Rexroth
Bosch Group

Rexroth VCP-Operating Concept

R911310666
Edition 02

Application Manual

Title	Rexroth VCP-Operating Concept
Type of Documentation	Application Manual
Document Typecode	DOK-SUPPL*-VIC*BED*02*-AW02-EN-P
Internal File Reference	Document number, 120-2100-B380-01/EN
Purpose of Documentation	This document serves to describe the possible uses of small operator terminals of the VCP series.

Record of Revisions

Description	Release Date	Notes
120-2100-B380-01/EN	05.2005	First edition
120-2100-B380-02/EN	05.2006	Second edition

Published by Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2
D-97816 Lohr a. Main
Tel.: +49 (0) 93 52 / 40-0
Fax: +49 (0) 93 52 /40-45 85
http://www.boschrexroth.com/
Abt.: BRC/EPY (NH)

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

Contents

1 Important Notes

1.1 Symbols

The symbols in this document are used to draw your attention on notes and dangers.

1.1.1 General Symbols

⚠	**Danger** This symbol is used to refer to instructions which, if ignored or not carefully followed could result in personal injury.
☞	**Note** This symbol indicates application tips or supplementary notes.

1.2 Target Group

All configuration and programming work in connection with the automation system must be performed by trained personnel only (e.g. qualified electricians, electrical engineers)

The configuration and programming personnel must be familiar with the safety concepts of automation technology.

2 Important Directions for Use

2.1 Appropriate Use

2.1.1 Introduction

Rexroth products represent state-of-the-art developments and manufacturing. They are tested prior to delivery to ensure operating safety and reliability.

The products may only be used in the manner that is defined as appropriate. If they are used in an inappropriate manner, then situations can develop that may lead to property damage or injury to personnel.

Bosch Rexroth, as manufacturer, is not liable for any damages resulting from inappropriate use. In such cases, the guarantee and the right to payment of damages resulting from inappropriate use are forfeited. The user alone carries all responsibility of the risks.

Before using Rexroth products, make sure that all the pre-requisites for appropriate use of the products are satisfied:

- Personnel that in any way, shape or form uses our products must first read and understand the relevant safety instructions and be familiar with appropriate use.
- If the product takes the form of hardware, then they must remain in their original state, in other words, no structural changes are permitted. It is not permitted to decompile software products or alter source codes.
- Do not mount damaged or faulty products or use them in operation.
- Make sure that the products have been installed in the manner described in the relevant documentation.

2.1.2 Areas of Use and Application

The VCP-operating concept contains hardware and project planning software that allows to operate and control machines and installations and serves to visualize the information about the machine/installation to be operated required by the user.

Operation is only permitted in the specified configurations and combinations of hardware components and with the software and firmware specified in this documentation and in the relevant project planning manuals.

Typical applications are:

- Handling and assembly systems
- Packaging and foodstuff machines
- Printing and paper processing machines
- Machine tools

2.2 Inappropriate Use

Applying the hardware and project planning software used in combination with the VCP operating concept outside of the above-referenced areas of application or under operating conditions other than described in the document and the technical data specified is defined as "inappropriate use".

The hardware and project planning software may not be used, if

- they are subject to operating conditions that do not meet the above specified ambient conditions.
- Bosch Rexroth has not specifically released them for that intended purpose. Please note the specifications outlined in the general Safety Guidelines!

3 Safety Instructions for Electric Drives and Controls

3.1 Introduction

Read these instructions before the initial startup of the equipment in order to eliminate the risk of bodily harm or material damage. Follow these safety instructions at all times.Do not attempt to install or start up this equipment without first reading all documentation provided with the product. Read and understand these safety instructions and all user documentation of the equipment prior to working with the equipment at any time. If you do not have the user documentation for your equipment, contact your local Bosch Rexroth representative to send this documentation immediately to the person or persons responsible for the safe operation of this equipment. If the equipment is resold, rented or transferred or passed on to others, then these safety instructions must be delivered with the equipment.

⚠

WARNING
Improper use of this equipment, failure to follow the safety instructions in this document or tampering with the product, including disabling of safety devices, may result in material damage, bodily harm, electric shock or even death!

3.2 Explanations

The safety instructions describe the following degrees of hazard seriousness in compliance with ANSI Z535. The degree of hazard seriousness informs about the consequences resulting from non-compliance with the safety instructions.

Warning symbol	**Signal word and degree of hazard seriousness according to ANSI**
⚠	**DANGER** Death or severe bodily harm will occur.
⚠	**WARNING** Death or severe bodily harm may occur.
⚠	**CAUTION** Bodily harm or material damage may occur.

Fig. 3-1: Hazard classification (according to ANSI Z535)

3.3 Hazards by Improper Use

⚠ **DANGER**
High voltage and high discharge current! Danger to life or severe bodily harm by electric shock!

⚠ **DANGER**
Dangerous movements! Danger to life, severe bodily harm or material damage by unintentional motor movements!

⚠ **WARNING**
High electrical voltage due to wrong connections! Danger to life or bodily harm by electric shock!

⚠ **WARNING**
Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electrical equipment!

⚠ **CAUTION**
Surface of machine housing could be extremely hot! Danger of injury! Danger of burns!

⚠ **CAUTION**
Risk of injury due to improper handling! Bodily harm caused by crushing, shearing, cutting and mechanical shock or incorrect handling of pressurized systems!

⚠ **CAUTION**
Risk of injury due to incorrect handling of batteries!

3.4 General Information

- Bosch Rexroth AG is not liable for damages resulting from failure to observe the warnings provided in this documentation.
- Read the operating, maintenance and safety instructions in your language before starting up the machine. If you find that you cannot completely understand the documentation for your product, please ask your supplier to clarify.
- Proper and correct transport, storage, assembly and installation as well as care in operation and maintenance are prerequisites for optimal and safe operation of this equipment.
- Only persons who are trained and qualified for the use and operation of the equipment may work on this equipment or within its proximity.

- Furthermore, they must be trained, instructed and qualified to switch electrical circuits and equipment on and off in accordance with technical safety regulations, to ground them and to mark them according to the requirements of safe work practices. They must have adequate safety equipment and be trained in first aid.
- Only use spare parts and accessories approved by the manufacturer.
- Follow all safety regulations and requirements for the specific application as practiced in the country of use.
- The equipment is designed for installation in industrial machinery.
- The ambient conditions given in the product documentation must be observed.
- Use only safety features and applications that are clearly and explicitly approved in the Project Planning Manual.
- For example, the following areas of use are not permitted: construction cranes, elevators used for people or freight, devices and vehicles to transport people, medical applications, refinery plants, transport of hazardous goods, nuclear applications, applications sensitive to high frequency, mining, food processing, control of protection equipment (also in a machine).
- The information given in the documentation of the product with regard to the use of the delivered components contains only examples of applications and suggestions.

The machine and installation manufacturer must

- make sure that the delivered components are suited for his individual application and check the information given in this documentation with regard to the use of the components,
- make sure that his application complies with the applicable safety regulations and standards and carry out the required measures, modifications and complements.
- Startup of the delivered components is only permitted once it is sure that the machine or installation in which they are installed complies with the national regulations, safety specifications and standards of the application.
- Operation is only permitted if the national EMC regulations for the application are met.
- The instructions for installation in accordance with EMC requirements can be found in the documentation "EMC in Drive and Control Systems".
- Technical data, connections and operational conditions are specified in the product documentation and must be followed at all times.

The machine or installation manufacturer is responsible for compliance with the limiting values as prescribed in the national regulations.

- Technical data, connections and operational conditions are specified in the product documentation and must be followed at all times.

3.5 Protection Against Contact with Electrical Parts

This section refers to equipment and drive components with voltages above 50 Volts.

Touching live parts with voltages of 50 Volts and more with bare hands or conductive tools or touching ungrounded housings can be dangerous and cause electric shock. In order to operate electrical equipment, certain parts must unavoidably have dangerous voltages applied to them.

DANGER
High electrical voltage! Danger to life, severe bodily harm by electric shock!

- Only those trained and qualified to work with or on electrical equipment are permitted to operate, maintain or repair this equipment.
- Follow general construction and safety regulations when working on high voltage installations.
- Before switching on power the ground wire must be permanently connected to all electrical units according to the connection diagram.
- Do not operate electrical equipment at any time, even for brief measurements or tests, if the ground wire is not permanently connected to the points of the components provided for this purpose.
- Before working with electrical parts with voltage higher than 50 V, the equipment must be disconnected from the mains voltage or power supply. Make sure the equipment cannot be switched on again unintended.
- The following should be observed with electrical drive and filter components:

Wait five (5) minutes after switching off power to allow capacitors to discharge before beginning to work. Measure the voltage on the capacitors before beginning to work to make sure that the equipment is safe to touch.

- Never touch the electrical connection points of a component while power is turned on.
- Install the covers and guards provided with the equipment properly before switching the equipment on. Prevent contact with live parts at any time.

- A residual-current-operated protective device (RCD) must not be used on electric drives! Indirect contact must be prevented by other means, for example, by an overcurrent protective device.
- Electrical components with exposed live parts and uncovered high voltage terminals must be installed in a protective housing, for example, in a control cabinet.

To be observed with electrical drive and filter components:

DANGER
High electrical voltage on the housing! High leakage current! Danger to life, danger of injury by electric shock!

- Connect the electrical equipment, the housings of all electrical units and motors permanently with the safety conductor at the ground points before power is switched on. Look at the connection diagram. This is even necessary for brief tests.
- Connect the safety conductor of the electrical equipment always permanently and firmly to the supply mains. Leakage current exceeds 3.5 mA in normal operation.
- Use a copper conductor with at least 10 mm^2 cross section over its entire course for this safety conductor connection!
- Prior to startups, even for brief tests, always connect the protective conductor or connect with ground wire. Otherwise, high voltages can occur on the housing that lead to electric shock.

3.6 Protection Against Electric Shock by Protective Low Voltage (PELV)

All connections and terminals with voltages between 0 and 50 Volts on Rexroth products are protective low voltages designed in accordance with international standards on electrical safety.

WARNING
High electrical voltage due to wrong connections! Danger to life, bodily harm by electric shock!

- Only connect equipment, electrical components and cables of the protective low voltage type (PELV = Protective Extra Low Voltage) to all terminals and clamps with voltages of 0 to 50 Volts.
- Only electrical circuits may be connected which are safely isolated against high voltage circuits. Safe isolation is achieved, for example, with an isolating transformer, an opto-electronic coupler or when battery-operated.

3.7 Protection Against Dangerous Movements

Dangerous movements can be caused by faulty control of the connected motors. Some common examples are:

- improper or wrong wiring of cable connections
- incorrect operation of the equipment components
- wrong input of parameters before operation
- malfunction of sensors, encoders and monitoring devices
- defective components
- software or firmware errors

Dangerous movements can occur immediately after equipment is switched on or even after an unspecified time of trouble-free operation.

The monitoring in the drive components will normally be sufficient to avoid faulty operation in the connected drives. Regarding personal safety, especially the danger of bodily injury and material damage, this alone cannot be relied upon to ensure complete safety. Until the integrated monitoring functions become effective, it must be assumed in any case that faulty drive movements will occur. The extent of faulty drive movements depends upon the type of control and the state of operation.

DANGER
Dangerous movements! Danger to life, risk of injury, severe bodily harm or material damage!

- Ensure personal safety by means of qualified and tested higher-level monitoring devices or measures integrated in the installation. Unintended machine motion is possible if monitoring devices are disabled, bypassed or not activated.

Pay attention to unintended machine motion or other malfunction in any mode of operation.

- Keep free and clear of the machine's range of motion and moving parts. Possible measures to prevent people from accidentally entering the machine's range of motion:

– use safety fences
– use safety guards
– use protective coverings
– install light curtains or light barriers

- Fences and coverings must be strong enough to resist maximum possible momentum, especially if there is a possibility of loose parts flying off.
- Mount the emergency stop switch in the immediate reach of the operator. Verify that the emergency stop works before startup. Don't operate the machine if the emergency stop is not working.
- Isolate the drive power connection by means of an emergency stop circuit or use a starting lockout to prevent unintentional start.
- Make sure that the drives are brought to a safe standstill before accessing or entering the danger zone. Safe standstill can be achieved by switching off the power supply contactor or by safe mechanical locking of moving parts.
- Secure vertical axes against falling or dropping after switching off the motor power by, for example:
  - mechanically securing the vertical axes
  - adding an external braking/ arrester/ clamping mechanism
  - ensuring sufficient equilibration of the vertical axes

The standard equipment motor brake or an external brake controlled directly by the drive controller are not sufficient to guarantee personal safety!

- Disconnect electrical power to the equipment using a master switch and secure the switch against reconnection for:
  - maintenance and repair work
  - cleaning of equipment
  - long periods of discontinued equipment use
- Prevent the operation of high-frequency, remote control and radio equipment near electronics circuits and supply leads. If the use of such equipment cannot be avoided, verify the system and the installation for possible malfunctions in all possible positions of normal use before initial startup. If necessary, perform a special electromagnetic compatibility (EMC) test on the installation.

3.8 Protection Against Magnetic and Electromagnetic Fields During Operation and Mounting

Magnetic and electromagnetic fields generated near current-carrying conductors and permanent magnets in motors represent a serious health hazard to persons with heart pacemakers, metal implants and hearing aids.

WARNING
Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electrical equipment!

- Persons with heart pacemakers, hearing aids and metal implants are not permitted to enter the following areas:
  – Areas in which electrical equipment and parts are mounted, being operated or started up.
  – Areas in which parts of motors with permanent magnets are being stored, operated, repaired or mounted.

- If it is necessary for a person with a heart pacemaker to enter such an area, then a doctor must be consulted prior to doing so. Heart pacemakers that are already implanted or will be implanted in the future, have a considerable variation in their electrical noise immunity. Therefore there are no rules with general validity.
- Persons with hearing aids, metal implants or metal pieces must consult a doctor before they enter the areas described above. Otherwise, health hazards will occur.

3.9 Protection Against Contact with Hot Parts

CAUTION
Housing surfaces could be extremely hot! Danger of injury! Danger of burns!

- Do not touch housing surfaces near sources of heat! Danger of burns!
- After switching the equipment off, wait at least ten (10) minutes to allow it to cool down before touching it.
- Do not touch hot parts of the equipment, such as housings with integrated heat sinks and resistors. Danger of burns!

3.10 Protection During Handling and Mounting

Under certain conditions, incorrect handling and mounting of parts and components may cause injuries.

CAUTION
Risk of injury by incorrect handling! Bodily harm caused by crushing, shearing, cutting and mechanical shock!

- Observe general installation and safety instructions with regard to handling and mounting.
- Use appropriate mounting and transport equipment.
- Take precautions to avoid pinching and crushing.
- Use only appropriate tools. If specified by the product documentation, special tools must be used.

- Use lifting devices and tools correctly and safely.
- For safe protection wear appropriate protective clothing, e.g. safety glasses, safety shoes and safety gloves.
- Never stand under suspended loads.
- Clean up liquids from the floor immediately to prevent slipping.

3.11 Battery Safety

Batteries contain reactive chemicals in a solid housing. Inappropriate handling may result in injuries or material damage.

CAUTION
Risk of injury by incorrect handling!

- Do not attempt to reactivate discharged batteries by heating or other methods (danger of explosion and cauterization).
- Never charge non-chargeable batteries (danger of leakage and explosion).
- Never throw batteries into a fire.
- Do not dismantle batteries.
- Do not damage electrical components installed in the equipment.

Be aware of environmental protection and disposal! The batteries contained in the product should be considered as hazardous material for land, air and sea transport in the sense of the legal requirements (danger of explosion). Dispose batteries separately from other waste. Observe the legal requirements in the country of installation.

3.12 Protection Against Pressurized Systems

Certain motors and drive controllers, corresponding to the information in the respective Project Planning Manual, must be provided with pressurized media, such as compressed air, hydraulic oil, cooling fluid and cooling lubricant supplied by external systems. Incorrect handling of the supply and connections of pressurized systems can lead to injuries or accidents. In these cases, improper handling of external supply systems, supply lines or connections can cause injuries or material damage.

CAUTION
Danger of injury by incorrect handling of pressurized systems!

- Do not attempt to disassemble, to open or to cut a pressurized system (danger of explosion).
- Observe the operation instructions of the respective manufacturer.
- Before disassembling pressurized systems, release pressure and drain off the fluid or gas.
- Use suitable protective clothing (for example safety glasses, safety shoes and safety gloves)
- Remove any fluid that has leaked out onto the floor immediately.

Environmental protection and disposal! The media used in the operation of the pressurized system equipment may not be environmentally compatible. Media that are damaging the environment must be disposed separately from normal waste. Observe the legal requirements in the country of installation.

4 Application Description for Small Operator Terminals

4.1 The Concept

A uniform functionality and operating structure constitute key aspects of the operating and monitoring concept across the entire product family.

The small operator terminal of construction type VCP relieves the controller completely of operating and monitoring tasks.

In this context, the small operator terminal reads all required data independently from the controller, and processes this further internally. You can use scripts to extend the internal processing as needed. On request, the device writes data or data sets (of recipes) to the controller. The device independently controls the display and the status LEDs.

All small operator terminals of construction type VCP are programmed in the same way using the programming software VI Composer.

4.1.1 Uniform device features

All small operator terminals of the VCP construction type are equipped with:

– Displays with temperature-compensated contrast or brightness control
– Flash memory
– Buffered RAM
– Real-time clock
– Watchdog timer
– Lithium battery with voltage monitoring
– USB ports for memory stick
– Standard or field bus interfaces for communication with the controller

All small operator terminals with a keyboard are additionally equipped with:

– Editing keys
– Control keys
– Function keys with status LEDs
– Slide-in identification strips for the function keys

The operating system of all small operator terminals offers:

– Application ID
– Multilingual applications

- Option to customize the interface parameters
- Automatic error correction
- Softkey functionality for all function keys
- A help system for screens and variables
- Password protection function for screens and variables
- Scaling of variable values
- Dynamic attributes for texts and variables
- A message system for status messages
- A message system for error messages
- Recipe data management function
- Print logs
- Running time meters
- System variables for internal functions.
- Use of any Windows fonts
- Display of images
- Display of sets of curves.

Small operator terminals with a touch screen additionally offer:

- Full-graphics user interface including buttons
- A keyboard that is shown automatically when a variable is selected

4.2 Programming Small Operator Terminals

You program the small operator terminals of construction type VCP in the same way using the programming software VI Composer.

For this purpose, install the programming software on your PC.

– Start the software and select the corresponding entries for the device type and the desired communication protocol.
– Create all of the components of the project, consisting of languages and a controller.
– Compile the project and load the resulting terminal file via Ethernet into the small operator terminal.
– Connect the operating device with the controller.

4.2.1 Hardware Prerequisites

To carry out the installation, you need a basic knowledge of Microsoft Windows. This information is not provided here. If you have any queries in this regard, refer to the Microsoft Windows manuals or online help.

Hardware Requirements

Your computer should fulfill the following hardware and software requirements, to run the programming software VI Composer:

– Pentium processor with 2 GHz
– 512 Mbyte working memory (RAM)
– 200 Mbyte free hard disk memory
– CD ROM drive
– USB port
– Mouse

4.2.2 Installing VI Composer

The installation of the programing software VI Composer includes all required directories, files and entries in Windows.

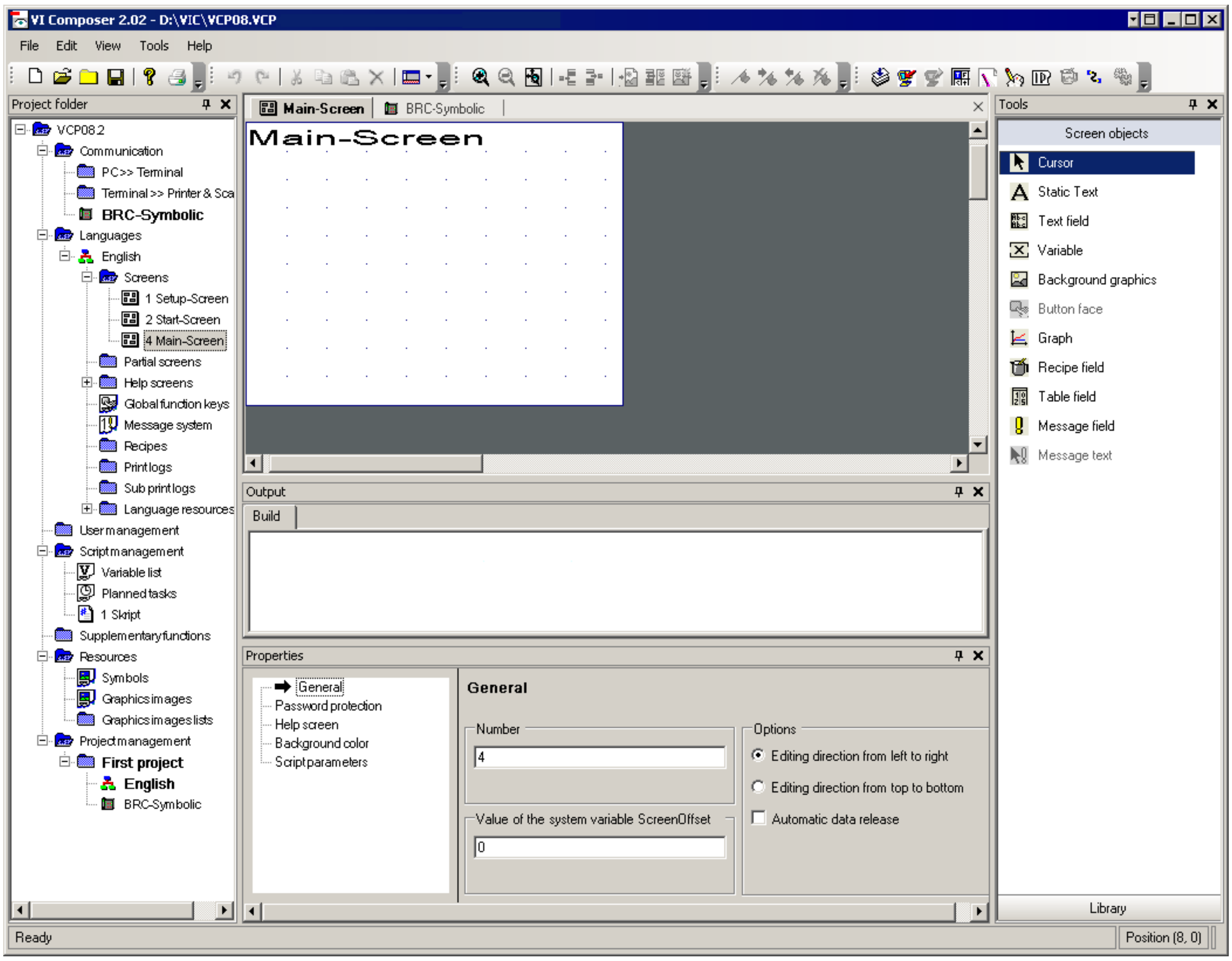
VI Composer runs under the operating systems Windows 2000 and Windows XP.

To install VI Composer, insert the installation CD, select the desired language and start the installation process. Follow the instructions in the installation dialog box.

5 Programming

5.1 Programming Interface

When all of the windows are open, the programming interface looks roughly like this:



General

Number: 4

Value of the system variable ScreenOffset: 0

Options:
- [✓ Editing direction from left to right]
- Editing direction from top to bottom
- Automatic data release

Position (8, 0)

Fig. 5-1: Complete overview of VI-Composer

The interface consists of windows that you can dock at a required location (dockable windows) or position anywhere you like (floating windows). The remaining free space is used as the working area. In this working area, the programming software displays screen, list, recipe and graphic editors that you can use to edit objects (for example, a screen editor for the main screen).

The menu bar and toolbar appear at the top of the main window.

The status bar appears at the bottom of the main window. It displays the particular properties of the elements currently selected in an editor (position and size, for example).

