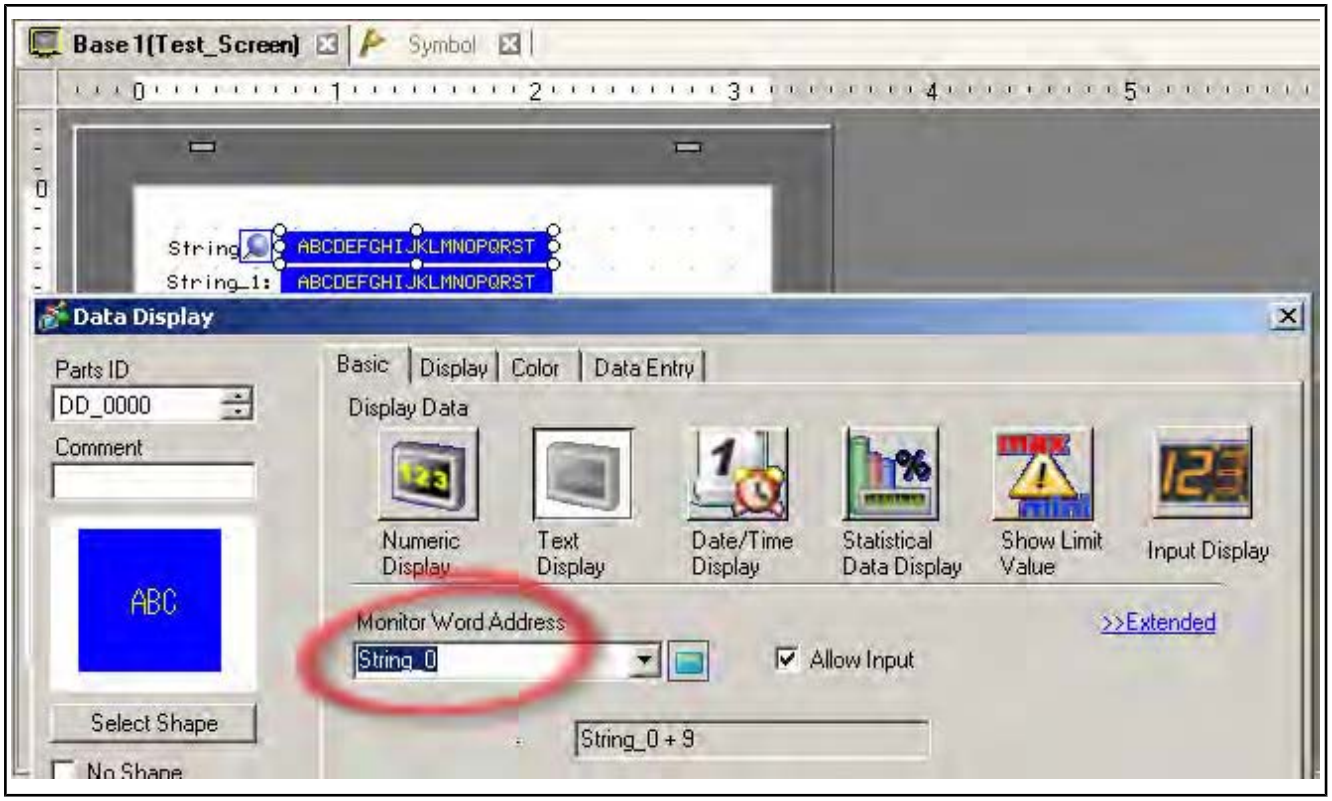

Fig. 1-30: GP-Pro EX - Using symbol variables

The number of the registers to be interpreted is defined via the data type used in the individual display fields.

For display fields of type "Text Display", the number of characters to be displayed are defined via the number of the registers to be interpreted.

Example:

Displaying and entering different variables

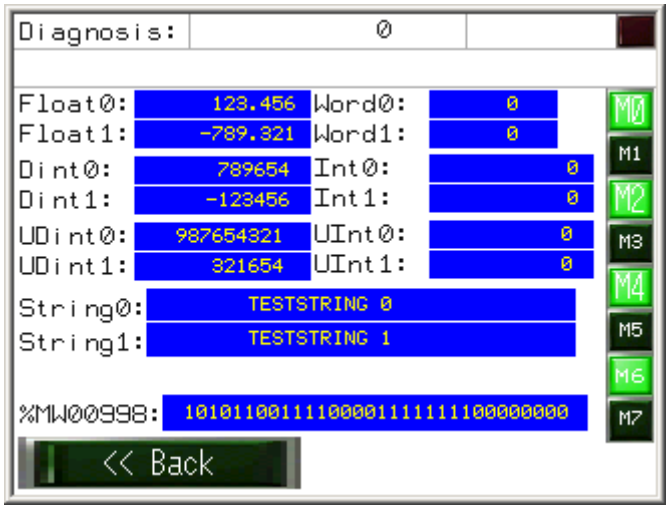

Fig. 1-31:

2 Service and Support

Our service helpdesk at out headquarters in Lohr, Germany, will assist you with all kinds of enquiries. Out of helpdesk hours please contact our German service department directly.

	Helpdesk	Service Hotline Germany	Service Hotline Worldwide
Time [1]	Mo-Fr 7:00 am - 6:00 pm CET	Mo-Fr 6:00 pm - 7:00 am CET Sa-Su 0:00 am - 12:00 pm CET	Outwith Germany please contact our sales/service office in your area first. For hotline numbers refer to the sales office addresses on the Internet.
Phone	+49 (0) 9352 40 50 60	+49 (0) 171 333 88 26 or +49 (0) 172 660 04 06	
Fax	+49 (0) 9352 40 49 41	–	
e-mail	service.svc@boschrexroth.de	–	
Internet	http://www.boschrexroth.com You will also find additional notes regarding service, maintenance (e.g. delivery addresses) and training.		

1) Central European Time (CET)

Preparing Information

For quick and efficient help please have the following information ready:

- detailed description of the fault and the circumstances
- information on the type plate of the affected products, especially type codes and serial numbers
- your phone, fax numbers and e-mail address so we can contact you in case of questions.

Index

Notes

Rexroth
Bosch Group

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Phone +49 (0)93 52-40-50 60
Fax +49 (0)93 52-40-49 41
service.svc@boschrexroth.de
www.boschrexroth.com

R911326406

Printed in Germany
DOK-IL*1G*-MOD*TCP*V04-AW02-EN-P