Project Folder window

The **Project folder** window displays the entire project in the form of a tree structure. You can expand any branches within the tree structure that are marked with a plus sign. Additional branches or objects may appear below the branches.

Within the project folder, you can move or copy objects using the drag & drop function (for example, to add a language and a controller to a project).

Properties window:

The **Properties** window always displays the properties of the particular branch or object that is currently selected in the project folder. Any changed properties are transferred directly by the programming software.

Output window:

The **Output** window shows all messages that may be generated during compilation of the terminal file.

Tools window:

You can choose screen objects or library in the tools window.

Screen objects:

To create screen objects, first select a tool from the **Tools** window. The object icon is then displayed on the mouse pointer while you work with a tool.

Library:

You put screen objects into the library to use them again in other projects. To call a library function perform a right-click on the title bar of the library window.

☞ See chapter “Working with Screen Objects“ on page 6-6.
See chapter “Working with Libraries“ on page 6-68.

Hiding windows:

Using the **Autohide** attribute from the context menu, you automatically minimize a window to the size of an icon when it is not in use. This icon is then displayed at the very bottom of the main window. If you move the mouse pointer over the icon, the window reopens in its original location.

Fig. 5-2: Window displayed as icon

To dock a window, follow these steps:

1. Using the mouse, right-click the title bar of the window.
2. Select **Dockable** from the context menu.

Keeping the left-hand mouse button depressed, drag the window to the edge of the main window until the window dimensions match the size of the main window.

Release the mouse button.

To undo the dock operation, select **Floating** from the context menu.

5.2 File Menu

The File menu contains all of the functions you require to create a new project folder, open an existing project folder, save or close the active project folder and quit the programming software package.

5.2.1 File Menu, Open a Project Folder

Select the **Open project folder** menu item to open an existing project folder.

Follow the steps below to open an existing project folder:

1. Select **Open project folder** from the **File** menu.
2. The **Open** dialog appears.
3. Navigate to the project folder required.
4. Select the project folder required.
5. Confirm with **OK**.

The **Open** dialog closes and the selected project folder is opened. In the **Project folder** window, the project folder is entered as a single folder icon with the name of the terminal type.

5.2.2 File Menu, New Project Folder

Select the **New project folder** menu item to create a new project folder. You can start by using an empty or predefined template.

Use predefined templates if you want to save time, especially for programming operating devices equipped with touch screens. When using these templates, all you need to do is adapt the design and implement your own additions.

You can choose to create the new project folder as a project, template or library.

Follow the steps below to create a new project folder:

1. Select **New project folder** from the **File** menu.

The wizard used for creating a new project folder now opens and guides you through a series of dialogs:

- Create project folder, template or library (templates)
- Create project folder, template or library (place to store)
- Create project folder, template or library (terminal type)
- Create project folder or template (protocol type)

Once the wizard has completed all steps, the **Project folder** window contains the new project folder as at least one project icon with the name of the terminal type.

5.2.2.1 Create Project Folder, Template or Library (Templates)

The **Create project folder, template or library (templates)** dialog is the first dialog that the wizard shows you to create a new project folder.

1. Select an **empty template** or a **predefined template**.
2. Select **Project** or **Template** or **Library**.
3. Click **Next**.

5.2.2.2 Create Project Folder, Template or Library (Place to Store)

The **Create project folder, template or library (place to store)** dialog is the second dialog that the wizard shows you to create a new project folder.

Select the directory where you want to save the new project folder. If you do not select a directory, the programming software's root directory is automatically used.

The storage location last entered is automatically offered again.

A list with the last 8 project folders is shown beneath the input field.

Follow the steps below (**variant 1**):

1. Enter a name for the new project folder into the input field (overwrite existing entry, if necessary).

Make sure the file extension is correct!

Template = File name.vcp
Project = File name.vct
Library = File name.vcb

2. Click the **Next >**button.

Follow the steps below (**variant 2**):

1. Click **Select**.
2. Select a directory in the dialog window that appears.
3. Enter a name for the new project folder in the **File name** field.

Make sure the file extension is correct!

Template = File name.vcp
Project = File name.vct
Library = File name.vcb

4. Confirm your actions by clicking the **Save** button.
5. Click the **Next >**button.

Follow the steps below (**variant 3**):

1. Select the name of an existing project folder from the bottom display field.

Make sure the file extension is correct!

Template = File name.vcp
Project = File name.vct
Library = File name.vcb

2. Change the name.
3. Click the Next > button.

You can also return to the previous page. To do so, click the **Back <** button.

5.2.2.3 Create Project Folder, Template or Library (Terminal Type)

The **Create project folder, template or library (terminal type)** dialog is the third dialog that the wizard shows you to create a new project folder.

If you have already created a project folder, the last selection is automatically highlighted.

The icons illustrate the graphical capabilities of the individual devices.

Follow the steps below to specify the terminal type:

1. Select a terminal type from the list.
2. Select the memory size of your device (if possible).
3. Click the **Next >**button.

You can also return to the previous page. To do so, click the **Back <** button.

5.2.2.4 Create Project Folder or Template (Protocol Type)

The **Create project folder or template (templates)** dialog is the fourth and final dialog that the wizard shows you to create a new project folder.

If you have already created a project folder, the last selection is automatically highlighted.

The icons indicate whether this is a field bus connection or a point-to-point connection.

Follow the steps below to specify the protocol type:

1. Select a protocol type from the list.
2. Click the **Finish** button.

You can also return to the previous page. To do so, click the **Back <** button.

5.2.3 File Menu, Save Project Folder

Select the **Save project folder** menu item to save the current project folder. This action overwrites the existing file rather than creating a new one.

5.2.4 File Menu, Close Project Folder

Use **Close project folder** from the menu to complete processing of the current project folder.

If you have made additional changes to the project folder since the last time you saved, the system asks you whether you want to save the project folder before closing.

5.2.5 File Menu, Exit

To quit the programming software, use the Exit menu item.

If a project folder is still open, this is closed first. If you have made changes to the current project folder before closing, the system asks you whether you want to save the project folder.

5.3 View Menu

The view menu offers the following menu items:

- Zoom out
- Zoom in

- Restore workspace
- Close all windows
- Project folder
- Properties
- Tools
- Output

Zoom in:

If an editor is open (i.e. screen editor) the view on the screen is zoomed out by one step each per click on the menu item.

Zoom out:

If an editor is open (i.e. screen editor) the view on the screen is zoomed in by one step each per click on the menu item.

Restore workspace:

Select the menu item **Restore workspace** to get back the previous arrangement, size and number of opened windows.

Close all windows:

Select the menu item **Close all windows** to watch the work space without any other windows.

Project folder:

Opens and closes the **Project folder** window. The check beside the menu item indicates that the window is open.

Properties:

Opens and closes the **Properties window**. The check beside the menu item indicates that the window is open.

Tools:

Opens and closes the **Tools** window. The check beside the menu item indicates that the window is open.

Output:

Opens and closes the **Output** window. The check beside the menu item indicates that the window is open.

5.4 Tools Menu

The **Tools** menu offers you a number of different functions, depending on the particular element within the project tree that is currently selected.

5.4.1 Tools Menu, Options

These are the options you use to define the default settings for the programming software.

You can change these settings at any time. These options are saved together with the project.

5.4.1.1 Options, Project Management

The project management options are divided into the following areas.

- Backup databases
- Current database
- Template directory
- Undo (single-step undo).

Backup Databases area:

Select one of three possible backup variants.

- If you select **None**, no backup is generated when you reopen a database.
- If you select **One** (single backup), a backup file is generated when you reopen a database. The backup file is then overwritten every time this database is reopened. This backup file is always called DATABASENAME.000.
- If you select **Many** (multiple backup), a backup file is generated when you reopen a database. Another backup file is created every time this database is opened. These files are assigned ascending numbered file extensions.

If you wish, you can also store the backup files in compressed form.

Current Database area:

Decide whether you want to compress the current database after exiting it. This will reduce the memory requirement considerably.

Always make sure that you have enough memory available on your hard disk to allow you to unpack a compressed file again.

Template Directory area:

Select the path to be used for the templates. You can then create a new database on the basis of a template.

Undo (single-step undo) area:

Activate the Single-Step Undo check box to cancel the last action.

The undo function does not apply to all actions. To check whether a particular action can be undone, view the status of the button in the tool bar.

You must enable the undo function before opening or creating a file. The undo function can not be reversed while you are programming.

5.4.1.2 Options, Graphic Editors

The graphic editor options are divided into the following areas.

- Capture
- Representation
- Capture / Grid
- Status display.

Capture area:

You can define separate settings for the horizontal and vertical capture function. Elements that are not aligned by means of a grid can be positioned more easily in a screen using the capture function.

You can choose between the following options:

- Deactivate the capture function
- Adopt the standard font (norm font) values for the capture function
- Apply the numerical values to the horizontal and vertical direction.

Capture / Grid area:

In this area you can specify whether

- You want to display a grid in screens,
- The color of the grid.

Status Display area:

Decide whether you want to display the position of the mouse cursor using **Pixel** units or **Grid** units in the status bar. The Grid unit corresponds to the area taken up by one character of the standard font (norm font).

Representation area:

You can assign separate colors to the input and output variables to differentiate between them more clearly. A colored frame is then displayed around the variable.

This area also lets you choose the zoom factor to be used when displaying a screen.

To display **Background images with a frame** in screens (so they are more apparent), select the corresponding check box.

The **Brief info about screen element variable** check box activates a small window with a brief explanation for the element in the screen selected by the mouse. If the element is linked to a controller variable, the full controller address is displayed. If the element is linked to a system variable, the name of the variable is displayed.

To **display subscreens** that are linked, you must select the corresponding check box.

With the **Display referenced image list** check box, you can decide whether to display the default image in selection image variables.

By activating the **Fit working area to editor size** check box, you can ensure that the graphic editor is maximized in the window.

5.4.1.3 Options, List Editors

The list editor options are divided into the following areas.

- Font
- Variable list
- Text and image list
- Default text length for text lists

Font area:

This area displays the font used in the list editors by default.

To change a font, click **Select**. Any existing entries in a list are automatically converted to the selected font.

Variable List area:

To activate an automatic syntax check for the address definition in a variable list, select the appropriate check box in this area. The check is performed after a line of the variable list is exited.

Text and Image List area:

When the **Display values hexadecimally** check box is selected, you can also display and enter the values in text and image lists in hexadecimal format. Always precede the hexadecimal value with the letter H.

Default Text Length for Text Lists area:

This area allows you to determine whether the length of texts entered in text lists is to be unlimited or limited. If you do not want texts to exceed a specific length, enter a maximum length in the **Single character** field.

5.4.1.4 Options, Global Settings

The global settings options are divided into the following areas.

- Representation,
- Options for S3 file generation,
- While starting,
- Open editors,
- When inserting from a library.

Representation area:

To display flashing elements in a screen in the strikethrough format, select the **Display Attribute Flashing** check box.

Options for S3 File Generation area:

You only need this entry field if a project cannot be compiled without errors and you call on help from our hotline. In this case you will get option codes with whose help you will find out further information about the compilation error.

While starting area:

Check the checkbox if the previous opened file should be loaded automatically.

Open editors area:

Since now you had to open an editor by a double-click onto the item of the project folder. Now you only need to perform a single click as default. To open the editors with an double-click again, uncheck the checkbox **with single click**.

When Inserting from a Library area:

Determine the steps the programming software should take if there is already an element with a particular name in the project and you want to add an element with the same name from a library.

Choose one of the options below:

1. The element from the library is assigned a different name when it is added to the project.
2. The element from the library is not added.
3. A prompt is displayed. This must be confirmed before you can add the element.

Regarding 1: The date and time are appended to the name of the element being added. This clearly differentiates the element from the existing element.

5.4.1.5 Options, Message Editor

Using the message editor options, you can make basic settings for the editor that you use to create messages. The message editor options are divided into the following areas.

- Grid
- Status display
- Representation
- Message number

Grid area:

You can display a grid for character boundaries as an input help. Select the **Display** check box to activate this function. You can also select a **Color** for the grid.

Status Display area:

The status bar at the bottom of the screen in the message editor shows the current cursor position, displayed in either **Dots** (pixel) or **Grid** units.

Representation area:

With the **Output variables** field, you can select the **Color** of output variables in the message editor from a list provided. This makes it easier for you to identify the output variable within a message text.

With the **Zoom** field, select the default zoom factor to be used for displaying a message in the message editor.

You can also choose to display a brief description of the variables (name and address) while the mouse cursor is pointed at the variables area.

In addition, you can display the width of the terminal display using markers in the message editor.

Message Number area:

The area displays the default font to be used to display the message numbers. To change the default font, use the mouse to click the **Select font** button.

To have messages automatically numbered in sequence when created, activate the **Automatic message number**(ing) check box.

When the **Display values hexadecimally** check box is selected, you can also enter and display the message numbers in hexadecimal format. Always precede the hexadecimal value with the letter H (example: H001A).

5.4.1.6 Options, Print Log Editors

The print log editor options provide a number of settings that you can use to define the appearance of the print log in the editor.

You can define settings in the following areas:

- Font
- Grid
- Representation.

Font area:

This area shows the default font used to edit the print logs in the editor. To change the default font, use the mouse to click the **Select font** button. You can only select fonts that are non-proportional (OEM fonts). The size of the characters is also restricted.

By setting up the printer to use the same font, you create a WYSIWYG display.

Provided that your print logs do not contain characters that are reserved in the ASCII table for control characters, you can authorize conversion to ASCII for translation support. If you are using these reserved characters, this information would be lost during conversion.

Grid area:

You can display a grid for character boundaries as an input help. Select the **Display** check box to activate this function. You can also select a **Color** for the grid.

Representation area:

You can use different colors to differentiate between elements in the print log that have been assigned the **Non-printable** and **Output variables** properties.

Select a zoom factor to increase the screen segment in the print log editor.

You can also choose to display a brief description of the variables (name and address) while the mouse cursor is pointed at the variables area.

Select the **Display subprint log** check box if you want to be able to distinguish subprint log elements from elements belonging to the main print log.

To adjust the display of the print log editor when opened to the size of the working field, select the corresponding check box.

5.4.1.7 Options, Translation Support

For the translation support function, you can decide in general whether or not to use an extended functionality. The following areas are provided for these options.

- While Exporting and
- While Importing

Export area:

With While Exporting, you can define whether to use pixels or grids (characters) when specifying the position of an element.

Sorting area:

Possible sorting options are Horizontally-Vertically or Vertically-Horizontally.

Horizontally-Vertically means you want to sort text items using the horizontal position as the main criterion and the vertical position as the sub-criterion.

Vertically-Horizontally means you want to sort text items using the vertical position as the main criterion and the horizontal position as the sub-criterion.

Import area:

Specify whether you want the programming software to notify you if the position of a translated element has changed in relation to the original element position.

You can also determine whether you want a warning to be displayed if the translated text is longer than the original text. If this is the case, you can double-click the corresponding message to open the text editor (screen editor, message editor, text list editor) and reposition or check the text.

However, any changes to position or text length made here will not be reflected in the export file.

5.4.2 Tools Menu, Defining the Interface

Define the interface parameters for connection between the PC and the operating device.

First, select the COM interface whose values you wish to change.

The Interface Parameters area contains default values. To change those values, simply select other values from the appropriate lists.

The default values can be restored at any time. To do so, click the **Default values** button.

To adopt the interface parameters from the S3 file belonging to the translated project, click the **S3 file** button.

5.4.3 Tools Menu, Application ID

The application ID is used to identify a particular application.

The application ID is stored in the S3/SB/CB file of a project and is therefore transferred to the operating device after the download operation.

The same ID is stored in the project management file.

You can compare the ID of a project management file and the ID of an S3/SB/CB file.

Similarly, you can compare the IDs of an S3/SB/CB file and the contents of the operating device. For this, you need to establish a connection between the PC and operating device (by means of a download cable).

The application ID consists of the following elements:

- ID Text
- Version
- Date
- Time of
- Count
- Postfix

ID text:

The maximum length of the ID text is 13 characters. You can specify the file name of the project using the 8.3 format, for example, and edit the ID text as required.

Version:

The version of the programming software is identified by a 5-character string. This text can not be edited.

Date:

The date of creation is represented by a 6-character string. This text can not be edited.

Time of:

A 6-character string indicates the time at which the project was compiled. This text can not be edited.

Count:

The counter is represented by a 4-character string and specifies the number of compilations performed. This text can not be edited.

Postfix:

The Postfix is a random number consisting of 2 characters. This number can not be edited.

Use the **Refresh** button to update the entries in all fields.

The **File name** area allows you to select an S3/SB/CB file created using the programming software and output its application ID.

To do this, select the file and then click the **Refresh** button in the **S3/CB file** area.

In the **Operating device** area, you can read the application ID from the connected operating device. You can either define the interface parameters for the connection separately or use the parameters from the selected S3/SB/CB file. To define the interface parameters, click the **Parameters** button.

5.4.4 Tools Menu, Firmware Update

You can easily replace the firmware for the VCP 01 small operator terminal by using the option that allows you to load the firmware together with the programming software on the small operator terminal. This description does not apply to small operator terminals with a ARM9-CPU, on which the latest firmware is always loaded with the terminal file.

Firmware download process:

1. Select the directory and the file name for the S3 file of the firmware.
2. Select the terminal.
3. Select the firmware version.
4. Create the firmware file (S3 file).
5. Download the firmware to the device

Enter a target directory and a file name for the firmware file. Alternatively, select an existing directory and file.

Select the check box to start a download immediately after the firmware file has been created.

Then click **Next >** to continue.

5.4.4.1 Firmware Update, Terminal Type

Select a terminal type from the left list box.

If the device can be equipped with memories of different sizes, select the appropriate memory size value from the right list box.

Then click **Next >** to continue. Alternatively, click **< Back** to return to the previous window.

5.4.4.2 Firmware Update, Version

Select a firmware version.

Note that the listed firmware versions only apply to the previously selected device.

Then click **Next >** to continue. Alternatively, click **< Back** to return to the previous window.

5.4.4.3 Firmware Update, Generate

The last window of this wizard illustrates the progress of the file creation process.

Once the firmware file has been created successfully, you can either

- Immediately load the file into the device (download cable required) or
- Load the file into the device at a later stage.

Follow the steps below to load the file into the operating device immediately:

1. Connect the device with the PC using the download cable.
2. Set the user mode switch of the operating device to Delete Application Memory.
3. Connect the device with the supply voltage.
4. Once the message Flash is Erased (or a similar message) is displayed, reset the user mode switch to the normal position (leave operating device switched on).
5. When the device displays DOWNLOAD 1, click the Download button.

The download is complete when the device reboots.

5.4.5 Tools Menu, Transmit S3 File, Download

You can load an S3 file (compiled project) into the small operator terminal VCP 01 with serial interface individually. To do so, you directly connect the PC with the small operator terminal using a download cable or a modem. In this case, the operating device must also be connected with a modem.

Enter the S3 file in the **File** area.

In the Download Type area, choose between the following:

- Default, if you want to download directly from a PC to the small operator terminal (baud rates automatically adjusted).
- Modem (fixed parameters), if you are using modems for the connection (fixed baud rate setting).
- Adjustable parameters (configurable parameters) if you want to work with special interface settings.

To change the parameters for the serial interface, click the **Parameters** button.

The parameters for the modem are assigned fixed values:

Parameter	Value
Baud rate	19200 Baud
Parity	Odd
Data bits	7
Stop bits	1
Handshake	Software handshake

Fig. 5-3: Parameters for Modem

In the **Interface** area, select the COM interface that you want to use to connect the small operator terminal.

To check and, if necessary, edit the application ID settings, click the **App-ID** button.

The following modem connection requirements must be fulfilled:

- The system parameters for the SER2 interface must match the above values.
- The **Enable automatic download** or **Enable automatic upload** check box must be activated.
- A modem (transparent) connection must be established before a download/upload is carried out.

5.4.6 Tools Menu, Transmit S3 File, Upload

With this function, you can load the S3 file from the small operator terminal VCP 01 to the PC. To do so, you can directly connect the PC with the small operator terminal using a download cable or establish a connection using a modem. In this case, the small operator terminal must also be connected with a modem.

Before uploading, enter a name for the S3 file so that the file can be stored on the PC.

You can use the interface settings stored in the S3 file for the S3 file transfer or specify your own settings.

Click the **Use modem parameters** check box to establish a connection with a modem.

Click the **Parameters** button to configure the parameters of the serial interface or the connected modem.

The following values must be set as modem parameters:

Parameter	**Value**
Baud rate	19200 Baud
Parity	Odd
Data bits	7
Stop bits	1
Handshake	Software handshake

Fig. 5-4: Parameters for Modem

Click the **Start** button to start the upload operation.

The upload function is primarily used for archiving purposes. The S3 files can not be reloaded into the programming software (to make changes to the project, for example).

The cable used for uploading is the same as that used for downloading.

The following modem connection requirements must be fulfilled:

- The system parameters for the SER2 interface must match the above values.
- The **Enable automatic download** or **Enable automatic upload** check box must be activated.
- A modem (transparent) connection must be established before a download/upload is carried out.

5.4.7 Tools Menu, Transmit Recipe Data Sets

The data set transfer tool (DSTT) is a tool for exchanging recipe data sets.

It uses two tree views to represent the structure of the recipes and data sets.

The structure of the data sets in a file or on an operating device is displayed on the left side. The right side shows the structure of the data sets in a file.

You can exchange the data sets between the files and the operating device.

Highlighted data sets can be copied or deleted. You can change the write-over identifier of data sets in files.

To copy data sets, use the **>>** and **<<** copy buttons.

The following menu items are available in a popup menu or via the keyboard:

- Refresh structure
- Copy
- Paste
- Delete
- Select All
- Overwriteable (Make over-writable)
- Non-overwriteable (Make not overwriteable)

Note:

Data in a recipe data set can only be edited using the programming software.

You can use a popup menu to delete the messages in the status window of the data set transfer tool.

The parameters for the serial interface of the PC are displayed to the bottom left of the dialog.

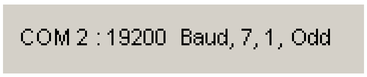

COM 2 : 19200 Baud, 7, 1, Odd

Fig. 5-5: Interface parameters for data set transfer tool

These parameters are used to transfer a data set both to and from the operating device.

Meaning of the parameters:

- COM 2 = Number of the serial interface
- 19200 Baud = Baud rate of transmission
- 7 = Number of data bits
- 1 = Number of stop bits
- Odd = Not even parity

5.4.8 Tools Menu, Documentation

You can document the content of a project in an RTF file. To determine the scope of documentation, select the required project elements arranged in a tree structure.

You can further adjust the documentation layout by selecting and deselecting different documentation parameters.

The following options are available:

- Checking/unchecking a documentation element
- Starting documentation for the selected element
- Document from selected element downward

5.4.8.1 Documentation Parameters, Global Settings

In the **Screen reference lists** area, choose between two options: either display the screen reference lists in the documentation with their name only or specify the current screen for a variable value.

If the screen reference list is specified by name only, the documentation includes a reference to the list. The contents of the list are not output.

You can also specify the current screen to be switched to if a variable assumes the value that you enter in the field next to the appropriate radio button.

5.4.8.2 Documentation Parameters, Projects

If you activate the **Document general project information** check box, the documentation support function takes account of all entries provided under general project information.

5.4.8.3 Documentation Parameters, Screens

In the **Variables** area, choose whether you want to specify the variable positions in pixels or grid units.

Select the **Include numbers in graphic** check box to have the variables of a screen numbered.

The variable description function of a screen is only activated if the **With variable description** check box is activated.

Select the **Empty documentation value for variables** check box if you want only to display a frame for the variable.

Select the **Include text list strings in variable description** check box in the **Text list** area if you want to list the text strings of text lists linked to selection text variables.

For the text strings to be listed, the value entered in the **Max. number of text list strings** field must be greater than the actual number of text list strings. This option allows you to selectively document the text lists that have a limited number of text string entries.

In the **Function keys** area, decide whether you want to document the functions of function keys of a screen.

5.4.8.4 Documentation Parameters, Recipes

In the **Variables** area, choose whether you want to specify the position of variables in recipes in pixels (dots) or grid units.

The variables in a recipe are displayed in a numerical sequence when you activate the corresponding check box.

The description of the variables in a recipe only appears when you select the **With variable description** check box.

Select the **Include text list strings in variable description** check box in the **Text list** area if you want to list the text strings of text lists linked to selection text variables.

For the text strings to be listed, the value entered in the **Max. number of text list strings** field must be greater than the actual number of text list strings. This option allows you to selectively document text lists that have a limited number of text string entries.

In the **Data sets** area, define whether you want to document the data set values for the recipe.

5.4.8.5 Documentation Parameters, Help Screens

In the **Variables** area, choose whether you want to specify the position of variables in help screens in pixels (dots) or grid units.

The variables in a help screen are displayed in a numerical sequence when you activate the corresponding check box.

The description of the variables in a help screen only appears when you select the **With variable description** check box.

Select the **Include text list strings in variable description** check box in the **Text list** area if you want to list the text strings of text lists linked to selection text variables.

For the text strings to be listed, the value entered in the **Max. number of text list strings** field must be greater than the actual number of text list strings. You can thus selectively document all text lists in a help screen that contain fewer than a particular number of entries.

5.4.8.6 Documentation Parameters, Terminal Messages

In the **Variables** area, you can choose to specify the position of variables in system messages using either pixel (dots) or grid units.

The variables in a system message are displayed in a numerical sequence when you activate the corresponding check box.

The description of the variables in a system message only appears when you select the **With variable description** check box.

Select the **Include text list strings in variable description** check box in the **Text list** area if you want to list the text strings of text lists linked to selection text variables.

For the text strings to be listed, the value entered in the **Max. number of text list strings** field must be greater than the actual number of text list strings. You can thus selectively document all text lists in a system message that contain fewer than a particular number of entries.

5.4.8.7 Documentation Parameters, Messages

You can select the following functions from the **Output format** area:

- Activate the tabular display of messages if you want to display the messages in text format, rather than graphically.
- Activate the Include Numbers in Graphic option if you want to display the variables so that they are numbered consecutively for graphical output.
- Activate the With Variable Description option to display a short description below the message for each variable it contains.
- Activate the Empty Documentation Value for Variable option to display a frame for graphical outputs and a line instead of the documentation value for text outputs.

In the **Variable position in** area, you can choose to specify the position of variables in messages using either pixel (dots) or grid units.

The description of the variables in a message only appears when you select the **With variable description** check box.

Select the **Include text list strings in variable description** check box in the **Text list** area if you want to list the text strings of text lists linked to selection text variables.

For the text strings to be listed, the value entered in the **Max. number of text list strings** field must be greater than the actual number of text list strings. You can thus selectively document all text lists in a message that contain fewer than a particular number of entries.

5.4.9 Tools Menu, Translation Support

The translation support function allows the user to export all text elements of a language so that they can be translated externally. The text is exported to a Unicode file. This file can easily be imported and edited in any editor that supports Unicode (such as Editor or Wordpad). When saving the file, make sure that 'text file' has been selected as the file type and that Unicode is specified as the code type. After translation, you can import the text elements again.

Translation support applies to texts in:

- Screens, subscreens, help screens, terminal messages, edit screens
- Messages
- Recipes
- Text lists

Follow the steps below to export the text items of a language:

1. Select the language in the left column (project languages).
2. From the **Edit** menu select the menu item **Export texts**
3. Enter a storage location and a file name in the dialog that next appears and confirm with Save.

An output window displaying the current export status appears.

Once the export is complete, the directory path and file name of the export file are shown in the **Export file path names** column of the **Translation support** window.

The display of the directories and file names indicates whether a language has already been translated.

The export file can be translated into another language using any text editor. Make sure not to edit the first two lines in the file!

5.4.9.1 Translation Support, Export

Use this function to export all of the project text elements and translate them externally.

Follow the steps below to start the translation support tool, export function:

1. Select a language in the project tree.
2. From the **Tools** menu, select **Translation support, Export texts**.

The **Save text elements of the language to a file** dialog appears. You can change the predefined path, which is saved for each specific language to the database. The same path is proposed if you want to import the texts at a later stage.

3. Confirm with **OK**.

The output window appears. Any problems that have occurred during the export are displayed here.

You can edit the ASCII file using any editor.

Make sure that the formatting is not damaged and that the first two lines and object IDs are not changed.

Structure of the ASCII file:

Column 1 = Object ID for the language

Column 2 = Membership

Column 3 = Horizontal position

Column 4 = Vertical position

Column 5 = Length

Column 6 = Font

Column 7 = Text

Each time an export file is created, a new identification code is created and also stored in the database. A message is displayed to the user if the entries do not match when the file is imported; the user can then either terminate or continue the import process.

The following applies to the maximum length of the individual text types:

- Static texts = Unlimited (<4096)
- Message texts = Constant defined for message texts (currently 255)
- Texts in text lists = Unlimited (<4096)

5.4.9.2 Translation Support, Import

Use this function to import texts that you have first exported and then translated.

Follow the steps below to start the translation support tool, import function:

1. Select a language in the project tree.
2. From the **Tools** menu, select **Translation support, Import texts**.

The **Import text elements of the language from a file** dialog appears. The same path and file used for the export are proposed.

3. Confirm with **OK**.

5.4.10 Tools Menu, Import

Use this function to import variables from symbol files that you created while programming the controller. These variables are then transferred to the programming software's variable list and can be used in programming.

You can also import text entries from external files. To do this, first select a text list where you want to enter the texts.

5.4.10.1 Import, Variables from Symbol File

A variable list must be open before variables can be imported from a symbol file.

The programming software provides a function for importing symbol files from other manufacturers. At present, the Bosch SXS format is supported (as of WINSPS 2.11):

To carry out the import, you require the protocol type/controller BUEP19E. Select this as the active controller and open the variable list. Follow the steps below:

1. If necessary, create a new controller (BUEP19E).
2. Open the variable list editor.
3. The Tools/Import/Variables from symbol file menu item is now enabled; activate it.

You can use the import function to perform the following tasks:

- Fill an empty variable list
- Compare existing variable list entries with the import file. In this case, you must select the variable entries in the variable list for comparison. Only selected lines are included in the comparison process. Any additional entries that are recognized at data read-in time as not yet created are then added to the variable list (provided they were selected). This step occurs regardless of whether or not the entries are already contained in the variable list.

5.4.10.2 Import, Variables from Symbol File, Step 1

The Symbols area allows you to import either all symbols (without restrictions) or only symbols that contain the specified string. In this case, the SXS file is searched for symbols with names that contain the specified character string.

The Default Access Mode area allows you to specify Byte or Word as the default access method.

The default access method is applied to all symbols whose access method can not be clearly identified.

The Create Variable Name From area allows you to choose whether to create the variable names on the basis of the symbolic name of the symbol, the symbol's comment or both.

Click Next to continue this procedure.

5.4.10.3 Import, Variables from Symbol File, Step 2

Select a symbol file. Use the **Select** button to search the computer's directory for the symbol file that you want to open and import.

If the symbol file is in the same directory as the programming software, simply enter the name of the symbol file in the input field.

Click **Next** to continue this procedure.

To return to the previous step, click **Back**.

5.4.10.4 Import, Variables from Symbol File, Step 3

Step 3 lists all symbols that can be imported. A symbol is displayed to the left of each entry in the list box. The symbols indicate the status of the entry.

Use the space bar to select or deselect the highlighted entry for the import process. (This is a toggle function).

Complete the procedure with **Finish**.

To return to the previous step, click **Back**.

☞ The data in the list are compared in one direction only. The comparison is always restricted to import data >> variable list data. For this reason, the status always refers to the variable list entry.

5.4.10.5 Import, Text List Items from File

The programming software provides a function for generating text list items from Bosch SXS files (WINSPS 2.11 or higher). This option is available in conjunction with the BUEP19E and PROFIBUS-DP protocols.

To use the import function, you require a controller that supports the formats listed above. First, activate this controller; then open the text list editor containing the text list into which the import will take place. Follow the steps below:

1. If necessary, create a new controller that supports the required format. Now activate this controller.
2. Edit the text list to be imported into or create a new text list.
3. The Tools/Import/Text List Items from File menu item is now enabled; activate it.

You can use the import procedure to

- Fill an empty text list.
- Compare existing text list items with the import file. This requires you to select the text list items (entries) to be compared. Only selected lines are included in the comparison process. Any additional items that are recognized at data read-in time as not yet created are added to the text list (provided they were selected).

The following options are available during an import process:

1. You can determine whether to import sequential function texts or operand texts.
2. You can determine whether to create only one item or whether to scan and generate the data from one specific file.
3. For sequential function texts: the number of items to be generated. There are two fixed values and two automatically generated values for the number of items to be created. The option From Directory Name (Automatically) creates 32 items, provided the selected file is located in a subdirectory that identifies the controller type CL200 in accordance with the WINSPS convention. Otherwise 64 items are generated. With this option and the two fixed values, this number of items is generated even if the scanned file contains fewer items. In this case, a blank is generated as a text. The option From Data (Automatically) only generates the number of items contained in the scanned data.
4. You can select a file or enter an address (using the Bosch syntax) and a description. If you wish to read from a file, you need an SXS file for sequential function texts. The operand texts can only be generated from a cross-reference list file that has been created using the WINSPS software version 2.11.

Once the data are scanned, all items are displayed in a list. A legend below the list indicates whether the items are unchanged, new or modified. An arrow pointing upwards in the icon center indicates the items to be imported. Press the space bar to highlight and select any items you do not want to import. The import arrow will disappear. If the space bar is pressed again, the import arrow will reappear.

Two additional items with the values 0 and 8224 are created when operand texts are generated. In this case, the text consists of a blank.

5.4.10.6 Import, Importing into a Text List, Step 1

During step 1 of the import process, you should specify whether to import

- Sequential function texts for diagnosis purposes in screens showing the sequential function status or
- Operand texts for diagnostic purposes in screens showing sequential function diagnostic information.

The Number of Entries To Create area allows you to choose whether you want to create:

1. 32 entries
2. 64 entries
3. 32 entries automatically from the directory. For directories with fewer than 32 entries, the remaining entries are represented by blanks.
4. as many entries as possible automatically from the data.

In addition, you can decide to import the entire contents of a file or one line only.

Click **Next >** to continue this procedure.

5.4.10.7 Import, Importing into a Text List, Step 2

Importing from one line:

Enter the information using the Bosch syntax into the input fields of the Entry of a Single Address with Comment area.

Importing from a symbol file:

Select a symbol file. Use the **Select** button to search the computer's directory for the symbol file that you want to open and import.

If the symbol file is in the same directory as the programming software, simply enter the name of the symbol file in the input field.

Click **Next >** to continue this procedure.

To return to the previous step, click **< Back**.

5.4.10.8 Import, Importing into a Text List, Step 3

In the **Length of symbolic description** area, you can specifically limit the length of the entry for import to a specific length or decide to ignore the symbolic identifier (description) during import.

By default, no limit applies to the length of the symbolic identifier during import.

In the **Gap** (character between symbol land comment) area, enter a character to be inserted between the symbolic identifier and the comment to separate them visually. If you do not enter a character, the identifier and comment are joined together.

The conditions that apply to the **Length of comment** area are the same as those that apply to the symbolic identifier.

It is not necessary to specify additional settings when importing the contents of only one line.

Click **Next >** to continue this procedure.

To return to the previous step, click **< Back**.

5.4.10.9 Import, Importing into a Text List, Step 4

Step 4 displays all importable symbols in a list. A symbol is displayed to the right of each entry in the list box. The symbols indicate the status of the entry.

Use the space bar to select or deselect (toggle function) the highlighted entry for the import procedure.

Complete the procedure with **Finish**. To return to the previous step, click **< Back**.

The data in the list are compared in one direction only. This comparison direction is always: import data >> text list data. For this reason, the status always refers to the text list entry.

5.4.11 Tools Menu, Export

Copy an image of the current window to the clipboard.

5.4.11.1 Export, Exporting the Window Contents

You can use the **Export window contents** function to copy an image of the current window contents to the clipboard.

The data copied to the clipboard can then be used as required: for example, the obtained images can be inserted into a document.

This function can not be used to create a copy from an image because the data copied to the clipboard are in a bitmap format. For this purpose, use the function Copy/Cut/Paste instead.

5.4.12 Tools Menu, Optimizing a Database

Frequent changes made to a project can result in unnecessary expansion of the project database.

However, it is possible to optimize a database that has been extensively fragmented in this way. Before doing so, ensure no project is open.

Before optimizing a database, you may want to create a backup of the database on a secure data medium.

If a copy has not yet been created, you can terminate the procedure with the **No** button. Otherwise, start the procedure with the **Yes** button.

The process can not be stopped. If the computer crashes or becomes no longer operable for some reason while the optimizing procedure is running, the project database will be damaged.

Enter the name of the project database to be optimized in the dialog displayed and confirm with **OK**.

5.5 Help Menu

The help for the programming software is context-sensitive. This means by pressing the F1 key, a help topic related to the item hat has focus is invoked.

5.5.1 Help Menu, Contents

This menu item allows you to display help for the programming software. The **Contents** tab appears on the left-hand side.

On the **Contents** tab, you can navigate through the help topics just as you would with any file browser (Explorer).

Chapters are displayed as books and single topics are displayed as pages.

To display a topic, simply click the corresponding page.

To print a chapter or a topic, select it and click the **Print** button. In the dialog that appears, specify whether you want to print the selected topic or print the selected topic and all related subchapters.

5.5.2 Help Menu, Index

This menu item allows you to display help for the programming software. The **Index** tab appears on the left-hand side.

Enter a keyword into the **Index** tab. All matching index entries are automatically listed below the input box.

Double-click an index entry in the list to view the associated topic.

5.5.3 Help Menu, Browse

This menu item allows you to display help for the programming software. The **Search** tab appears on the left-hand side.

Enter a keyword into the **Search** tab. If you click the List Topics button, the entire help system is searched and all topics that include the keyword are listed below the input box.

Double-click a topic in the list to view it.

5.5.4 Help Menu, About

Displays the software version level.

Please have this information on hand whenever you contact the hotline.

The **Internet: www.boschrexroth.com** button brings you to the homepage of Bosch Rexroth.

By clicking the **File list** button, you obtain a list of all programming software files installed.

5.5.5 Help Menu, Rexroth in the Internet

This menu item opens your default Internet browser and the homepage of Bosch Rexroth is automatically loaded.

5.5.6 Help Menu, Tip Of The Day

This menu item opens the **Tip of the day** dialog. This dialog presents you helpful tips on the software each time the programming software is started.

5.6 Terminal Type

A project is always created on the basis of a terminal type. This means that parameters such as display size, number of function keys and so on, are predefined in advance.

However, you can also port a project to another terminal type, taking into account any restrictions due to the changed hardware conditions.

Based on the terminal type, you can create a project by creating contents in the project tree branches and defining parameters.

Branch	Contents
Communication	Parameters for the interfaces on the PC and on the operating device. Parameters for the protocols used by the operating device to communicate with controllers. Controller variables
Languages	Screens, function keys, messages, recipes, print logs, text lists
User management	Passwords
Script management	Scripts and script variables
Supplementary functions	Parameters for the polling area, date & time, running time meters, Unicode, status information, date input options, screen saver
Resources	Symbols, images, image lists
Project management	Projects with startup language, project languages, project controller

Fig. 5-6: Project tree contents

5.6.1 Terminal Type, Change

If you want to port a project from one terminal type to another, select the required terminal from the **Type** field.

Depending on the model, you can select one of the procedures below for grayscale display.

- 16
- 8
- 4
- 4 + Dithering

5.6.2 Terminal Type, Memory Size

Select the appropriate **memory size** for operating devices with different memory capabilities.

5.6.3 Terminal Type, Touch Parameters

You can define the following default settings for touch-sensitive terminals:

Default sound:

This is the default setting for the signal tone that sounds when the button is pressed.

The following options are available:

- No beep (no sound)
- Beep when pressed (sound for press touch)
- Beep when released (sound for release touch)
- Beep when pressed and released (sound for press and release)
- Continuous tone while pressed (sound during press touch)

Enter the duration of the beep in seconds under **Time**.

Default Image for System Icon:

Enter the image to be displayed if a system icon is not available.

See chapter “System Icon“ on page 6-127.

5.6.4 Terminal Type, Color Palette

You can change the color combination used in operating devices that can display grayscale or color. The changed color combinations are then available throughout the project.

The list has two columns: color 1 and color 2.

Since an element is usually displayed with one color, color 1 and color 2 are identical in this case. If you have a flashing element, you must use a color combination where color 1 is not the same as color 2. The color of the element then changes from one color to the other.

Use the numerical values for Red, Green and Blue to combine new colors or **edit** existing colors.

The color combinations with the number 16 to 63 are supplied for flashing elements. The default flashing frequency is 1 Hz.

5.6.5 Terminal Type, Fonts

Any WINDOWS fonts can be used for programming purposes. You must however specify the fonts in advance.

Each font specified is listed with its font size (in points) and font style (bold or italic).

5.7 Comment

You can enter comments of any length about the database created.

Example of some useful details you might include:

- Database version
- Last changed on (date)
- Database created by (name)
- Machines and machine variants for which the project has been created.

5.8 Communication

The operating device can communicate with the PC, a controller, a logging printer or a scanner.

To add a controller as a communication partner to the project, follow the steps below:

1. Right-click with the mouse on the **Communication** branch.
2. Select **New controller** from the menu.

The **Protocol selection** dialog opens.

3. Select the name of a controller.
4. Confirm with **OK**.

The **Create controllers** dialog appears.

5. Enter a new name, if necessary.
6. Confirm with **OK**.

The name of the controller is added as an element in the **Communication** branch.

5.8.1 Protocol Selection

Choose the protocol that you want to use for your project to exchange data between the operating device and PLC.

For this purpose, select the name of the protocol to be used and confirm with **OK**.

The next dialog allows you to assign another name to the selected protocol; that is, the name to be displayed for the protocol in the list box of the Controllers tab.

To ensure smooth exchange of data between the PLC and the operating device, you must define the protocol parameters.

In addition, you need to create a system of allocation (in the variable list) between the variables, their addresses and the type of data.

The type of addressing used depends on the PLC.

See chapter “3S serial“ on page 7-2.
See chapter “Bosch BUEP19E“ on page 7-15.
See chapter “BRC-Symbolic“ on page 7-25.
See chapter “DeviceNet“ on page 7-35.
See chapter “IndraLogic“ on page 7-61.
See chapter “PROFIBUS-DP raw“ on page 7-74.

5.8.1.1 Communication Parameters

Press the **Edit** button to open a dialog where you can change the communication parameters for the selected communication protocol.

See chapter “3S serial“ on page 7-2.
See chapter “Bosch BUEP19E“ on page 7-15.
See chapter “BRC-Symbolic“ on page 7-25.
See chapter “DeviceNet“ on page 7-35.
See chapter “IndraLogic“ on page 7-61.
See chapter “PROFIBUS-DP raw“ on page 7-74.

5.8.2 PC >> Terminal

This function is available for all terminals with an SER2 serial interface.

Define identical communication parameters on both ends.

You can exchange the following information by means of this connection:

- Recipes
- Data sets
- Terminal files

5.8.2.1 PC >> Terminal, Enable Transfer

By selecting the **Enable automatic download** check box, you instruct the operating device to recognize and start a download automatically. This also applies to the **Enable automatic upload** check box.

5.8.2.2 PC >> Terminal, Interface Parameters

The interface parameters defined for the connection between the PC and terminal influence the download process.

First, select the COM interface whose values you wish to change.

The Interface Parameters area contains default values. To change those values, simply select other values from the appropriate lists.

The default values can be restored at any time. To restore the values, click the Default Values button.

To adopt the interface parameters from the S3 file belonging to the compiled project, click the S3 File button.

The PC interface is set up with these default values.

Parameter	Value
Baud rate	19200 Baud
Parity	Odd
Data bits	7
Stop bits	1
Handshake	Software handshake
Port (interface)	COM2

Fig. 5-7: Default settings for the RS232 interface

5.8.3 Terminal >> Printer & Scanner

This function is available for all operating devices with an SER2 serial interface. Operating devices with Windows CE operating system only use the USB interface for printing.

Define the communication parameters required to connect the operating device with a logging printer or a scanner.

5.8.3.1 Terminal >> Printer& Scanner, Interface Parameters

The operating device is equipped with an RS232 interface. This interface is used to connect the device with a logging printer or a scanner.

The interface uses default values when the programming system is started for the first time.

Parameter	Value
Baud rate	19200 Baud
Parity	Odd
Data bits	7
Stop bits	1
Handshake	Software handshake

Fig. 5-8: Default settings for the RS232 interface

5.8.3.2 Terminal >> Printer & Scanner, Scanner Parameters

Scanner extension:

To run a scanner on the SER2 interface of the operating device, you must first activate the **use** option for scanner parameters.

Scanner parameters:

Enter the default and initialization parameters. These parameters initialize the scanner and define the specific data transfer information.

In the **Default** field, enter an ASCII string that activates the scanner.

In the **Initialization** field, enter an ASCII string that configures the scanner.

The parameters entered can not be validated since you can not select a specific scanner type.

You can connect any scanner with an RS232 interface that transmits ASCII characters.

Scanner <--> Terminal protocol:

Press **Transfer data with ENTER key** (Data take-over with ENTER key) if you want to press the Enter key every time a bar code is scanned, in order to enter the data.

If you are using a scanner that requires confirmation of receipt once the scanned value has been transferred, activate the **Terminal sends ACK signal following receipt** (Terminal sends ACK after receive data) function.

To allow the terminal to recognize when a scanned value begins and ends, enter the characters sent by the scanner beforehand and afterward as the **data prefix** and **data postfix**.

5.8.3.3 Terminal >> Printer & Scanner, Unicode

For printer control, the characters intended for output must be converted according to a conversion table. Characters that are not contained in the conversion table are not printed.

Three recognized character tables are used worldwide.

- European standard
- GB2312-80
- GB18030

Choose the character table for your requirements by selecting the relevant radio button from the **Printer standard** area.

In the **Language switching** area, you can enter a string in the **Activation code** area. This string is sent to the printer in order to activate the correct character set before every print operation.

Refer to the printer's user guide for information on entering the correct activation code.

Example:

- Switch to GB2312-80 character set = FS & (Hex: 1C 26)
- Switch to European standard = FS. (Hex: 1C 2E)

In China, the **built-in standard** code type is used. The table below shows the difference between this character set and GB2312-80:

Unicode	GB2312-80	Built-in standard
0x3000	0x2121	0xA1A1
0x5509	0x3026	0xB0A6
0x54C0	0x3027	0xB0A7
0x7691	0x3028	0xB0A8
0x9F44	0x777E	0xF7FE
... 0xH	0xL 0x(H+0x80)	0x(L+0x80)

Fig. 5-9: Conversion table for the built-in standard character set

5.8.4 Protocol, Loop-through Operation

Using loop-through operation, you can link the programming unit to the operating device that connects through the signals to the controller.

If you are using the BRC-Symbolic protocol, loop-through operation allows automatic recognition of the programming unit. With point-to-point connections, you must activate/deactivate and monitor loop-through operation using system variables.

☞ While loop-through operation is activated, the terminal's COM1 interface can not be used for other applications (such as a scanner, printer, transfer of recipe data sets).

☞ See chapter “Loop-through Operation“ on page 6-208.

5.8.5 Encoding of Alphanumerical Strings

Text character identification:

For alphanumeric strings read by the operating device from the controller, you must specify the standard code to be used to interpret the characters.

For 8-bit characters:

- ISO 8859-1 (Latin-1)
- IBM OEM CP437
- ANSI (MS Windows)

For 16-bit characters:

- 16-bit Unicode

For characters of variable bit length:

- Unicode UTF-8

5.9 Languages

The Languages category combines all of the elements that belong to a national language.

The elements are as follows:

- Global function keys
- Message system
- Recipes
- Print log
- Subprint logs
- Language resources

5.9.1 System Defaults

In system defaults, you define parameters that are valid for the current language. Starting with the number for the current language, you can set specific fonts to be used for texts, the screen displayed at the start of the user interface and much more.

5.9.1.1 System Defaults, Identifier

In the **Language number** area, you can specify a number for the current language in the range of 1 to 16. You will reuse this language number if you create a language selection using a text list.

These settings you can assign a number for the **customer version** to serve as a reference.

The valid range of values is 0 to 255.

The value is saved in the **UserVersion** system variable.

You can output this system variable in any input/output screen.

There is no other functional link in the operating device. You can not change the value in the operating device.

See chapter “UserVersion“ on page 6-157.

5.9.1.2 System Defaults, Base Screens

Within a project, you have five screens whose functions differ from a standard screen for input and output.

Startup:

The startup screen is the first one displayed after the electrical supply to the operating device is switched on. It is only displayed for 5 seconds. During this time, all LEDs on the operating device are turned on so that they can be checked for proper functioning. The operating device then switches to the main screen. If you press the Enter key or Setup button while the startup screen is displayed, the operating device switches to the setup screen.

Setup:

You can only access the setup screen if you press the Enter key or the Setup button during the initialization phase (the startup screen is displayed for five seconds).

On the setup screen, it may be useful to include the following functions:

- Activate/deactivate download function
- Set protocol parameters
- Select PLC protocol

- Set date and time
- Display firmware level of the operating device

If you program these functions using system variables, the operator can select the corresponding parameter from text lists.

Password:

If you wish to use the same screen for each password query, create a password screen and specify its name here.

It may be useful to allow all function keys go to the main screen so that the operator does not get stuck on the password screen.

Main:

The main screen is always displayed after the initialization phase has been completed.

The user interface must start from here. Starting from this screen, branches can be created to all other screens. The main screen can contain all of the elements of a standard input/output screen.

Default help:

The default help screen is always displayed if no specific help screen has been set up for the current screen. It is also displayed if no help screen is available for the variable and current screen.

5.9.1.3 System Defaults, Standard Format

You have the option to select a font for the current language so that language-specific characters can be displayed. This font is used as the standard font for all texts you input during programming.

Fonts selected in Options can disable this setting for representation in various editors.

5.9.1.4 System Defaults, Reference Screen

In the **Time-controlled target** area, you can specify a base screen that is displayed if the operating device is not used for a set period of time.

You can choose all of the user interface screens as the base screen with the exception of the following:

- Startup screen
- Setup screen
- Password screen

In the base screen, note that you may need to reactivate password protection. Enter a value between 10 sec. and 9999 sec. for the **time**.

5.9.2 Screens

Screens refer to the contents of a page on the operating device display. This means that screens vary in size, depending on the operating device being used.

You can display and enter texts and variables (250) in screens.

You can define the layout of these screens and use images to customize them to your own specific needs. To display basic elements that are identical in every screen, you can create these elements in subscreens and link the subscreens with screens.

For each screen, you can program specific softkeys or function keys to allow you to jump to other screens. You can also program cursor keys to open other screens.

Furthermore, you can use the function keys to change variable values.

If you use the same screen names in all languages, you can use global screen elements to design similar screens. Global screen elements (for example, global variables) appear in each screen with the same name.

5.9.2.1 Screen, General

The general parameters for a screen are divided into the following areas:

- Number
- Value of the **ScreenOffset** system variable
- Options

Number:

Each screen of a project has a unique number. If you do not enter a number, the next consecutive number is automatically assigned.

You can use the screen number to carry out the following for screens:

- Enter screens in a text list and display as a screen menu
- Switch screens over from the controller
- Write documentation for screens

Value of the ScreenOffset variable:

Enter a value to be used to multiplex controller variables.

If a screen is called up with screen offset, the offset value is entered in the **ScreenOffset** system variable. At run time, the offset value is added to the address of a controller variable in this screen that is multiplexed with the **ScreenOffset** system variable.

Example:

If the screen offset is 20 and the basic address of the controller variable to be multiplexed is MW100, the result is a controller address of MW120 when the screen is called up.

Options area:

Choose the direction in which the next variable is accessed when you press the Enter key after editing a variable value.

Activate automatic data release if you wish to have the option to edit variable values immediately after changing to the current screen. Otherwise, you will need to press the Data Release key.

Automatic data release can not be used with operating devices that use a touch screen.

See chapter “ScreenOffset“ on page 6-223.

5.9.2.2 Screen, Password-Protection

The parameters to password-protect a screen are divided into the following areas:

- Access level
- Reset password

Access Level area:

The access level corresponds to a threshold value for password management. You use it to determine if and when operators must enter a password.

Configurable Values	Default Value
0 to 255	0

Fig. 5-10: Access level

The initial access level default value for each screen is 0. This corresponds to the status general release.

If the threshold value is above the view level, the operator can only display this screen by entering the correct password.

If the threshold value is above the edit level, the operator can only change the variables contained in the screen by entering the correct password.

Reset Password area:

If you activate the Reset Password parameter, password-protection is reactivated after you exit this screen. If you return to this screen, you must enter your password again.

5.9.2.3 Screen, Help Screen

You can create a specific help screen for each screen and display it by selecting the Help key, a corresponding button or clicking a free area of the display background.

If you wish to provide a help screen for the current screen, select an existing help screen.

You can not use the **Activate help screen** parameter for operating devices equipped with a keyboard. You activate a free area of the display background to call up the help screen.

5.9.2.4 Screen, Background Color

For each screen, you can select a separate color for the entire screen area. All elements that may be contained in a screen cover over the background color!

You can only use this parameter for operating devices that use gray scales or color display.

5.9.2.5 Screen, Script Parameters

At the same time a screen is displayed, you can execute a script. Enter the name of the script in the **Display script** field.

5.9.2.6 Screen, Function Keys

You can redefine the functions of the keys (function keys and special function keys) for each screen. Keys defined for one screen specifically are also referred to as softkeys.

The specific function of a key within a screen takes precedence over the global function assigned to the same key.

Possible definitions are as listed below:

- Direct jump to screen
- Indirect jump to screen (Screen reference list required)
- Write a value to a variable (0 to 255) when a key is pressed
- Write a value to a variable (0 to 255) when a key is released

Direct jump to screen:

Setting up a function key to call up a screen directly allows the operator to access a specified screen quickly by simply pressing this function key. The authorization to access the screen is checked by the password management function; if necessary, a password screen is displayed first. In this case, a valid password must be entered before the screen is called up.

Indirect jump to screen:

The controller can effect an indirect jump to a screen. The operator always presses the same function key. At run time, the controller writes a value to the variable for screen reference control. The operating device reads this value and compares it to the values in the screen reference list. The screen that is assigned the corresponding value in the screen reference list is then accessed.

Follow the steps below to jump to a screen indirectly:

- Create a screen reference list
- Define a transfer variable
- Specify a screen reference list in the function key table.

5.9.2.7 Linking with Screen

First decide whether the operator should be able to press a key and jump to a screen directly or whether he/she should switch to a screen whose number can be controlled by the PLC.

All of the screens already created are available in a list for direct selection.

Follow the steps below to select a screen for direct screen change:

1. Select a field from the **Screen** column.
2. Select **Edit** from the context menu.
3. Select the **Reference to screen (direct)** selection button.
4. Select the required screen from the list.
5. Confirm with **OK**.

The name is then entered in the table field.

To select screens indirectly, you must first create screen reference lists. Screen reference lists already created are provided in a list.

Follow the steps below to select a screen reference list for an indirect screen change:

1. Select a field from the **Screen** column.
2. Select **Edit** from the context menu.
3. Select the **Reference to screen list (indirect)** selection button.
4. Select the name of the required screen reference list in the list.
5. Confirm with **OK**.

The name is then entered in the table field.

5.9.2.8 Linking with Variable

Follow the steps below to select a variable name:

1. In the **Press variable** or **Release variable** column, select a field.
2. Select **Edit** from the context menu.
3. In the dialog box that opens, select the name of the required variable.
4. Confirm with **OK**.

The name is then entered in the table field. To transfer a variable value to the controller only for the time the key remains pressed, select the same variable name for the release variable and enter a value of 0 (zero) for this action.

5.9.2.9 Linked Subscreens

Subscreens are linked to screens to ensure screen elements are identical and always appear in the same positions. This allows you to standardize the design of the project.

The left list box displays all of the existing subscreens.

The right list box shows all of the subscreens linked to the current screen.

You can use the buttons in-between to perform the following actions:

> Adds the selected subscreen to the linked subscreens. A subscreen can be added only once.

>> Adds all of the listed subscreens to the linked subscreens.

< Removes the selected subscreen from the list of linked subscreens.

<< Removes all of the subscreens from the list of linked subscreens.

Complete your entry with **OK** to return to the previously used function.

5.9.2.10 Screen Background Color

Static text and variable screen elements can be displayed in different color combinations. The selected screen element is assigned a background and foreground color from the list for this purpose.

Select the screen element and choose **Parameters screen element** from the context menu.

The list has two columns: color 1 and color 2.

The background for the screen element is generally displayed in color 1. For the background to flash, a different color must be selected for color 2. The background color then switches to and from one color to the other. If color 2 is now the same as the foreground color, it creates the impression that the element is being switched on and off.

5.9.3 Subscreens

Subscreens are used to ensure a consistent design across all screens containing the same elements. To achieve this, you define a subscreen which contains the required elements, and then link it to each of the screens that are supposed to have the same design.

Subscreens can contain all the elements of a standard screen with the exception of tables, recipe and message fields.

When compiled, the screen and linked subscreens are joined to become one resulting screen.

5.9.4 Help Screens

To optimize usability, you can create a help screen for each screen and each input variable. To call up this help on the operating device, press the Help key, a button that has been programmed for this purpose or click a free area of the display background.

If data release is not requested, a help text appears for the screen. The help text for the variable that is currently selected appears, provided the editor for entering a variable value has been activated.

The default help screen appears if no specific help screen has been linked to a screen or a variable.

A help screen is the same size as a standard screen. You can enter static texts, background images, output variables and tables in a help screen. To ensure consistent design, help screens can also be linked to subscreens.

5.9.4.1 Default Help Screen

The default help screen is always displayed if you have not linked any help screen to the screen or input variable. The default help screen is always available and only displays a blank page if it is not programmed.

5.9.4.2 Help Screen, Background Color

You can select a background color or color combination for the background of a screen.

A color combination comprises color 1 and color 2.

Color 1 means that the background usually has this color.

Color 2 means that the background alternates between color 1 and 2 in flash mode.

Selecting contrasting colors for color 1 and color 2, e.g. no. 17 for the background and no. 16 for the foreground, will ensure that the element remains legible when flashing.

However, if you select color no. 1 for the background and color no. 16 for the foreground, the element will 'disappear' when it is flashing.

5.9.5 Global Function Keys

Global function keys can perform a function from any part of the project. Any settings defined for global keys apply to screens only if no other functions have been assigned to the keys within that screen.

Possible functions are listed below:

1. Jump to a screen
2. Transfer variable value when a key is pressed
3. Transfer variable value when a key is released
4. Activate the LED in a function key when a message has been received from the serial or parallel message system

5.9.6 Message System

The message system is an integral part of the user interface. Messages are reactions to events that are communicated to the operator in an intelligible form. A distinction is made between internally and externally generated messages, depending on where the event occurred. The diagram below shows the structure of the message system.

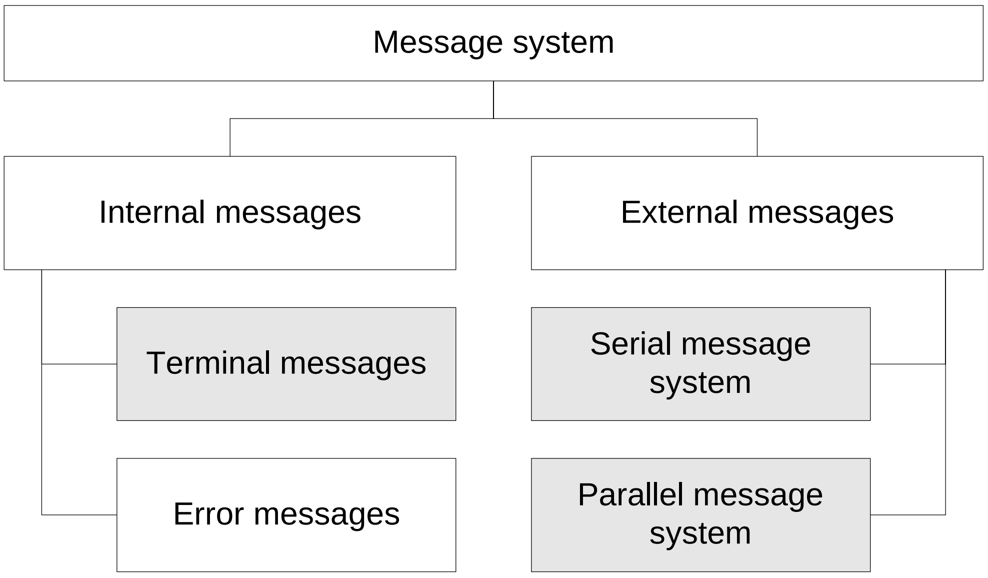


Fig. 5-11: Structure of the messages system

The areas shown in gray color can be freely designed during programming.

5.9.6.1 Messages

The message editor assigns texts to message numbers that may be transmitted from the PLC to the operating device.

Messages are distinguished by their priority. The priority is determined by a combination of message number and message group where the following applies:

The lower the message number and group number, the higher the message priority.

Messages can be created on the basis of the following specifications:

- A maximum of 3000 messages
- A maximum of 8 message groups
- A maximum of 255 characters per message
- A maximum of 2 variables per message
- A maximum of 255 line breaks per message

The message editor is in tabular form.

The table consists of two columns allowing you to enter the message numbers and message contents.

For automatic assignment of numbers to messages, the appropriate check box must be selected, in the Message Editor options dialog. Existing messages can automatically be numbered consecutively later.

A scale is displayed above the column for the message contents. The scale is displayed with **Pixel** or **Grid** as the unit depending on the setting in the message editor options dialog.

Any text can be entered in the column for the message contents. Up to two output variables and up to 255 line breaks can also be included in the message. Line breaks are displayed within the row as a gray rectangular. Every line break counts as a message character, thereby reducing the total number of displayable characters for the message.

Within the input field of the message editor, marks indicate the width of the display of the selected operating device. The marks can be activated or deactivated in the message editor options dialog.

The messages can be displayed on the operating device with the font defined for the message field. This font can be selected freely depending on the operating device type.

The font used to display the message numbers in the message editor can be changed individually in the **Tools** menu, **Options** menu item, **Message editor** tab. Any of the Windows fonts can be selected.

You can either display all of the messages in the message field of a screen or display them individually using the system variables.

Any messages received are acknowledged with either the **Acknowledge** special key or a function key. The function key must control the value of the **RepmanQuitKey** system variable.

Message Parameters

Colors:

Displays the background color and foreground color assigned to the message. These colors are used to display the message on the operating device.

To replace with other colors, click **Assign**.

The setting can be viewed immediately after you return to the message.

Screen change:

If a message is called up, a screen of the operator's choice can be called up at the same time.

The screen does not have to be called up immediately when the message is received, but can be called up once a message is selected within the message field.

- **Screen change on arrival of message**:
  If the controller transmits a message, the operating device automatically changes to the specified screen. You can select any screen here. The parameter when a message arrives does not apply to screen changes that have been enforced by the controller. Furthermore, the screen accessed does not necessarily have the same number as the message.
- **Screen change within the message screen**:
  Once messages arrived have been displayed in a screen's message field, the top message is indicated by the > cursor arrow. You can select other messages using the Cursor Up and Cursor Down keys. You can activate the **RepmanChgMask** system variable to effect a screen change to the screen of your choice according to the settings for the message. This parameter can therefore be used to design a message-based user interface. It allows you to access screens faster, which contain more detailed information on the message in question, to allow particular settings etc.

Help screen:

The help screen the operator can call up for message management can be specified in the Help Screen area.

Group:

The message entered previously can be assigned to a group (1 to 8).

The messages can then be displayed in different message screens in accordance with the established groups.

Advantages:

- Further classification of messages (e.g. according to the area of the plant)
- Further classification of messages according to priority
- New assignment option for parallel messages

For serial message channel:

- **Simultaneous screen change and message transfer:** (Simultaneous screen change and message call)
  If this check box is selected, a screen change takes place at the same time as the message transfer and the message is stored in the serial message memory.
  This only applies to messages transmitted by the serial message channel and messages to which 8000H was previously added.
  Example: The value 800AH is written to the serial message channel. In this case, screen number 10 is called up and the message with the number 10 assigned is stored in the serial message memory.

Transfer from the parallel to serial message memory: (Put message from parallel to serial message memory)

- **Store in ser. message memory:**
  Messages of the parallel message system can be stored in the memory for serial messages. This allows parallel messages to be captured that appear for a short time only. You can also capture the moment messages appear and disappear using the arrived/departed status identifiers.
- **With arrived/departed status:** (With appear/disappear status)
  Messages with this parameter are identified by a status attribute at the start of the entire message text.
  The following letters are provided to identify the status. You can, however, replace them with other letters. The letters are assigned to the relevant language so that you can choose different letters for each language.
  K = Arrived (Appears)
  G = Departed (Disappears)
  Q = To be acknowledged
  q = Acknowledged.
- **Automatically delete "arrived/departed":** (Automatic clear appear/ disappear)
  A message of the parallel message system with this parameter is stored in the memory for serial messages when it appears. The message is first labeled with the Arrived (appear) status here.
  Once the parallel message is no longer active, it is removed from the serial message memory once again.
- **Acknowledgment of message:**
  You can acknowledge messages with this parameter that have been stored in the memory for serial messages.
  Acknowledged messages are then labeled with the Acknowledged identifier.
  To acknowledge messages, you require the Acknowledge key, a function key which controls the **RepmanQuitKey** system variable, or a relevant button.

Printing the message in the serial message memory:

- **Automatic printing when the message arrives:** (automatic print of message)
  With this parameter, a message which reaches the operating device is not only displayed in the message field but also printed.
  You must select this parameter separately for each individual message which should be printed directly.
  A further requirement for automatic printing is that the **RepmanRepPrint** system variable be set to 1.
- **Automatic deletion of the message after printing:** (Automatic clear of message after printing)
  Messages with this parameter are deleted from the serial message memory immediately after they are printed on the logging printer.
  A further requirement for automatic printing is that the **RepmanRepPrint** system variable be set to 1.

- **Automatic printing after the message has been acknowledged:**
  (Automatic print after acknowledgment)
  Messages with this parameter are output on the printer immediately after they have been acknowledged.
  A further requirement for automatic printing is that the **RepmanRep-Print** system variable be set to 1.

Foreground and Background Color of the Message

Allows you to assign the background and foreground color combination to a message.

The setting can be viewed immediately after you return to the message.

Font for Editing Messages

Select a font to be used when you create or edit messages in the Messages window.

To save a new font in the list of fonts available for selection, click the **New font** button.

The selected font only applies when messages are displayed within the programming software, not on the operating device.

5.9.6.2 Controller Area, Activate Messages

Representation (option):

Select either reduced or extended representation of the parallel message system.

With the reduced representation you define the parallel message system with start address, size and polling time.

With the extended representation you define up to 8 sections of the parallel message system with start address, acknowledge areas, sizes, offsets, starts of message areas and ends of message areas.

Parallel message system

For the parallel message system, you must enter a starting address of the data area where the messages are stored in the controller in bit-coded form.

You specify the number of bytes to define the **size** of the area for the status messages in the controller.

You can specify a maximum of 256 bytes for this area.

By entering the **polling time**, you specify the interval at which the operating device reads the status messages' data area from the controller.

You can enter values from 0 to 25.5 seconds for the polling time.

The active messages are displayed in a screen with a message field for parallel messages. The status messages can be sorted according to various criteria.

5.9.6.3 Controller Area, Options

Parallel message system:

You specify a name for a variable with the same size as the variable for status messages in the field **start address acknowledgement area**.

Serial message system:

From the controller, you can **delete all acknowledged messages** if you write the bit pattern **E216h** to the controller address **Delete variable to messages** and write the control code **7FF5h** to the serial message channel.

If you want to **delete all messages**, you need to write the control code **7FFEh** to the serial message channel. If you want to use this function of the operating device, you must assign a 'variable for deleting messages'.

5.9.6.4 Terminal Area, Memory & Message

General parameters:

You can enter a message number **to display a message directly** (direct representation). The message number also indicates its **priority**. The lowest message number has the highest priority and the highest message number has the lowest priority.

All messages that have a higher priority than the message number specified here are handled using a special procedure in the operating device when they appear. These messages are indicated by the status LED flashing in the direct selection key of the message screen or are signaled by a terminal message.

Using the **Size of message buffer**, you define how many messages can be stored in the operating device. Specify the maximum number of messages to be managed by the operating device.

5.9.6.5 Terminal Area, Function Keys

Parallel message system

Select a function key with which you wish to have the current message of the parallel message system displayed directly.

Serial message system:

Select a function key with which you wish to have the current message of the serial message system displayed directly.

5.9.6.6 Terminal Area, Identifier

Status identifiers

Status identifiers can be used to reflect the status of a message in the message system.

Any character of the standard font can be used for status identification.

Status identifiers are only appended to messages stored in the serial message system.

The history starts at the time the message arrives. The **Arrived** (appears) status identifier is appended to the most recently arrived message. This status identifier is appended to both serial and parallel messages. In the case of parallel messages that have to be acknowledged, the Arrived (appears) status identifier is not appended until the messages have been acknowledged.

The **Departed** (disappears) status identifier marks the point of time when a message is cleared. This status identifier is only appended to parallel messages that were stored in the serial message memory.

The **To be acknowledged** status identifier (rather than Arrived) is appended to messages with the Acknowledgment of Message parameter enabled.

Once the message has been acknowledged, the **Acknowledged** status identifier is appended.

Group identifiers:

Group identifiers classify the messages in 8 groups And thereby assign a priority to the messages, at the same time.

The following applies: the higher the group number, the lower the priority of the message.

5.9.6.7 Terminal Area, Terminal Messages

Messages for terminal messages:

Specify the message to be output when the terminal indicates a **fatal error**.

Also specify a message to be output when a **communication error** with the connected controller occurs.

These error messages are stored in the serial message system.

5.9.6.8 Output Format, Status

General parameters:

You can have the messages displayed in indented format. To do so, specify the number of characters by which the lines are to be indented after the first line. The value you enter here refers only to the display of messages in the operating device's message field.

Message sorting:

Messages can be sorted for output. You can use the corresponding radio button in the Message Sorting area to define the sort criterion.

Using the corresponding check boxes in the representation of message area, you can mark which parameters of a message are also output. The parameters can be combined in any way.

Sort **by time, newest first**:

With this sorting order, the message that arrived last in the serial message memory appears at the top of the message field.

Sort **by time, oldest first**:

With this sorting order, the message that arrived first in the serial message memory appears at the top of the message field.

Sort **by priority**: If messages are sorted by priority, they are sorted in the message field in such a way that the message with the highest priority is at the top and the messages below are listed according to descending priority.

The lower the message number, the higher the message priority.

Sort **by group**:

With this sorting order, the message that has the lowest group number appears at the top of the message field. If several messages belonging to the same group are listed, they are sorted according to time, with the newest ones first.

Message representation:

Representation with **message group**:

The system outputs the group identifier before the message text.

Representation with **message number**:

The system outputs the message number before the message text.

Representation with **message date**:

The system outputs today’s date before the message text. In the date, the year can either be output as two digits or four digits. The value of the date is frozen with the message.

Representation with **message time**:

The system outputs the time before the message text. The value of the time is frozen with the message.

5.9.6.9 Output Format, Recording

General parameters:

You can have the messages displayed in indented format. To do so, specify the number of characters by which the lines are to be indented after the first line. The value you enter here refers only to the display of messages in the operating device's message field.

Message sorting:

Messages can be sorted for output. You can use the corresponding radio button in the Message Sorting area to define the sort criterion.

Using the corresponding check boxes in the representation of message area, you can mark which parameters of a message are also output. The parameters can be combined in any way.

Sort **by time, newest first**:

With this sorting order, the message that arrived last in the serial message memory appears at the top of the message field.

Activate this sorting if you like to print out the messages online.

Sort **by time, oldest first**:

With this sorting order, the message that arrived first in the serial message memory appears at the top of the message field.

Activate this sorting if you like to print out the messages online.

Sort **by priority**: If messages are sorted by priority, they are sorted in the message field in such a way that the message with the highest priority is at the top and the messages below are listed according to descending priority.

The lower the message number, the higher the message priority.

Sort **by group**:

With this sorting order, the message that has the lowest group number appears at the top of the message field. If several messages belonging to the same group are listed, they are sorted according to time, with the newest ones first.

Message representation:

Representation with **message group**:

The system outputs the group identifier before the message text.

Representation with **message number**:

The system outputs the message number before the message text.

Representation with **message date**:

The system outputs today’s date before the message text. In the date, the year can either be output as two digits or four digits. The value of the date is frozen with the message.

Representation with **message time**:

The system outputs the time before the message text. The value of the time is frozen with the message.

5.9.6.10 Output Format, Printer

Printer:

You can have the messages printed in indented format. To do so, specify the number of characters by which the lines are to be indented after the first line. The value you enter here refers only to printing messages on a logging printer.

You can choose from the following variants for outputting to the printer.

1. No output.
2. Print the entire message (lock), SER2 reserved exclusively for message output
3. Print formatted message (lock), SER2 reserved exclusively for message output
4. Print the entire message (temp), SER2 only reserved temporarily for message output
5. Print formatted message (temp), SER2 only temporarily reserved for message output

5.9.7 Recipes

You can set parameters in the following categories for every recipe:

- Transfer active
- Data set transfer from the terminal
- Data set transfer to the terminal
- Data set sorting

The recipes editor is divided into two halves. You have a view on the left similar to a screen. This section allows you to enter the static texts (e.g. item names and units) and variables specific to a particular recipe. You enter the variable values stored in a data set in the right-hand section of the editor.

Controller:

The data set is controller-specific. You must therefore select the name of the controller from a list.

Variable offset:

You must assign an offset to every variable in a data set. This offset is used to store the data in the operating device memory.

Example:

You should specify offsets for 4 variables in a data set.
Variable 1 = Byte variable
Variable 2 = Word variable
Variable 3 = Double-word variable
Variable 4 = Byte variable

The offsets are as follows:

Variable 1 = Offset 0
Variable 2 = Offset 1
Variable 3 = Offset 3
Variable 4 = Offset 7

Data set name:

In the Data Set Name column, enter the names of all data sets that should be used for the current recipe. Transfer to and from the controller is coordinated using this data set name.

Data set number:

In the Data Set Number column, enter a unique number to be used to identify the data set. Transfer to and from the controller is coordinated using this data set number.

Variable value:

In the left view, first select the variable and then enter a value that this variable should have in the relevant data set.

Storage location:

For each data set, you can specify whether it is stored in the operating device's Flash or RAM. Flash data sets are protected against direct access while you can change data sets in RAM directly.

See chapter “Working with Recipes“ on page 6-112.

5.9.7.1 Recipes, Transfer Active

Variable:

Specify the **starting address** of controller variable for the Write Coordination byte. It contains the Data Set Download Release bit. The controller releases data set transfer with this bit.

Enter the **size** of the controller variable. The size can vary depending on the operating device as this controller variable is often identical to that for the entire polling area.

Polling time:

Enter the value for the polling time in seconds.

The polling time is based on the total system load. Note the cycle times for other variables!

See chapter “Working with Recipes“ on page 6-112.
See chapter “Write Coordination Byte“ on page 6-247.
See chapter “Data Set Download Release“ on page 6-249.
See chapter “Size of the Polling Area“ on page 6-244.
See chapter “Polling Time“ on page 6-243.

5.9.7.2 Recipes, Data Set Transfer from the Terminal

You can load the data sets in the operating device to the controller. You can also load (any changed) data sets from the controller to the operating device. In this context, the data set transfer is always initiated by the operating device, but only when the controller has activated the corresponding release (Data Set Download Release bit in the Write Coordination byte).

Before a transfer is performed, the communication partner must be advised of the recipe number and data set number.

In the **Transfer from Terminal** area, enter the name of the variable for the recipe and data set number. To do this, click the button beside the input box. The operating device automatically enters the recipe number and data set number into this variable before data are transferred to the controller.

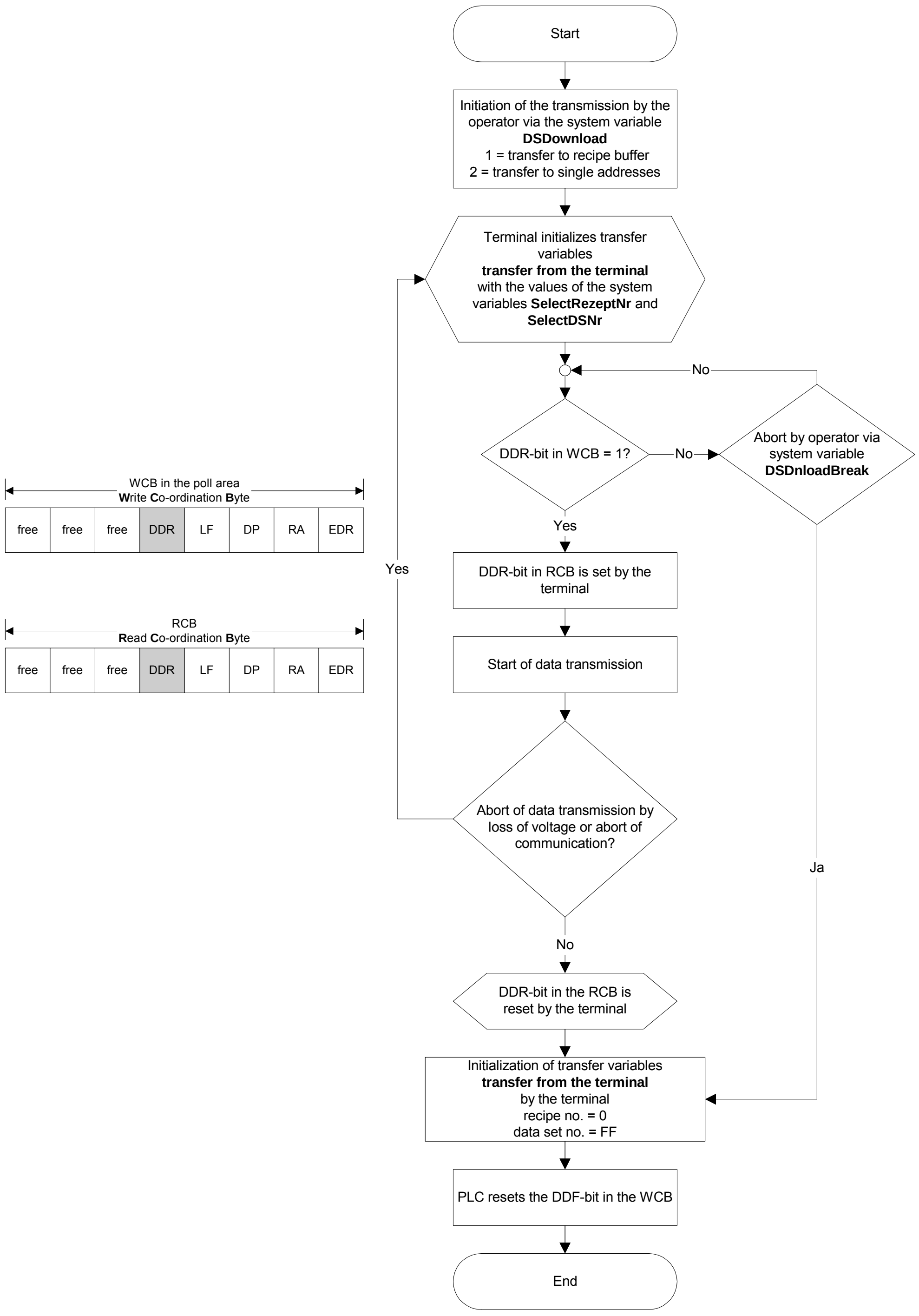

Start
Initiation of the transmission by the operator via the system variable **DSDownload** 1 = transfer to recipe buffer 2 = transfer to single addresses
Terminal initializes transfer variables **transfer from the terminal** with the values of the system variables **SelectRezeptNr** and **SelectDSNr**
No
DDR-bit in WCB = 1?
No
Abort by operator via system variable **DSDnloadBreak**
Yes
WCB in the poll area
Write **C**o-ordination **B**yte
free
free
free
DDR
LF
DP
RA
EDR
RCB
Read **C**o-ordination **B**yte
free
free
free
DDR
LF
DP
RA
EDR
Yes
DDR-bit in RCB is set by the terminal
Start of data transmission
Abort of data transmission by loss of voltage or abort of communication?
Ja
No
DDR-bit in the RCB is reset by the terminal
Initialization of transfer variables **transfer from the terminal** by the terminal recipe no. = 0 data set no. = FF
PLC resets the DDF-bit in the WCB
End

Fig. 5-12: Data transfer to the controller (operator-controlled)

5.9.7.3 Recipes, Data Set Transfer to the Terminal

Before a transfer is performed, the communication partner must be advised of the recipe number and data set number.

In the Transfer from Controller area, enter the name of the variable for the recipe and data set number. To do this, click the button. The controller must write the recipe number and data set number to this variable before data is transferred to the operating device.

Specify whether the data sets should be requested by the **number** or **name**. If they are to be requested by the name, enter the maximum **length of the name**.

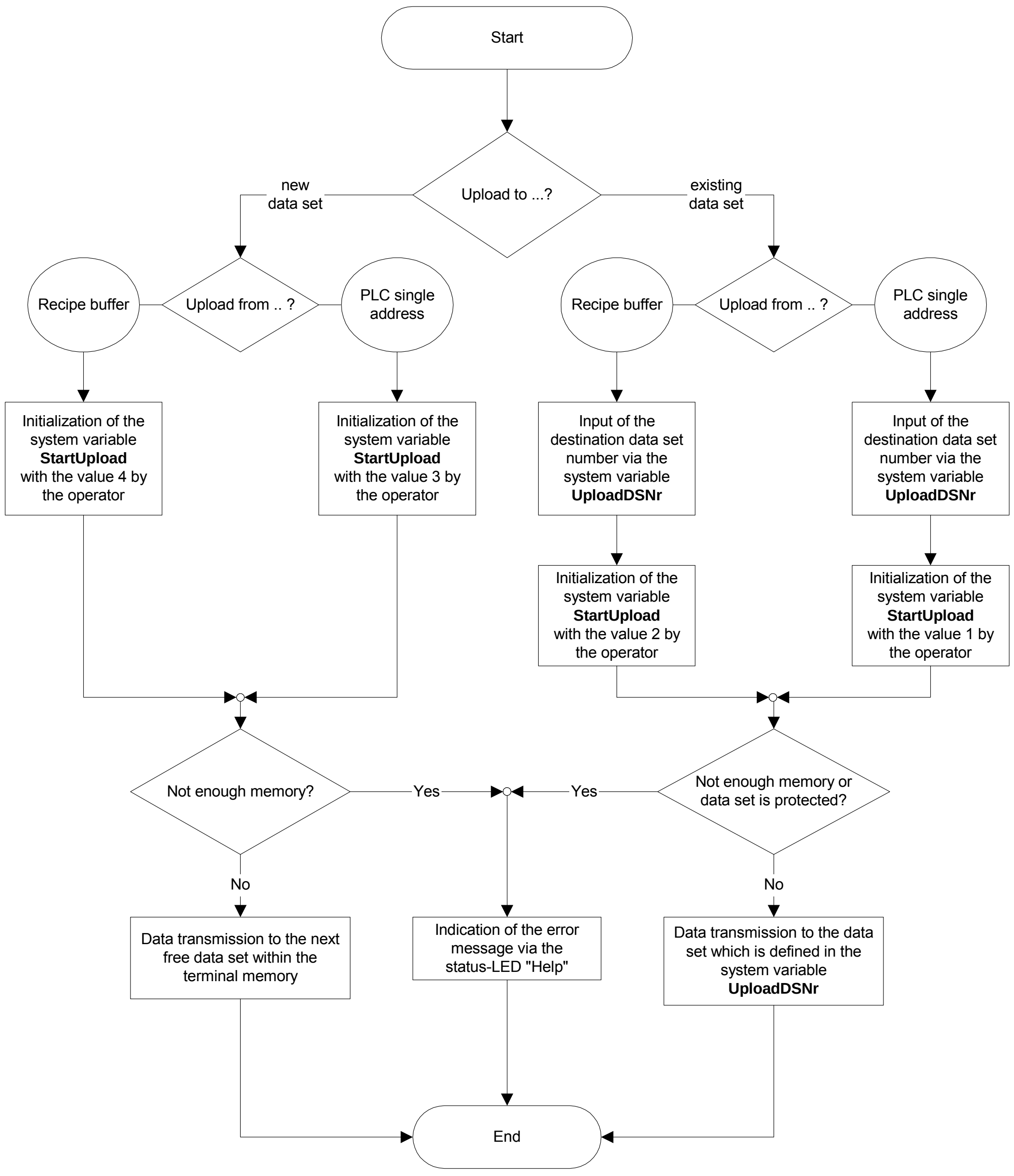

Start
Upload to ...?
new data set
existing data set
Recipe buffer
Upload from .. ?
PLC single address
Recipe buffer
Upload from .. ?
PLC single address
Initialization of the system variable **StartUpload** with the value 4 by the operator
Initialization of the system variable **StartUpload** with the value 3 by the operator
Input of the destination data set number via the system variable **UploadDSNr**
Input of the destination data set number via the system variable **UploadDSNr**
Initialization of the system variable **StartUpload** with the value 2 by the operator
Initialization of the system variable **StartUpload** with the value 1 by the operator
Not enough memory?
Yes
Yes
Not enough memory or data set is protected?
No
No
Data transmission to the next free data set within the terminal memory
Indication of the error message via the status-LED "Help"
Data transmission to the data set which is defined in the system variable **UploadDSNr**
End

Fig. 5-13: Data transfer to the operating device (operator-controlled)

5.9.7.4 Recipes, Data Set Sorting

Specify how the data sets should be sorted. The following are the sorting options:

- By data set number
- By data set name

5.9.7.5 Recipes, Number

Specify any **recipe number**, provided it is unique, to use as an identifier for the recipe.

5.9.7.6 Recipes, Controller Target Address

Specify the controller address for the **recipe buffer** which can be used to replace the recipe data sets.

You do not need to create a recipe buffer if you can write the data set values to individual controller variables!

5.9.7.7 Recipes, Font

The **Font** area shows the font used to display recipe elements in the programming software. To display the recipe in a different font, click the **New font** button.

5.9.8 Print Logs

A print log comprises the printer page layout, static texts, and output variables. Print logs can not contain graphical elements.

The print logs can not be displayed on the operating device. They are merely managed by the operating device and output using the SER2 or USB (only for Windows CE) interface.

The logs do not necessarily have to be output to a printer. The data sent can also be read in by a higher-level system (host computer) and processed further in any way.

A prefix and a postfix can be selected for each individual print log.

Prefixes and postfixes are escape sequences that are transferred before and after the actual print log. The sequences are stored in a list.

To ensure that print logs have a uniform layout, and to simplify the input of elements that are always the same, you can create subprint logs, and link them with the print log.

You can use the following options to select default settings in the programming software for the print log editor.

- Font
- Grid/grid color
- Color of non-printable elements
- Color of output variables
- Zoom factor when opening the log editor

To customize the print log to your own requirements, adjust settings in the following categories for each parameter:

- Adjust printer
- Format
- Controller

5.9.8.1 Print Logs, Adjust Printer

You can have one or several escape sequences sent to the printer before (prefix) and/or after (postfix) each print log. You can use these to:

- Generate a line feed
- Generate a page feed
- Change font
- Change font size
- Change font style

See the printer documentation for more information on the escape sequences you can define for your logging printer.

Enter the escape sequences with a unique name in the list.

5.9.8.2 Print Logs, Format

For the printer Page Settings, you can specify the Lines in Page and Characters in Line.

For detailed information on defining the page size, please refer to your printer manual.

5.9.8.3 Print Logs, Controller

A variable must be defined to be able to transfer the print log number from the controller to the operating device. If the controller transmits a command to print a print log, the operating device uses the log number currently stored in this variable.

To initialize the print log from the controller, the control code **7FF7h** must be written to the polling area.

The operating device writes the status of the print process to the same controller address. This status is output as a signed decimal number.

Value	Status
0	Printing complete
-1	Printer in use
-2	Print log not found
-3	Print log stopped

Fig. 5-14: Parameters for print logs

5.9.8.4 Print Logs, Number

Specify any **number**, provided it is unique, as an identifier for the current print log.

5.9.8.5 Print Logs, Escape Sequences

Prefix escape sequence:

Select an escape sequence to be sent to the printer before the current print log is output.

Postfix escape sequence:

Select an escape sequence to be sent to the printer after the current print log is output.

Page feed:

Select this parameter if a page feed should be executed before the current print log is printed.

5.9.8.6 Print Logs, Subprint Logs

To link a print log with one or more subprint logs, select the desired subprint logs from the left list box (Available Subprint Logs).

You use the **>** button to transfer the selected entries to the right list box (Linked Subprint Logs).

Using the **>>** button will transfer all of the entries to the right list box.

To remove entries from the right list box, mark them and use the **<** button.

You use the **<<** button to remove all entries from the right list box.

5.9.8.7 Print Log, Text

Text:

You can use static text both for printable and non-printable comments in a print log .

Select the **Non-printable element** check box if you only wish to display and not print the static text.

5.9.8.8 Print Log, Representation

Representation:

You can use attributes to influence how print log elements are displayed.

From the **Prefix** and **Postfix** fields, select escape sequences if you do not wish to have an element output in the standard way.

You can only select escape sequences already defined.

You can select/deselect standard attributes using the check box in the **Output attributes** field.

5.9.8.9 Print Log, Variable Type

Select a representation type for the variable.

Click the **Edit type** button if you wish to define its representation more specifically.

5.9.8.10 Print Log, Variable Reference

Double-click the name of a variable in the tree structure to select it.

A red flag beside the name indicates that it is correctly selected!

5.9.9 Subprint Logs

Subprint logs contain static texts and/or variables that are to be used repeatedly in several print logs - always in the same position and in the same form.

Print logs can be linked with any number of subprint logs.

There must be no overlap between elements of different subprint logs. If overlapping occurs, an error message is generated when the project is compiled.

5.9.10 Language Resources

All elements of a national language are considered language resources that you can use in various places within the current project.

5.9.10.1 Text Lists

A text list is an assignment of texts to numerical values. It is used wherever a text is to be displayed in screens instead of a numerical value from the controller.

In the left column of this table, enter the numerical value to which the text is assigned that will be entered into the right column.

To enable entry of hexadecimal numerical values, select Options from the Tools menu and select the Display Values Hexadecimally check box in the List Editors tab. The letter H must always precede hexadecimal numerical values!

The elements in a row are not entered into the system until you press the Enter key or use the Tab key to move to the next (blank) line.

The length of texts in the right column can be limited. This allows you to ensure that texts can also be displayed on operating devices equipped with smaller-sized displays.

Text Lists, Parameters

As a parameter for a text list, you can set the length of an individual text to be unlimited or limited to a specific value.

You can only enter a length if you first select the **Limited to** option.

If you wish to limit the length of a text subsequently, while existing texts will retain their full length, new ones added will need to be of the specified length.

The Font area shows the font used to display the texts of the text list in the programming software. Click the button if the texts are to be displayed in a different font.

When using proportional fonts, it is possible that more characters can be displayed in the selection text variable than the limit specified for the text length.

The value specified in the Options menu item of the Tools menu appears as the default value.

☞ See chapter “Options, List Editors“ on page 5-10.

Text Lists, Selection

You can control the selection of texts from a text list with the contents of a variant buffer. Each variant buffer bit represents the entry in a text list. If the bit is set, the text within the text list is enabled for selection (in a selection text variable).

Enter the name of the variable for the variant buffer in the **Variant buffer** area. In addition, you need to specify the size of the buffer in bytes.

The maximum buffer size is 32 bytes.

Example:

The size of the variant buffer is 1 byte. The bit pattern is as follows (LSB to MSB): 11110011.

The text list is as follows:

Number	Value	Text
0	0	Blue
1	10	Green
2	20	Red
3	30	White
4	40	Black
5	50	Orange
6	60	Yellow
7	70	Brown

Fig. 5-15: Sample text list

The selection text variable in a screen can thus display the following texts:

- Blue
- Green
- Red
- White
- Yellow
- Brown

Text Lists, Editing Font

Select the font you wish to use to edit the text lists in the programming software.

Text Attributes of the Text List

You can select the foreground and background colors as text attributes of the text list.

You can also select the attributes Underline, Flashing and Inverse attributes using the check boxes.

The available text attributes depend on the selected operating device type.

The **Attributes are valid** check box is used to activate the selected attributes.

5.9.10.2 Terminal Messages

Terminal messages are generated by the operating system as a result of internal plausibility checks. It is activated immediately after the corresponding event has occurred.

Pending terminal messages are signaled to the operator

- by a flashing Help key status LED and
- setting the **StateHelp** system variable to the logical value "1".

The text of the terminal message is displayed if you

- press the Help key or
- press a corresponding button or
- press on a free area of the display background.

For key-operated devices, the terminal message is displayed for the length of time the key is pressed. For operating devices with a touch screen, you can program an input/output screen specifically for displaying terminal messages. The button in which the **StateHelp** system variable is configured can be used to change to this screen at the same time.

If several terminal messages are pending at the same time, they will be displayed in the order of their numbers. The terminal message number "1" represents the highest priority.

You can change the text of the terminal messages to suit your needs. The size of one screen is available for each terminal message text. The terminal message text can be freely designed using the terminal-specific fonts. Additional character attributes or graphics are not possible.

Icons are available for terminal message display on operating devices equipped with a touch screen. This allows terminal messages to be displayed graphically.

Output of the texts is language-specific, i.e. if the user interface is multilingual, the terminal messages are displayed in accordance with the selected language. The terminal messages are assigned by means of terminal message numbers. The terminal message number stands for a predefined event.

A brief description consisting of 20 characters is used to provide an explanation of the system number. The length of the texts is designed to allow them to be displayed on one line, even on the smallest operating device display.

A newly created system contains the following terminal messages with brief descriptions:

Number	**Brief Descriptions**	**System icon**
1	Wrong format	
2	Value too large	
3	Value too small	
4	Replace battery	
5	Message overflow	
6	New message	
7	Message buffer full	
8	Invalid screen no	
9	Invalid message no.	
10	Print log invalid	
11	Interface in use	
12	Invalid password	

Fig. 5-16: Terminal messages and icons

Number	Brief Descriptions	System icon
13	Password unchanged	🔒
14	Overvoltage	🔋
15	Data set protected	✗
16	Illegal data set	✗
17	Data set unknown	12 ✗
18	Data set memory full	🟥
19	Data set active	✗ activ
20	Data set transfer	✗
21	Password missing	🔒 ✗
22	Editing mode active	activ
23	Data set file error	✗
24	Data set format	⊘
25	Number invalid	123 ✗
26	Loop-through active	▣
27	No data set address	✗ abc
28	Recipe unknown	✗
29	Data set download	✗
30	Scanner error	✗
31	Print log unknown	✗

Fig. 5-16: Terminal messages and icons

Number	Brief Descriptions	System icon
32	Language switching error	
33	Flashcard information	
34	New application	NEW S3
35	Format error	
36	Script error	

Fig. 5-16: Terminal messages and icons

Terminal Messages, Parameters

Background color:

Select a color for the background on which the terminal message should be displayed.

Status line:

Specify a text as status line. This text will be displayed in any screen by means of the **Status text** system variable as long as the terminal message is displayed.

See chapter “StatusText“ on page 6-216.

Terminal Messages, Memory

You can store important terminal messages in the serial message memory. To do this, select the number to be used to enter the terminal message in the serial message memory.

Terminal Messages, Touch Screen Terminals

System icon:

Select the system icon to be displayed instead of the terminal message.

Symbol colors:

Select a color combination for both the system icon foreground and background color.

See chapter “Terminal Type, Color Palette“ on page 5-34.

Terminal Message 1 - Wrong Format

You are attempting to enter an invalid data format into a variable field of the numerical editor. For example, the number of places entered before the decimal point exceeds the setting specified in the user interface.

Terminal Message 2 - Value Too Large

You are attempting to enter a value into a variable field of the editor that exceeds the variable's upper limit. The upper limit is defined in the user interface.

If you delete the terminal message text from the user interface, no terminal message will be issued, but the maximum permitted value will be entered instead.

Terminal Message 3 - Value Too Small

You are attempting to enter a value into a variable field of the editor that is below the variable's lower limit. The lower limit is defined in the user interface.

If you delete the terminal message text from the user interface, no terminal message will be issued, but the minimum permitted value will be entered instead.

Terminal Message 4 - Replace Battery

A test performed on the battery indicated that its capacity has fallen below the limit value. This test is repeated every 60 minutes. To avoid loss of data when replacing the battery, the information in the respective operating device's user manual must be complied with.

The same message appears when you remove the battery; switching the device off at this point will, however, result in the battery-backed data being lost!

Terminal Message 5 - Message Overflow

Indicates that the system is unable to process the external messages quickly enough. On display of this message, one message has already been lost.

Terminal Message 6 - New Message

This text is displayed when the operating device has received a new external message whose priority exceeds the programmed threshold value and no direct selector key has been assigned to the message screen.

Terminal Message 7 - Message Buffer Full

This text is displayed as a warning that the next external messages may overwrite the oldest or lowest-priority messages (depending on the configuration).

Terminal Message 8 - Invalid Screen Number

This text is displayed to indicate that a non-existent screen number has been transmitted by the controller via the serial message channel.

Terminal Message 9 - Invalid Message No.

This text is displayed to indicate that the controller has transmitted a message number that does not exist in the user interface.

Terminal Message 10 - Print Log Invalid

The operator or the controller attempted to start a print log that does not exist in the user interface.

Terminal Message 11 - Interface in Use

Interface X3 is already being used by another print job. You are attempting to transmit different types of data to the printer at the same time (e.g. to print recipes and messages).

Terminal Message 12 - Invalid Password

You entered a password which does not exist in the password management function. With this message, the previous access authorizations (view and edit level) are reset.

Terminal Message 13 - Password Unchanged

The operator did not enter the same new password the first and second time.

Terminal Message 14 - Overvoltage

The operating device has detected that the supply voltage is too high. Switch the device off immediately to avoid damage. Check the supply voltage.

Terminal Message 15 - Data Set Protected

You attempted to modify individual values of a data set stored in the Flash or to delete the entire data set.

Terminal Message 16 - Illegal Data Set

The data set number you specified as the destination for the data set copy process exists already or is outside the valid range (for example, Flash). The upload destination for a data set transfer is invalid (e.g. zero).

Terminal Message 17 - Data Set Unknown

The data set with the number you selected does not exist in the data set list.

Terminal Message 18 - Data Set Memory Full

You attempted to create a new data set but the data set memory is full.

Terminal Message 19 - Data Set Active

You attempted to erase or to copy to the active data set, or to select a data set even though the active data set is currently being edited.

Terminal Message 20 - Data Set Transfer

You attempted to initiate a data set transfer to the controller even though the previously initiated transfer has not yet been completed.

Terminal Message 21 - Password Missing

You attempted to switch to a password-protected screen or to edit a password-protected screen without having entered a password with sufficient authorization.

Terminal Message 22 - Editing Mode Active

You attempted to change to another screen while the operating device was in editing mode.

Terminal Message 23 - Data Set File Error

The data set file loaded from the PC to the operating device contains a syntax error. The error can be located by means of the line number system variable.

Terminal Message 24 - Data Set Format

The size or internal version identifier of a data set loaded from the PC to the operating device and the corresponding values in the programming software do not match.

Terminal Message 25 - Number Invalid

The bit pattern read from the controller is not valid for a floating point number. The number is output as 0.0.

Terminal Message 26 - Loop-through Active

The selected action was not performed because the loop-through operation is active.

Terminal Message 27 - No Data Set Address

The addresses for the data set transfer did not exist at the time of the controller's request.

Terminal Message 28 - Recipe Unknown

You attempted to select a recipe that does not exist in the operating device.

Terminal Message 29 - Data Set Download

You initiated a data set transfer to the controller (download), but the Data Set Download Release bit in the Write coordination byte (bit 4) has not yet been set by the controller.

Terminal Message 30 - Scanner Error

Three different types of errors may have occurred:

1. A value was scanned, but the editor required was not open yet.
2. The scanner does not support this variable type.
3. The parameter settings for the scanner (prefix and postfix) are not correct.

Terminal Message 31 - Print Protocol Unknown

You selected a print log that does not exist.

Terminal Message 32 - Language Change Error

The language number you wish to switch to does not exist.

Terminal Message 33 - Flash Card Information

The following errors may have occurred:

- Data error while downloading a project from the Compact Flash card.
- You inserted the Compact Flash card into the operating device.
- You removed the Compact Flash card from the operating device.

You can use the **CFCardError** system variable to display the type of error.

See chapter “CFCardError“ on page 6-222.

Terminal Message 34 - New Application Necessary

The project in the operating device or the project in the controller has been modified and the operating device is trying to access variables that meanwhile have been modified.

Terminal Message 35 - Format Error Time/PLC

The controller transmits an incorrect format for time and date.

☞ See chapter “Date and Time Image“ on page 6-233.

Terminal Message 36 - Script Error

An error occurred when a script was being edited.

5.9.10.3 Screen Reference Lists

Screen reference lists allow you to assign values to the screen names of your choice.

This allows you to design a user interface that can be controlled by the controller. The controller can do this by transferring a value to the operating device. You must specify a transfer variable for **Global control** for this purpose.

If the operator presses a key that is linked with a screen reference list, the value is read from the transfer variable and compared with the screen reference list. Then, the screen corresponding to the specified value is called up.

Screen reference lists enable different user interfaces in the case of machines with different versions for example.

Screen Reference Lists, Global Control

In the **Screen Reference List Control** area, enter the name of the transfer variable for global control of all screen reference lists. The controller writes the screen number that should be displayed on the operating device to this variable.

If a screen reference list is linked with its own control variable, global screen reference list control is not valid for this purpose.

Screen Reference Lists, Local Control

In the **Screen Reference List Control** area, enter the name of the transfer variable for control of the current screen reference list. The controller writes the screen number that should be displayed on the operating device to this variable.

Global control is thus disabled for the current screen reference list.

5.9.10.4 Edit Screens

If input variables are configured for an operating device with a touch screen, these variables are edited in the relevant edit screen. The editor is activated by clicking the input variable.

Five different types of edit screens are available:

- Decimal
- Hexadecimal
- Binary
- Increment
- Alphanumeric

See chapter “Working with Edit Screens“ on page 6-69.

Edit Screens, Background Color

From the list, select a color combination for the edit screen background.

See chapter “Terminal Type, Color Palette“ on page 5-34.

Edit Screens, Cancelling

If you enable the **Close if free area of edit screen is pressed** (Close at print in free edit screen area) parameter, you can close the Editor by pressing a free area of the edit screen display and the current screen reappears; the data is not saved in this case.

5.10 User Management

In the user management function, create an access profile for every operating device user. This profile consists of passwords and the related authorization levels.

See chapter “Working with Password Protection“ on page 6-153.

5.10.1 User Management, Passwords

In the password management function of the programming software, you can define any number of passwords, each with a length of 11 characters. When you are allocating the different passwords, think of how you want to structure access authorizations.

Example:

- Password for the manufacturer of the plant, machine, and so on
- Password for on-site service
- Password for the person setting up the machine, foreman, overseer
- Password for the operator of the system

The passwords are stored in the operating device’s Flash memory. These passwords are the basic setting that is active when you first start up the system after each download. The passwords are also stored in the operating device’s RAM.

You can reactivate the passwords stored in the Flash memory by writing to the system variable **FlashPasswd**.

You can change all passwords from the operating device, except for the master password (first password in the list). To do this, write the password to be changed to the system variable **MskchgPasswd**.

You must then write the new password twice to the system variable **ChangePasswd**. The new password is valid immediately, provided you enter the same new password twice.

If this is not the case, a corresponding terminal message is issued and the password is reset.

Passwords are stored and compared as 11-character strings. Use the alphanumeric editor to enter the passwords in the operating device.

Program passwords globally, and not on a language-specific basis.

See chapter “FlashPasswd“ on page 6-197.
See chapter “ScrchgPasswd“ on page 6-196.
See chapter “ChangePasswd“ on page 6-197.
See chapter “Working with Password Protection“ on page 6-153.

5.10.2 User Management, Parameters

Activate the **Deactivate password functionality in case of control screen change** parameter if you want the controller to switch to password-protected screens.

A change to a password-protected screen would be preceded by a password query without the operator knowing what to expect next.

5.11 Scripts

Organize the script variables in a variable list and all created scripts.

Plan tasks which start the scripts.

See chapter “Working with Scripts“ on page 6-71.

5.11.1 Scripts, General

Newly created scripts are automatically numbered sequentially. The **Number** field shows you the number of the current script.

5.11.2 Variable List

The script variable list is in tabular form. In each line, you can enter the name of a script variable, its data type and also the length, in the case of string variables. All script variables that you enter into this table exist globally in all scripts and are stored retentively.

Permitted types are:

- bool
- int
- uint
- double and
- string.

	Name	**Type**	**Length**
0	SkriptVar1	bool	
1	SkriptVar2	int	
2	SkriptVar3	uint	
3	SkriptVar3	double	
4	SkriptVar4	string	12
New>>			

Fig. 5-17: Example of a script variable list

5.11.3 Planned Tasks

You can assign at least four scripts that should start after boot from the operating device or start cyclically to a starting behavior and a time span in a table.

The table for planned tasks consists of the columns

- Script,
- Behavior,
- Time span and

- Time unit.

Column Script:

Select an already existing script.

Column Behavior:

Select **cyclically** to apply a continually returning task.

Select **after booting** to start a task after booting from the operating device.

Column Time span:

Enter a time that is applied as delay between cyclically returning tasks or as delay after booting of the operating device.

Column Time unit:

Select a mulitplier for the entered time span. You can select:

- 100 msec.
- 1 sec.
- 10 sec.
- 1 min.
- 1 hour

5.12 Supplementary Functions

The supplementary parameters which you can specify in the programming software are saved in the operating device.

5.12.1 Supplementary Functions, Polling Times

Set the polling times to the intervals at which the respective elements are to be polled. When entering the polling times, pay attention to the total load being placed on the communication path.

Choose times as long as possible, to minimize the communication load, and as short as possible, to avoid missing any events!

See chapter “Communication With a Controller“ on page 6-267.
See chapter “Polling Time“ on page 6-243.
See chapter “Working with Running Time Meters“ on page 6-225.

5.12.2 Supplementary Functions, Polling Area

The polling area consists of three segments:

1. Write coordination byte (1 byte or 1 word)
2. Serial message channel (2 bytes or 1 word)
3. Segment for controlling the status LEDs in the function keys (up to 12 bytes or 6 words)

You can

- operate the entire polling area with a single field variable OR
- operate each segment of the polling area with separate variables.

If you operate the polling area with 1 variable:

1. Specify the name of the variable for the polling area.
2. Specify the polling time.
3. Specify the size of the polling area.

If you operate the polling area with 3 variables:

1. Specify the name of the variable for the write coordination byte.
2. Specify the name of the variable for the serial message channel.
3. Specify the name and the size of the variable for controlling the status LED in the function keys.

See chapter “Working with the Cyclic Poll Area“ on page 6-240.
See chapter “Write Coordination Byte“ on page 6-247.
See chapter “Serial Message Channel“ on page 6-242.
See chapter “Status LEDs of Function Keys“ on page 6-239.

5.12.3 Supplementary Functions, Transfer Date & Time

Send to control:

Here you can specify the name of the variable used for the data associated with the date, time and day of the week in the controller as well as the polling time for data exchange.

Transfer of:

Select the elements to be transferred. If no element is selected, no data transfer will take place. For the date, choose whether you want the year to be transmitted as a 2 or 4-digit value.

See chapter “Working with a Real-Time Clock in the Operating Device“ on page 6-233.
See chapter “Set Clock in Operating Device“ on page 6-231.
See chapter “Date and Time Image“ on page 6-233.
See chapter “Setting the Real Time Clock from the Controller“ on page 6-234.
See chapter “Transferring the Real-Time to the Controller“ on page 6-235.

5.12.4 Supplementary Functions, Receive Date & Time

Receive from control:

Here you can specify the name of the variable used for the data associated with the date, time and day of the week in the controller.

See chapter “Working with a Real-Time Clock in the Operating Device“ on page 6-233.
See chapter “Set Clock in Operating Device“ on page 6-231.
See chapter “Date and Time Image“ on page 6-233.
See chapter “Setting the Real Time Clock from the Controller“ on page 6-234.
See chapter “Transferring the Real-Time to the Controller“ on page 6-235.

5.12.5 Supplementary Functions, Reset Running Time Meters

Eight running time meters are available in the operating device.

For the **Start/stop variable** (control byte), enter the variable name which constitutes the address where the controller can influence the running time meters in the operating device.

If bit X is set in the control byte when polling is carried out, the running time meter X is incremented.

Using the address of the **Reset variable** (Reset Byte), you can reset the running time meters in the operating device.

The **polling time** specifies the time intervals at which the operating device is to increment the running time meters.

The running time meters are activated in the operating device as soon as you have entered a variable name for the control byte and specified a value for the polling time. If the polling time is 0 or if there is no address for the control byte, the Running Time Meter function in the operating device is off.

You can specify an address in the controller for each running time meter. The operating device stores the value of the corresponding running time meter to this address when requested by the controller to do so. For this purpose, the controller needs to write the hexadecimal code **7FCFh** into the serial message channel of the polling area.

For each variable, you must provide a 32-bit memory area in the controller!

See chapter “Working with Running Time Meters“ on page 6-225.
See chapter “Write Values of Running Time Meters to Controller“ on page 6-228.

5.12.6 Supplementary Functions, Set Running Time Meters

Specify a variable name for each running time meter. The value of the running time meter is placed into these variables.

See chapter “Working with Running Time Meters“ on page 6-225.
See chapter “Write Values of Running Time Meters to Controller“ on page 6-228.

5.12.7 Supplementary Functions, Unicode

Subareas (subrange) list:

Unicode fonts are split up into code ranges that are assigned to specific character types or languages. In order to save memory in the operating device, not all Unicode subranges are used. Instead, you can choose the subranges that you need to create text elements.

The "Memory needs for each font" field indicates how much memory the operating device needs for the selected subranges.

The subranges available in the programming software are not the same as those of www.unicode.org.

In the following table you find an assignment of the software program's subranges to the original Unicode subranges.

Name for Unicode subrange in software program	Unicode chart name	Unicode character code range taken into account by the programming software
--		
(Latin-1; always available)	C0 Controls and Basic Latin; C1 Controls and Latin-1 Supplement	U+0000 to U+00FF (0 to 255)
Latin erweitert /		
Latin Extended	Latin Extended-A; Latin Extended-B; IPA Extensions; Spacing Modifier Letters; Combining Diacritical Marks	U+0100 to U+036F (256 to 879)
Griechisch / Greek	Greek	U+0370 to U+03FF (880 to 1023)
Kyrillisch / Cyrillic	Cyrillic; Cyrillic Supplementary	U+0400 to U+052F (1024 to 1327)
Armenisch / Armenian	Armenian	U+0530 to U+058F (1328 to 1423)
Hebräisch / Hebrew	Hebrew	U+0590 to U+05FF (1424 to 1535)
Arabisch / Arabic	Arabic; Syriac; Thaana	U+0600 to U+08FF (1536 to 2303)
Devanagari / Devanagari	Devanagari	U+0900 to U+097F (2304 to 2431)
Bengali / Bengali	Bengali	U+0980 to U+09FF (2432 to 2559)
Gurmukhi / Gurmukhi	Gurmukhi	U+0A00 to U+0A7F (2560 to 2687)
Gujarati / Gujarati	Gujarati	U+0A80 to U+0AFF (2688 to 2815)
Oriya / Oriya	Oriya	U+0B00 to U+0B7F (2816 to 2943)
Tamil / Tamil	Tamil	U+0B80 to U+0BFF (2944 to 3071)
Telugu / Telugu	Telugu	U+0C00 to U+0C7F (3072 to 3199)
Kannada / Kannada	Kannada	U+0C80 to U+0CFF (3200 to 3327)
Malayalam / Malayalam	Malayalam; Sinhala	U+0D00 to U+0DFF (3328 to 3583)
Thai / Thai	Thai	U+0E00 to U+0E7F (3584 to 3711)
Laotisch / Lao	Lao	U+0E80 to U+0EFF (3712 to 3839)
Tibetanisch / Tibetan	Tibetan; Myanmar	U+0F00 to U+109F (3840 to 4255)
Georgisch / Georgian	Georgian	U+10A0 to U+10FF (4256 to 4351)
Koreanisch(Hangul) / Hangul	Hangul Jamo	U+1100 to U+11FF (4352 to 4607)

Fig. 5-18: Unicode subranges

Name for Unicode subrange in software program	**Unicode chart name**	**Unicode character code range taken into account by the programming software**
	Ethiopic; Cherokee; Unified Canadian Aboriginal Syllabics; Ogham; Runic; Tagalog; Hanunóo; Buhid; Tagbanwa; Khmer; Mongolian; Limbu; Tai Le; Khmer; Symbols; Phonetic Extensions; Latin Extended Additional	U+1200 to U+1EFF (4608 to 7935)
Griechisch erweitert / Greek Extended	Greek Extended	U+1F00 to U+1FFF (7936 to 8191)
Interpunktion und Symbole / Punctuation and Symbols	General Punctuation; Superscripts and Subscripts; Currency Symbols; Combining Diacritical Marks for Symbols	U+2000 to U+20FF (8192 to 8447)
Symbole 2 / Symbols 2	Letterlike Symbols; Number Forms; Arrows	U+2100 to U+24FF (8448 to 9471)
Symbole 3 / Symbols 3	Box Drawing; Block Elements; Geometric Shapes	U+2500 to U+25FF (9472 to 9727)
Symbole 4 / Symbols 4	Miscellaneous Symbols	U+2600 to U+26FF (9728 to 9983)
Mathematik 1 / Mathematical 1	Dingbats; Miscellaneous; Mathematical Symbols-A; Supplemental Arrows-A	U+2700 to U+27FF (9984 to 10239)
Braille Muster / Braille Patterns	Braille Patterns	U+2800 to U+28FF (10240 to 10495)
Symbole 5 / Symbols 5	Supplemental Arrows-B; Miscellaneous Mathematical; Symbols-B	U+2900 to U+29FF (10496 to 10751)
Mathematik 2 / Mathematical 2	Supplemental Mathematical Operators	U+2A00 to U+2AFF (10752 to 11007)
Symbole 6 / Symbols 6	Miscellaneous Symbols and Arrows	U+2B00 to U+2BFF (11008 to 11263)
CJK-Radicals	CJK Radicals Supplement	U+2C00 to 2EFF (11264 to 12031)
KangXi Radicals	KangXi Radicals; Ideographic Description Characters	U+2F00 to 2FFF (12032 to 12287)
CJK Symbole / CJK Symbols	CJK Symbols and Punctuation	U+3000 to 303F (12288 to 12351)
Hiragana	Hiragana	U+3040 to 309F (12352 to 12447)

Fig. 5-18: Unicode subranges

Name for Unicode subrange in software program	Unicode chart name	Unicode character code range taken into account by the programming software
Katakana	Katakana	U+30A0 to 30FF (12448 to 12543)
	Bopomofo; Hangul Compatibility; Jamo; Kanbun; Bopomofo; Extended; Katakana Phonetic Extensions	U+3100 to 31FF (12544 to 12799)
CJK Zusätze / CJK Extras	Enclosed CJK Letters and Months; CJK Compatibility; CJK Unified Ideographs Extension A; Yijing Hexagram Symbols	U+3200 to 4DFF (12800 to 19967)
CJK Ideogramme	CJK Unified Ideographs	U+4E00 to 9FFF (19968 to 40959)
	Yi Syllables; Yi Radicals	U+A000 to ABFF (40960 to 44031)
Koreanisch (Hangul)	Hangul Syllables	U+AC00 to D7FF (44032 to 55xxx)
Spezial (Special)		U+D800 to U+F8FF
CJK-Kompatibilitäts-Ideogramme	CJK Compatibility Ideographs	U+F900 to FAFF (63744 to 64255)
Sonderformen 1 (Special forms 1)	Alphabetic Presentation Forms; Arabic Presentation Forms-A	U+FB00 to FDFF (64256 to 65023)
Sonderformen 2 (Special forms 2)	Variation Selectors; Combining Half Marks; CJK Compatibility Forms; Small Form Variants; Arabic Presentation Forms-B; Halfwidth and Fullwidth Forms; Specials	U+FE00 to U+ FFFF (65024 to 65535)

Fig. 5-18: Unicode subranges

5.12.8 Supplementary Functions, Status Information

Specify a variable name for every status element you want to use in the current project.

Image of screen number:

The image of the screen number corresponds to the number of the screen currently shown by the operating device.

Read coordination byte:

The Read coordination byte consists of 8 bits whose states are written by the operating device and read by the controller at cyclic intervals.

Table index:

If tabular data can be displayed and edited on the operating device, you have the possibility of receiving the current position of the cursor in the table as the table index. In this case, the row number is stored in the variable for the table index during the next polling procedure.

Keyboard image:

In a chain of bytes, there is one bit that displays the status of each key of an operating device. If the bit for a key is set to logical 1, this means that the key is pressed. Once the key is released, the bit is set to logical 0 again.

To enable the keyboard image to be read, request code 7FFCh must first be written into the cyclical polling area. The operating device then writes the current keyboard image into the agreed variable in the controller.

Each operating device can have a specific number of keys and therefore has its own keyboard image.

Image of language number:

The image of the language number corresponds to the screen number as an integer value.

Fractional digit control:

The variables for the fractional digit control enable you to define the number of decimal places (fractional digits) to be used to change the representation of decimal numbers. This variable is read once when you boot the operating device and can not be refreshed during runtime!

Example:

A decimal number has 7 digits: Two of these digits are defined as global fractional digits + offset. The value -12345 is to be displayed.

- Fractional digit control 0 = Result -123.45
- Fractional digit control 1 = Result -12.345
- Fractional digit control 2 = Result -1.2345

See chapter “Image of the Screen Number“ on page 6-251.
See chapter “Read Coordination Byte“ on page 6-244.
See chapter “Send Keyboard Image to Controller“ on page 6-232.
See chapter “Image of the Keyboard“ on page 6-251.

5.12.9 Supplementary Functions, Data Input

Screen change:

Activate the parameter **Allow screen change with active editor** if, despite having an active editor e.g. for a variable value, you want to exit from the current screen without first withdrawing the data release.

You can not normally change to another screen while an editor, e.g. for a variable value, is still active. You must first withdraw the data release before changing screens.

Input variables:

Activate the parameter **Edit reverse** if the element that you are currently editing is to be displayed using an inverse color assignment.

Use the parameter **Change input variable with Enter key** to ensure that the editor automatically changes to the next input variable of the screen after you have entered the data and pressed Enter or an equivalent key.

Table editor:

Choose whether the table editor is to advance **row-by-row** or **column-by-column** after you press Enter.

5.12.10 Supplementary Functions, Screensaver

For operating devices that support the screensaver function, enter the wait time. If no operation has occurred by the time this period has expired, the screensaver is activated.

As an option, you can choose

- whether you want to use a screensaver.
- whether you only want to use the screensaver for screens without cyclical variables.
- whether you want to allow the screensaver for screens with cyclical output variables.

5.13 Resources

Elements that you can reuse and apply to different projects are described as resources. These elements are not assigned to a language and should therefore, if possible, not contain any language-specific parts, e.g. a graphic should not contain any text.

5.13.1 Images and Symbols

Images:

You can use images as:

- Background images in screens
- Content for image lists that represent variable values
- Icons for internal error messages

- Images for buttons
- A frame for buttons

You can import images or insert them as objects (OLE).

All images that you create are available in the programming software at the location where you can work with images.

If you insert an image as an embedded object (OLE), it is associated with the image creation program. Double-click the embedded object to open the application program that you used to create the object. You can now edit the object directly. The system enters the changes to the object into the programming software after you close the application.

Icons:

Icons are images with two colors and are available for touch screen operating devices.

The advantage of using icons is that you can customize the foreground and background color. If icons are used in buttons, the colors of the icon are determined depending on the color settings of the button (for example, you can program a button to change color when it is pressed).

In addition, you can reuse icons conveniently since the color is determined by the button or the background.

5.13.1.1 Inserting New Images

Enter a name for the new image into the **Image Name** field.

Decide if you want to insert the image as an OLE object or insert the image from a file.

Images inserted as OLE objects can be edited in the server program by simply double-clicking the image.

If the image is inserted from a file, the link to the original image is lost. The programming software saves a copy of the image which you can not edit.

Follow the steps below to insert an image from an existing file:

1. Enter a name for the image into the **Image name** column.
2. Complete the entry with Enter.
3. In the **Insert new image** dialog box, select the **Insert image from a file** button.

The **Open** dialog appears.

4. Navigate to the desired file.
5. Click the file, to select it.
6. Click **Open** to confirm your selection.

On the right side of the window, the image is now displayed in original size.

You can use the sizing handles to resize the image. The new numeric values for height and width are automatically entered in the table on the left. You can also enter values in the table and view the resulting image displayed on the right.

The image is completed and you can now insert another image into the table.

Follow the steps below to use another program to insert a new image (OLE). The image file already exists.

1. Enter a name for the image into the **Image name** column.
2. Complete the entry with Enter.
3. In the **Insert new image** dialog box, select the **Insert image as OLE object** button.

The **Insert object** dialog appears.

4. Select the radio button **Create from File**.
5. Click **Browse**.

The **Browse** dialog appears.

6. Navigate to the desired file.
7. Click the file, to select it.
8. Click **Open** to confirm your selection.

You are returned to the Insert Object dialog.

9. Confirm with **OK**.

The image is now displayed on the right side of the window using the set width and height.

Follow the steps below to use another program to insert an image (OLE). The image has not been created yet.

1. Enter a name for the image into the **Image name** column.
2. Complete the entry with Enter.
3. In the **Insert new image** dialog box, select the **Insert image as OLE object** button.

The **Insert object** dialog appears.

4. Select the **Create new** radio button.
5. Select an image application.
6. Click **OK**.

The image application opens.

7. Create an image in the image application.
8. Close the image application by selecting **Exit and return to OLE image** in the **File** menu.

The image is now displayed on the right side of the window using the set width and height.

5.13.2 Image Lists

An image list is a table which assigns images to numerical values. It is used wherever an image is to be displayed in a screen instead of a numerical value from the PLC.

The window for image lists consists of two sections. The left section of the window displays a table, the right section displays a blank screen.

The screen located on the right allows you to specify the size of the image, and to control the representation in the process.

The table on the left consists of four columns and at least two rows.

The first row of the column Value contains the word Default. "Default" indicates, that the image in this row is displayed as long as no valid value is read from the controller. Otherwise, enter the numerical value in the column Value that the controller must transmit to the terminal for the corresponding image to be displayed.

5.13.2.1 Inserting a New Image

The following example is used to illustrate how to insert an object (an image in this case).

All types of objects can be inserted including documents, tables or charts, etc.

1. Enter a name for the image into the Image Name column.
2. Click Enter to complete your entry.
3. Click the program want to use to create an object (image) (example: CorelDraw (1)).
4. Click **OK** to confirm your selection.
5. Create the object using the selected program.
6. Return to the programming software (using the menu item which exits the program)

On the right side of the window, the image is now displayed in screen size.

You can use the sizing handles to resize the image. The new numeric values for height and width are automatically entered in the table on the left. You can also enter values and view the result on the right side.

The image is completed and you can now insert another image into the table.

(1) CorelDraw is a registered trademark of the Corel Corporation

5.14 Project Management

The project management function allows you to assemble the elements from the communication and language categories that are to be compiled as a project. The name of the Conposer file is used for the name of the project automatically.

5.14.1 Project Management, Activating a Project

If you have created more than one project, you must activate the project that you currently want to work with.

Follow the steps below to activate a project, possibility 1:

1. Mark the **Project management** branch in the project folder.
2. Select the name for the active project in the **Active project** area of the **Properties** window.

Follow the steps below to activate a project, possibility 2:

1. Using the right mouse-button, click the name of a project.
2. Select **Activate** from the context menu.

The name of the activated project is then shown in **bold** print.

5.14.2 Project Management, Languages

Language selection:

Select the languages that are to be included in the current project. The list on the left displays all languages contained in the current project folder. The list on the right displays the languages contained in the current project.

> copies the selected language(s) into the **active languages** list.

>> copies all languages into the **active languages** list.

< removes the selected language(s) from the **active languages** list.

<< removes all languages from the **active languages** list.

Start-up language:

The operating device must start with one language. Select the start-up language from the list.

5.14.3 Project Management, Communication

Select a protocol that the operating device can use to communicate with the controller.

> copies the selected protocol into the **active protocol** list.

< removes the selected protocol from the **active protocol** list.

5.14.4 Project Management, Terminal File

For the terminal file the name of the current project is used automatically. A terminal file is the result of the compilation process and has the extension CB.

Load this file into the terminal. You can carry this out in different ways:

- by CompactFlash card
- by USB stick
- by ethernet

Click on the **Download** button to open the dialog of the transmission.

5.14.4.1 Create INI File And Transfer Files

Along with the terminal file (.CB), also transfer the protocol driver (.DLL) and the runtime (.EXE). The storage location settings in the terminal is predefined but can be changed as required.

The default is **file:\\\FlashDrv\TSvisRT\Project name\**.
For each further project a directory with the name of the project is set under **..\TSvisRT\**.

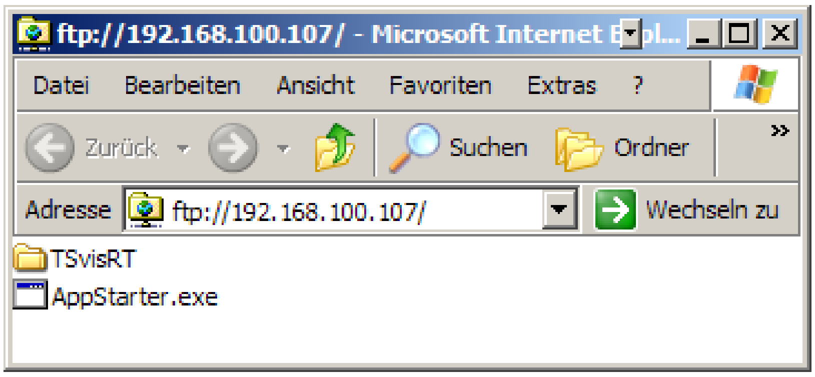
- If you enter a **file address** (file:\\\FlashDrv\...), the files are stored in this folder and the operating device accesses the files in its own Flash memory.
- If you enter a **URL address** (http:///MyServer.NET/...), the files are stored on this server and the operating device accesses the centrally stored files via the Ethernet.

If you want to **transfer the files to a storage medium**, select the relevant disk drive from a list of available disk drives.

Start the transfer by clicking the **Start** pushbutton.

The **TSvisRT** subdirectory is automatically created on the storage medium and the files are stored there.

For operating devices with integrated ftp server you have the possibility to display the content of the internal flash file system by means of a browser.



TSvisRT
AppStarter.exe

Fig. 5-19: Flash file system in the operating device (example)

6 Tutorial

The tutorial describes in detail the functions offered by the operating devices. By using the step-by-step instructions, you will quickly learn how to program the function described.

6.1 Working with Program Call Parameters

You can start VI Composer from the console and submit call parameters. You use the following syntax for it:

[Program directory]Composer.exe [–n/N [Template directory]Template name] [-o/O Output directory] [Project directory]Project name

☞ Directory names which contain blanks must be put into apostrophes! For example: "C:\Programme\rexroth\composer 2.1 [EN]"

Parameter	**Meaning**
[Program directory]Composer.exe	Name of the directory where the Composer.exe file is located.
[–n/N [Template directory]Template name]	A new project is generated from a template. You have to assign the template file with the complete directory, but without the file extension. Without a directory name the current directory is used.
[-o/O Output directory]	Name of the directory where the output file is written to.
[Project directory]Project name	Name of the directory where the project is written to with the assigned name. You have to enter the name of the project without the file extension. Without a directory name the current directory is used.

Fig. 6-1: Call parameters

Example 1:

[Program directory]Composer.exe

VI Composer starts.

In case of having already generated a project and having checked the checkbox **load automatically the last opened file** in the global settings this project is opened.

Example 2:

[Program directory]Composer.exe [Project directory]Project name

VI Composer starts with the assigned project.

If the assigned project is not existing VI Composer starts and shows an error message.

Example 3:

[Program directory]Composer.exe [-o/O Output directory] [Project directory]Project name

VI Composer starts with the assigned project.

If the assigned project is not existing VI Composer starts and shows an error message.

The subdirectory "TSvisRT\Project name" is generated automatically. In this directory the output files (TSvisRT_CE.exe, TSvisRT.ini, SPSTxxxx.dll, Projectname.CB) are copied after compilation of a VCPxx.2 project.

Example 4:

[Program directory]Composer.exe [–n/N [Template directory]Template name] [Project directory]Project name

VI Composer starts and generates a new project from the assigned template.

If the assigned template is not existing VI Composer starts and shows an error message.

The name of the project is automatically used as the project name in the project management, for the application ID and for the output file (Projectname.CB).

Example 5:

[Program directory]Composer.exe [–n/N [Template directory]Template name] [-o/O Output directory] [Project directory]Project name

VI Composer starts and generates a new project from the assigned template.

If the assigned template is not existing VI Composer starts and shows an error message.

The name of the project is automatically used as the project name in the project management, for the application ID and for the output file (Projectname.CB).

The subdirectory "TSvisRT\Project name" is generated automatically. In this directory the output files (TSvisRT_CE.exe, TSvisRT.ini, SPSTxxxx.dll, Projectname.CB) are copied after compilation of a VCPxx.2 project.

6.2 Working with Screens

6.2.1 Screen Structure

Screens with specific functions form the basic components of the screen structure.

System screens:

- Setup screen
- Start-up screen
- Password Screen
- Main screen

User screens:

- Input/output screens

The screen structure is made up of a network of input and output screens. There is no hierarchy. Input and output screens are located at nodes of the network. These screens contain a selection field from which you can choose the names of other screens.

In each input/output screen you can use control keys, function keys and buttons to access all other screens.

You can select different system screens for each language of a project.

6.2.2 System Screens

System screens are based on the input/output screen type. Some restrictions apply, due to the compulsory initialization phase and the fact that no communication has yet taken place with the controller.

System screens facilitate programming, and allow the system to become directly operable. In this way, the initialization phase becomes a fixed component of the project.

You can select any screen as the system screen.

As all screens are created on a language-specific basis, you can define other screens as system screens for each language.

As no communication has yet been established with the controller during the initialization phase, the following restrictions apply to system screens.

- The setup screen and startup screen can not be accessed by selecting a screen externally
- No controller variables can be displayed on the setup screen and startup screen

6.2.2.1 Setup Screen

You can only access the setup screen if you select the **Enter** key or the Setup button during the initialization phase (the startup screen is displayed for five seconds).

On the setup screen, it may be useful to include the following functions:

- Activate/deactivate download function
- Set protocol parameters
- Set date and time
- Display firmware level of the operating device

If you program these functions using system variables, the operator can select the corresponding parameter from text lists.

Password Protection for Setup Screen

A special procedure applies to password protection on the setup screen.

If you set the system variable **ScrchgPasswd** as the first editable variable on the setup screen, you can enter the password independently of the access level (exception 255). This also allows you to set password protection for the setup screen.

For the setup screen, the access level only impacts at the edit level, meaning the content is always visible to the operator.

Suppressing the Setup Screen

You can choose to hide the setup screen if you do not need to display it.

To do this, set the access level for this screen to the value 255. The setup screen can then not be accessed from the startup screen (using the Enter key or the Enter button).

6.2.2.2 Startup Screen

The startup screen appears for about five seconds after you switch on the operating device. This time is fixed, and this can not be changed.

On the startup screen, you can only display static texts and system variables. Due to the time sequence used, it is not possible to enter variables here.

When the startup screen is being displayed, you can press the Enter key to go to the setup screen. You can not go to the setup screen if the access level of the setup screen is set to 255.

On the startup screen, you can display the following information, for example:

- Service address
- Machine type
- Program version

6.2.2.3 Password Screen

In a project in which specific screens or variables are to be protected from unauthorized access, you must create a password screen.

In your password screen, you must create the system variable **ScrchgPasswd**.

Select the name of the password screen in the language properties, to activate the password screen.

You can create your own password screen for each language used in the project.

See chapter “Password Screen and Password Functions“ on page 6-155.
See chapter “ScrchgPasswd“ on page 6-196.

6.2.3 Input/Output Screen

The user interface of a project is primarily made up of input and output screens. You can display the following contents in these screens:

- Static text
- Text fields
- Variables
- System variables
- Background images
- Set of curves (graph)

- Buttons
- Recipe fields
- Table fields
- Message fields

You can also assign the following functions to an input/output screen:

- Sub-screens
- Help screens
- Background color
- Key functions (softkey function)

6.3 Working with Screen Objects

You can use the following objects to configure a screen. You can use a wide range of parameters to adjust these objects to suit your requirements.

6.3.1 Static Text

Static texts are displayed using a font of your choice. Depending on the operating device that you are using, you can set the following static text attributes:

- Inverse
- Flashing
- Underline
- Font
- Dynamic attributes
- Foreground color
- Background color.

6.3.2 Text Field

It makes sense to use a text field if you want to insert more than two words as contiguous text in a screen. A wizard will guide you step-by-step through the individual dialogs required to create a text field.

6.3.2.1 Text Field (Position / Size / Color)

Specify the following parameters for the text field: position, dimension, text spacing and text size. The unit for these specifications is pixels.

Furthermore, you can select the foreground and background colors and define dynamic attributes.

The preview for the text field becomes active only when you modify an existing text field. Select the **Display** check box to activate this function.

If you have created a new text field, click the **Next >** button.

Icon	**Description**
→	Enter the distance of the text field from the left edge of the screen.
↦	Enter the horizontal dimension of the text field.
↓	Enter the distance of the text field from the upper edge of the screen.
↧	Enter the vertical dimension of the text field.
→	Enter the distance of the text from the left edge of the text field.
↦	Enter the horizontal dimension of the text.
↓	Enter the distance of the text from the upper edge of the text field.
↧	Enter the vertical dimension of the text.
■	Select a foreground color.
□	Select a background color.

Fig. 6-2: Position, size and color of the text field

☞ See chapter “Dynamic and Static Attributes“ on page 6-63.

6.3.3 Variables

All operating devices support standard usage variable types. The connected controller determines the number of variable types permitted.

The variable type determines the range of values and the number of significant places.

Type	**Size**	**Range of Values**
Bit	1 bit	0, 1
Byte	1 Byte	–128 to +127
Byte	1 Byte	0 to 255
Word	2 Bytes	–32768 to +32767
Word	2 Bytes	0 to 65535
LWord	4 Bytes	–2147483648 to +2147483647
LWord	4 Bytes	0 to 4294967295
LWord	4 Bytes	±1,2 x 10-38 to ±3,4 x 10+38
ASCII	42 Bytes	0 to 255

Fig. 6-3: Variable types

In the programming software, you define a variable as a screen element. The screen element Variable is made up of the:

- Symbolic name
- Controller address
- Representation type
- Field type
- Field length
- Format
- Documentation value
- Limits
- Scaling
- Communication type
- Access type
- Editor
- Variable type
- Attributes (static or dynamic)
- Character set
- Help screen.

See chapter “Working with System Variables“ on page 6-155.

6.3.3.1 Symbolic Names

In the programming software, you assign a symbolic name to each variable. This name can have up to 255 characters.

6.3.3.2 Controller Address

You use the controller address to specify the storage location in the controller.

Depending on the protocol selected, the system carries out a syntax check. To avoid incorrect input, you can call a syntax diagram for each protocol in the online help for the programming software.

Note whether a variable will be accessed on a byte, word, or double-word basis.

6.3.3.3 Representation Type

You can choose from the following options for displaying variables on the operating device

- Decimal number
- Alphanumeric
- Selection text
- Selection image
- Floating point number
- Hexadecimal number
- Binary number
- Bars
- Curves

Decimal Number

In the **Field Type** field, define whether the variable is an input or output variable. If **Password** is selected, output is suppressed on the operating device or replaced by an asterisk (*).

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

The representation **format** comprises four main details:

1. The field length, that is, the total length including signs and decimal point.
2. The number of fractional digits, that is, the places that should be displayed after the decimal point (for example, two places for currencies).
3. The restriction to permit positive values only.
4. The option of filling digits that can not be used up by a low value (with few digits) with zeros.

In case of input variables you can enter upper and lower limit values for the **range monitoring** (area supervision) which are not to be exceeded. Assign a color to every limit value to indicate that the upper or the lower value has been exceeded.

The **Scaling** function allows you to adjust the input value to meet specific conditions:

- Factor
- Divisor
- Addend.

In the **Variable type** area, you define how the variable value is interpreted:

- Standard type
- Timer
- Counter or
- BCD Number.

Select an **Editor** that is to be used to enter the values of an input variable:

- By using the numeric keys (standard editor) or
- Using the PLUS and MINUS keys only (Increment editor) or
- Using both variants (Mix-mode editor).

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing Enter only
- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

If you want to start a script after entering the variable value, select the name of a script from the **Post editing script** field.

Standard

The significance of the displayed digits increases from right to left. You can display places either with leading zeros and/or a decimal point. The representation refers to the data types bit, byte, word, and Lword. The maximum length depends on the data type. There are no blanks between the characters. The variable appears in the controller either in binary format or in special timer or counter formats.

Example:

A decimal number with two decimal places.

10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	Significance
0	1	2	3	4	5	Displayed = 123,45

Key	Function
0 to 9	Enters the numbers 0 to 9.
Decimal point	Enters the decimal point.
Cursor Right	Moves the cursor one position to the right.
0 to 9	Enters the numbers 0 to 9.
Decimal point	Enters the decimal point.
Cursor Right	Moves the cursor one position to the right.
Cursor Left	Moves the cursor one position to the left.
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.
Plus	1st case: Variable is selected. The value is deleted and you can enter a new value. 2nd case: Cursor was moved within a positive value. The value is not changed. 3rd case: Cursor was moved within a negative value. The negative sign for the value is deleted.
Minus	1st case: Variable is selected. The value is deleted, and a negative sign is inserted at the least-significant position. You can enter a new value. 2nd case: Cursor was moved within a positive value. A negative sign is placed in front of the value. 3rd case: Cursor was moved within a negative value. The value is not changed.
Delete	Deletes the position where the cursor is located, and also deletes the sign.

Fig. 6-4: Key functions for decimal numbers of the type Standard

BCD Format

The significance of the displayed digits increases from right to left. You can display integers as BCD numbers with leading zeros.

The representation refers to the data types bit, byte, word, and Lword. The maximum length is 8 digits. There are no blanks between the characters. The variable appears in BCD format in the controller.

For one byte, the range of values spans from 00 to 99.

Example for a BCD number:

10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	Significance
0	1	2	3	4	5	Displayed = 1234_D

It is important that you take particular care when inputting BCD numbers.

You can enter numeric values as standard decimal numbers using the Standard editor. You can also increment or decrement each individual digit of the BCD value using the Mixmode editor. You can only use the Increment editor to make incremental changes to individual digits in a value with decimal transfer. This corresponds to the procedure used by a decade switch.

For scaled variables, the value in the controller changes by +/- 1. However, the value displayed also depends on the scaling specified.

Key	**Function**
0 to 9	1. Standard and Mixmode - Enter the numbers 0 to 9 2. Increment - No function
Decimal point	Enters the decimal point.
Plus	1. Standard - No function 2. Mixmode and Increment - Increments the value at the cursor and influences the more significant digits when the range of values is exceeded.
Minus	1. Standard - No function 2. Mixmode and Increment - Decrements the value at the cursor and influences the more significant digits when the range of values is not reached.
Cursor Right	Moves the cursor one position to the right.
Cursor Left	Moves the cursor one position to the left.

Fig. 6-5: Key functions for decimal numbers of the type BCD

Key	Function
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.
Delete	1. Standard and Mixmode - Variable is selected: The value is deleted and you can enter a new value. 2. Standard and Mixmode - Cursor was moved within the value: The character is deleted and the more significant digits are moved to the left. 3. Increment - No function

Fig. 6-5: Key functions for decimal numbers of the type BCD

Alphanumeric

In the **Field Type** field, define whether the variable is an input or output variable. If **Password** is selected, output is suppressed on the operating device or replaced by an asterisk (*).

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

If you want to start a script after entering the variable value, select the name of a script from the **Post editing script** field.

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing Enter only
- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

For the **format** of the variable, define the following information:

- Field length
- Display size and
- Display height.

The **field length** determines the data length to be stored in the controller.

The **display size** determines how many characters can be displayed at maximum width.

With **display height** you can reserve enough space to be able to display characters using larger font types.

Alphanumeric type variables function as follows:

For alphanumeric display, ASCII strings are read in byte format from the controller, and displayed in the operating device. The number of characters displayed varies, depending on the options offered by the operating device. A variable of the type Alphanumeric can not be longer than one display line. Longer texts are truncated.

The controller address specifies the start of the string. It does not contain a length byte, as this is not required.

You can use the plus and minus keys to input alphanumeric characters. The system variables Shift and ShiftCase are also available for upper case (Shift) and lower case (ShiftCase) respectively. You can use these keys to enter the additional characters displayed on the numeric keys.

To use the system variables, link the system variables as press and release variables with a function key to the screen. During input, the operator must press the function key and the corresponding numeric key.

You can use the Password field type to enable concealed password entry on the operating device. However, you can only enter numbers here. An “X” appears for each digit you enter.

Key	Function	With Shift	With Shift-Case
0	Enters the number 0	( ) ° 0	( ) ° 0
1	Enters the number 1	STU1	STUstu1
2	Enters the number 2	VWX2	VWXvwx2
3	Enters the number 3	YZ%3	YZ%yz%3
4	Enters the number 4	JKL4	JKLjkl4
5	Enters the number 5	MNO5	MNOmno5
6	Enters the number 6	PQR6	PQRpqr6
7	Enters the number 7	ABC7	ABCabc7
8	Enters the number 8	DEF8	DEFdef8
9	Enters the number 9	GHI9	GHIghi9
Decimal point	Enters the decimal point.	: ? ! .	:?!:?!.

Fig. 6-6: Key functions for alphanumeric variables

Key	**Function**	**With Shift**	**With Shift-Case**
Plus	Enters the numbers 0 to 9, the letters A to Z and a to z	<=>+	<=><=>+
Minus	Enters the numbers 0 to 9, the letters A to Z and a to z	\ * / –	*/*/-
Cursor Right	Moves the cursor one position to the right.		
Cursor Left	Moves the cursor one position to the left.		
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.		
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.		
Delete	Deletes the character at the cursor position.		

Fig. 6-6: Key functions for alphanumeric variables

See chapter “Working with Password Protection“ on page 6-153.

Selection Text

In the **Field Type** field, define whether the variable is an input or output variable.

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing Enter only
- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

For the **format**, you can specify the **field length** and **field height** of the selection text.

The field length also determines the maximum length for the text strings in the text list. If the field length differs from the length of the texts in the list, a dialog appears asking which length is valid.

The following **access types** are available:

- normal access
- selective access

In addition, the following options are available for the DIN measurement protocol:

- Article Administration
- Delete Article Administration

Normal access:

All of the texts in the list can be selected.

To do so, create a text list, fill it with the required texts and then link it to the variable that you are just about to create.

Selective access:

Only texts whose corresponding bit is set to logical 1 in the variant buffer are displayed.

For this access type, you need to create a variable for the variant buffer.

The variant buffer is read only once from the controller during the initialization phase of the operating device.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

Select a **text list** whose texts are to be displayed with the variable.

You can choose to display a text instead of a numeric value. To do this, you must create a text list.

In the text list, you assign numeric values to the corresponding texts.

The operating device reads the value of the variable from the controller, replaces the numeric value with text, and displays this text. If a value is read from the controller, and you have not defined a corresponding text for this value, the system displays a number of question marks.

If the Selection Text type is used for an input variable, you can choose to limit the field height to one or several lines.

If you specify a field height of 1, the system always only displays one text from the text list. If the field height is greater than 1, a correspondingly higher number of texts from the text list is displayed. The active text is displayed inversely.

Key	Function
0-9	No function
Decimal point	No function
Plus	Selection in ascending order (after the final value in the text list is reached, the value at top of the text list is selected next).
Minus	Selection in descending order (after the first value in the text list is reached, the value at the bottom of the text list is selected next).
Cursor Right	Moves the cursor one position to the right.
Cursor Left	Moves the cursor one position to the left.
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.
Delete	Deletes the character at the cursor position.

Fig. 6-7: Key functions for selection texts

Selection Image

In the **Field Type** field, define whether the variable is an input or output variable.

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing Enter only
- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

For the **format** you can specify the **field length** and **height** of the selection image.

The following **access types** are available:

- normal access
- selective access

In addition, the following options are available for the DIN measurement protocol:

- Article Administration
- Delete Article Administration

Normal access:

All of the texts in the list can be selected.

To do so, create a text list, fill it with the required texts and then link it to the variable that you are just about to create.

Selective access:

Only texts whose corresponding bit is set to logical 1 in the variant buffer are displayed.

For this access type, you need to create a variable for the variant buffer.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

Select an **image list** whose images are to be displayed with the variable.

Selection image type variables function as follows:

You can choose to display images instead of numeric values, in the same way as you can use text to represent numeric values. In an image list, first of all assign individual images to the numeric values. The numeric values do not need to be contiguous or sorted consecutively. Then, in a screen create a variable field for the selection image variable. In the dialog field for the representation type "Selection Image" link the variable with the image list. The corresponding image will then be displayed in the operating device, depending on the controller values used. The default image will be displayed for controller values that have not been specified in the image list.

Note that all of the images in an image list must be the same size, to ensure that they cover each other completely. Furthermore, make sure that the images used are not too large, to avoid slow display buildup.

You may need to modify the polling time accordingly.

Key	Function
0-9	No function
Decimal point	No function
Plus	Selection in ascending order (after the final value in the text list is reached, the value at top of the image list is selected next).
Minus	Selection in descending order (after the first value in the image list is reached, the value at the bottom of the text list is selected next).
Cursor Right	No function
Cursor Left	No function
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.
Delete	No function

Fig. 6-8: Key functions for selection images

Floating Point Number

In the **Field Type** field, define whether the variable is an input or output variable.

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

In case of input variables you can enter upper and lower limit values for the **range monitoring** (area supervision) which are not to be exceeded. Assign a color to every limit value to indicate that the upper or the lower value has been exceeded.

Enter the field length for the display **format**. The field length comprises the:

- Sign
- Decimal Point
- Signs in front of the decimal point and
- Signs after the decimal point (fractional digits).

There are a number of options to display the variable value with **fractional digits**:

Absolute:

Displays the value with a fixed number of fractional digits.

Global units - offset:

Displays the value with a variable number of fractional digits. The number of digits stored in a variable is subtracted from the specified number of fractional digits. Enter the variable into the **status information** of the **Fractional digit control** field.

Global units + offset:

Displays the value with a variable number of fractional digits. The number of digits stored in a variable is added to the globally defined number of fractional digits. Enter the variable into the **status information** of the **Fractional digit control** field.

If you want to start a script after entering the variable value, select the name of a script from the **Post editing script** field.

The **Scaling** function allows you to adjust the input value to meet specific conditions:

- Factor
- Addend and
- Form reciprocal value.

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing Enter only
- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

Select an **Editor** that is to be used to enter the values of an input variable:

- By using the numeric keys (standard editor) or
- Using the PLUS and MINUS keys only (Increment editor) or
- Using both variants (Mix-mode editor).

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

Floating point type variables function as follows:

The significance of the displayed digits increases from right to left. The number can optionally be displayed with a decimal point. Scaling is only carried out using a factor. The operating device can also form the reciprocal value before display.

There are no blanks between the characters. In the controller, the variable appears in special floating point formats, for example, IEEE. Only some controllers support floating point numbers.

Key	Function
0 to 9	Enters the numbers 0 to 9.
Decimal point	Enters the decimal point.
Cursor Right	Moves the cursor one position to the right.
Cursor Left	Moves the cursor one position to the left.
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.
Plus	1st case: Variable is selected. The value is deleted and you can enter a new value. 2nd case: Cursor was moved within a positive value. The value is not changed. 3rd case: Cursor was moved within a negative value. The negative sign for the value is deleted.
Minus	1st case: Variable is selected. The value is deleted, and a negative sign is inserted at the least-significant position. You can enter a new value. 2nd case: Cursor was moved within a positive value. A negative sign is placed in front of the value. 3rd case: Cursor was moved within a negative value. The value is not changed.
Delete	Deletes the position where the cursor is located, and also deletes the sign.

Fig. 6-9: Key functions for floating point numbers

Hexadecimal Number

In the **Field Type** field, define whether the variable is an input or output variable.

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

Enter the field length for the display **format**.

If you want to start a script after entering the variable value, select the name of a script from the **Post editing script** field.

In case of input variables you can enter upper and lower limit values for the **range monitoring** (area supervision) which are not to be exceeded. Assign a color to every limit value to indicate that the upper or the lower value has been exceeded.

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing Enter only
- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

Hexadecimal number type variables function as follows:

The significance of the displayed digits increases from right to left. Hexadecimal numbers are displayed with the digits 0 to 9 and A to F in upper case, and with leading zeros.

The representation refers to the data types byte, word, and LWord. The maximum length is 8 digits. There are no blanks between the characters.

Example:

A hexadecimal number:

16^4	16^3	16^2	16^1	16^0	Significance
0	E	4	5	A	Displayed = 0E45AH

Key	Function
0 to 9	Enters the numbers 0 to 9.
Decimal point	No function
Cursor Right	Moves the cursor one position to the right.
Cursor Left	Moves the cursor one position to the left.
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.
Plus	Enters the characters 0 to 9 and A to F in ascending order.
Minus	Enters the characters 0 to 9 and A to F in descending order.
Delete	No function

Fig. 6-10: Key functions for hexadecimal numbers

Binary Number

In the **Field Type** field, define whether the variable is an input or output variable.

Additionally define, how the value is to be read:

- cyclically
- once or
- event-controlled.

To define the binary number representation format, you must specify the length of the field and the number of blank spaces in between (gap characters). You define the **format** of binary numbers by specifying the following information:

- Number of bits (maximum: 32)
- Number of gaps (maximum: 255)
- Field length (maximum: 7937) and
- Order: 01234567 (MSB-to-LSB) or
- Order: 876543210 (LSB-to-MSB)

You can insert blanks (gaps) between the individual digits to facilitate readability of binary numbers.However, depending on the operating terminal, this reduces the maximum number of places that can be displayed. The Field length field will tell you how many places are currently set once you have exited one of the two input fields using the tab key for example.

For the **data transfer**, you must also specify when a value is to be transferred:

- on pressing the Plus, Minus or Enter key or
- automatically on each change.

The PLC handshake procedure can be used for this process, if desired.

If you want to start a script after entering the variable value, select the name of a script from the **Post editing script** field.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

You use binary numbers to display individual bits, bytes, words, and Lwords. Select the number of bits and blank spaces for display. Both values are used to determine the entire field length.

There are a maximum of 32 bits for each variable. There can be no more than 255 blank spaces between the bits.

The significance of the displayed digits can be displayed in ascending order from either left to right or from right to left.

Example for displaying a binary number with or without blank spaces:

```
0 1 0 0              Blanks = 0
0   1   0   0        Blanks = 1
0     1     0     0  Blanks = 2
```

Key	Function
0 and 1	Enters the numbers 0 and 1.
2 to 9	No function
Decimal point	No function
Cursor Right	Moves the cursor one position to the right.
Cursor Left	Moves the cursor one position to the left.
Cursor Up	Moves the cursor to the next highest, editable variable in the display, and selects it. If the cursor is already positioned at the top-level variable, the lowest-level variable is selected.
Cursor Down	Moves the cursor to the next lowest, editable variable in the display, and selects it. If the cursor is already positioned at the lowest-level variable, the top-level variable is selected.

Fig. 6-11: Key functions for binary numbers

Key	Function
Plus	Enters the characters 0 and 1.
Minus	Enters the characters 0 and 1.
Delete	No function

Fig. 6-11: Key functions for binary numbers

Bars

In the **Field Type** field, define how the value is to be read:

- Cyclically
- Once or
- Event-controlled.

For the **Expansion** of the bar, choose whether it is to run **horizontally** or **vertically**.

For the **Representation** of the bar, you can choose between pre-defined templates (patterns) or your own graphics. This allows you to choose the appearance of the following bar elements:

- Bar
- Background
- Limit value not reached and (value fallen below)
- Limit value exceeded.

Bar:

The bar graphic is used to display the actual bar. If you select a graphic of a silo, for example, the bar is displayed as a silo.

Background:

The background graphic is used to display the area not yet covered by the bar.

Limit value not reached (value fallen below):

The bar is displayed using the 'limit value not reached' graphic if the value of the variable falls below the value of the 1st limit value.

Limit value exceeded (value exceeded):

The bar is displayed using the 'limit value exceeded' graphic if the value of the variable rises above the value of the 2nd limit value.

You can influence the **format** of the bar by specifying the:

- Width
- Height
- Reference value
- 1st limit value (1. Corner value)
- 2nd limit value (2. Corner value)

Width:

Enter the horizontal dimension of the bar in pixels.

Height:

Enter the vertical dimension of the bar in pixels.

Reference value:

The reference value is used to specify the variable value from which the bar is to grow. If you enter the name of a variable, this value will be calculated at runtime.

Example: The values from 0 to 100 are to be displayed as a horizontal bar.

1. For a reference value of 0, the bar grows from left to right as the values increase.
2. For a reference value of 50, the bar grows from the center, depending on whether the variable value lies above or below 50.

1st limit value (1. Corner value):

The 1st limit value determines the variable value from which the bar is to grow from the lower or left corner. If you enter the name of a variable, this value will be calculated at runtime.

Example: The values from 0 to 100 are to be displayed as a horizontal bar.

1. For a 1st limit value of 0, the bar grows from left to right as the values increase.
2. For a 1st limit value of 50, the bar only grows from left to right as of a variable value of 50. For variable values below 50, the bar is displayed with the 'limit value not reached' graphic.

2nd limit value (2. Corner value):

The 2nd limit value determines the variable value up to which the bar is to grow to the lower or right corner. If you enter the name of a variable, this value will be calculated at runtime.

Example: The values from 0 to 100 are to be displayed as a horizontal bar.

1. For a 2nd limit value of 100, the bar grows from left to right as the values increase.
2. For a 2nd limit value of 50, the bar grows from left to right up to a variable value of 50. For variable values above 50, the bar is displayed with the 'limit value exceeded' graphic.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

Bar-type variables function as follows:

You can use the representation type Bars only to **output** variable values.

The variable values are refreshed either cyclically or one time, when the screen is opened.

You use the height and width values to determine whether the bars run horizontally or vertically. From a particular reference point, the dimension of a bar can be:

- in a positive direction
- in a negative direction
- in both directions

Specify the width and height in the unit 'Character'. The entire bar can only ever accept the size of a multiple of a character. When the controller values are output, however, the bar changes its dimension by pixel size.

Use two limit values to define the range of values that a bar will display.

Use the first limit value to determine the value of the bar at the left or lower end.

Use the second limit value to determine the value of the bar at the right or upper end.

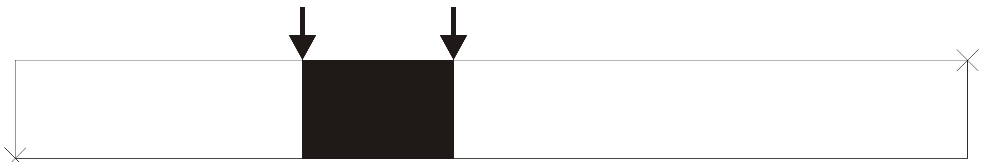


Fig. 6-12: Horizontal bars

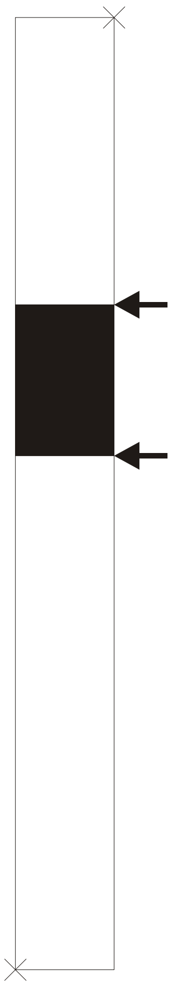


Fig. 6-13: Vertical bars

The range of values is limited to values from -32768 to +32767.

To display several bars in a screen, ensure that the controller addresses are consecutive and contiguous. This will speed up data transfer.

You can use four fill patterns for bars:

1. For the empty area of the bar (background).
2. For the filled area of the bar (foreground).
3. For the bar, if the lower limit value is not reached.
4. For the bar, if the higher limit value is exceeded.

The programming software contains four standard fill patterns. You can use any other images as fill patterns. Prior to use, you must import these images into the programming software or insert them as OLE.

Example of fill status display:

Four graphics have been created for the example. They display a container that is either empty or full. The word MIN is used to demonstrate a situation in which the value is not reached. And the word MAX to depict a situation in which the value is exceeded. The container in the middle depicts a container for a case where the operating device displays a midpoint controller value for the variables.

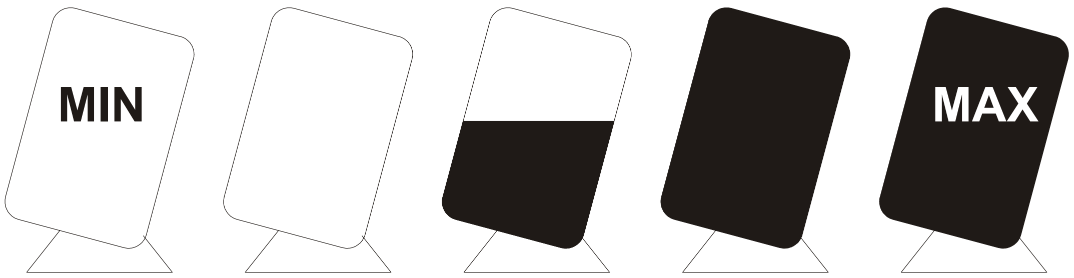


Fig. 6-14: Example of fill status display

You may only use values of the integer and unsigned integer data types for the display as a bar!

Curve

In the **Field Type** field, define how the value is to be read:

- Cyclically
- Once or
- Event-controlled.

You can influence the **format** of a curve by specifying the:

- Width
- Height

Width:

Specify the horizontal size of the bar in pixels.

Height:

Specify the vertical size of the bar in pixels.

The **documentation value** is a character string that fills the variable field in the screen. If the documentation value is shorter than the field, it is entered repeatedly.

Curve-type variables function as follows:

Use the representation type "Curve" to display a value table as a row of points in the operating device.

The address for the controller variable represents the start of the value table in the controller. Each value in the table describes one pixel of the curve.

A curve is defined by the following parameters:

- Maximum width (54 pixels for each curve variable)
- Maximum height (height of the display in the operating device)

Specify a length and height to determine the dimension of the curve in the unit 'Character'.

To produce a curve with a width of 54 pixels, several curve variables next to each other are required.

Insert a coordinate grid as a background image.

The operating device reads the variable values as an array from the controller, and inserts these as continuous consecutive height data. The value with the starting address (address +0) is displayed on the very left. Each subsequent piece of height data (address +n) is offset one pixel position to the right.

The height data for the curve is cyclically refreshed.

Example of a curve display:

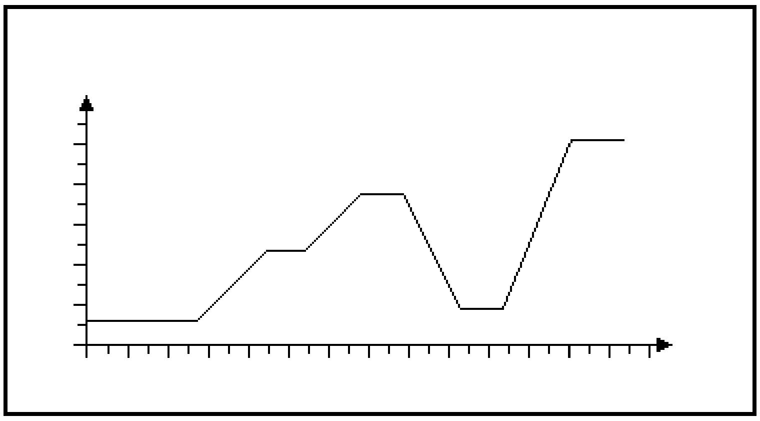


Fig. 6-15: Example for displaying a curve

6.3.3.4 Field Type

By selecting a field type, you determine whether the operator will be able to modify the variable's value or whether the value is just displayed.

For password entry, you can specify the field type in more detail.

Input

Select the field type Input to enable operators to change the value of a variable on the operating device.

The value of the variables is loaded from the controller when the screen is accessed.

If you select the attribute Cyclical, the system constantly updates the value of the variables, based on the interval specified in the polling time.

Before input, the operator must press the Data Release key. The operator can only change the value of the variables once the status LED for the data release is lit. Use the Enter key to write the value to the controller. The operator must then press the Data Release key. The status LED for the data release switches off.

Output

Select the field type Output to only display the value of the variables, but not allow the operator to change the value.

The value of the variables is loaded from the controller when the screen is accessed.

If you select the attribute Cyclical, the system constantly updates the value of the variables, based on the interval specified in the polling time.

Password

You can use the additional attribute Password to determine for an alphanumeric variable that the password is not visible on the operating device when it is entered.

Instead of displaying the values entered, the system displays the operator a string of “X“ when the password is entered.

Cyclical

The operating device always polls the controller for the value of a variable when the operator goes to a screen in which a value is to be displayed. However, to display actual values, the value must be continuously updated. Therefore, always select the field type Cyclical for displaying actual values.

The system will then continuously update the value of the variables, based on the interval set as the polling time.

6.3.3.5 Format

Only Positive

Use the attribute Only Positive to display variable values that are to be displayed without a sign. This means that the range of values that can be displayed changes, for example, for a byte, from between -128 and +127 to between 0 and 255.

You can also display positive decimal numbers with leading zeros.

Display Leading Zeros

For positive decimal numbers, you can display more significant digits with a value of zero as zeros.

For example:

If the field length is 5 digits, the number 25 is displayed with leading zeros as follows:

00025

Field Length

The field length of a variable is made up of:

- The sign
- The number of digits
- The decimal point

For the representation type Binary Number, the number of blank spaces is added to the number of digits, to determine the field length.

Fractional Digits

You can define the number of fractional digits for decimal and floating point numbers.

This does not change the field length, however, one digit for displaying the decimal point is lost.

6.3.3.6 Documentation Value

The documentation value for displaying a variable is a placeholder when you program with the programming software. Depending on the representation type you are using, the programming software specifies another documentation value, for example, "F" for hexadecimal numbers or "9" for decimal numbers.

The documentation value is also used for project documentation instead of a real controller value.

See chapter “Tools Menu, Documentation“ on page 5-21.

6.3.3.7 Limits

In the programming software, you can specify a lower and an upper limit for each variable to restrict operator input.

The lower limit is automatically set to 0 for variable values that are displayed with the attribute Only Positive.

If the operator tries to enter a value outside of these limits, one of the following terminal messages is issued:

- Value too small
- Value too large

The operator can ignore these terminal messages, but he must enter a value that lies between the limits, or use the Cursor Up or Cursor Down keys to go to another variable in the screen. The system then accepts the current controller value again.

6.3.3.8 Scaling

Scaled Input

To modify the values that the operator enters in the operating device in line with the values used in the connected controller, inverse scaling must be carried out.

The system uses the following formula to scale the input:

$$\textbf{Controller Value} = \frac{\textbf{Input Value of the Unit} - \text{Summand}}{\text{Factor}} \times \text{Divisor}$$

Fig. 6-16: Scaling of the input variables in the operating device

A rounding error may occur during scaling which is calculated using the following formula.

$$\left(\text{Input Value of the Unit} \times \text{Factor}\right) < \left(\text{Upper Limit} - \text{Divisor} / 2\right)$$

Fig. 6-17: Rounding of the input variables in the operating device

Scaled Output

You can scale output to modify the range of values to suit user interface requirements. The scaling data is used for both output and input in the operating device. This does not restrict the range of values for the variable. Scaling is only carried out in the operating device.

You use the following operands for scaling:

- Factor,
- Divisor and
- Addend.

Note that a factor or divisor with the value 0 is not permitted.

Operand	Range of Values
Factor	–32768 to –1, +1 to +32767
Divisor	+1 to +32767
Addend	–32768 to +32767

Fig. 6-18: Scaling decimal numbers

Operand	Range of Values
Factor	–999999999,99999999 to –0,00000001+0,00000001 to +999999999,99999999
Divisor	–999999999,99999999 to –0,00000001+0,00000001 to +999999999,99999999
Addend	–999999999,99999999 to +999999999,99999999

Fig. 6-19: Scaling floating point numbers

The operating device uses the following formula to scale the output:

$$\text{Output Value of the Unit} = \frac{\text{Controller Value} \times \text{Factor}}{\text{Divisor}} + \text{Summand}$$

Fig. 6-20: Scaling of the output variables in the operating device

Use the following formula to determine the operands.

$$\frac{\text{Current Controller Value} - \text{Lower Limit Controller Values}}{\text{Current Terminal Value} - \text{Lower Limit Output Values}} = \frac{\text{Upper Limit Controller Values} - \text{Lower Limit Controller Values}}{\text{Upper Limit Output Values} - \text{Lower Limit Output Values}}$$

Fig. 6-21: Scaling of the output variables

The following example will help you determine the operands.

Example:

Range of values for output values:
Lower limit for output value = 0
Upper limit for output value = 100
Current value in operating device = x
Range of values for controller values:
Lower limit for controller values = -4096
Upper limit for controller values = 4096
Current controller value = y

1. Inserting the variable values:

$$\frac{y - (-4096)}{x - 0} = \frac{4096 - (-4096)}{100 - 0}$$

$$\frac{y + 4096}{x} = \frac{4096 + 4096}{100}$$

Fig. 6-22: Inserting the variable values in the formula

2. Solving the equation:

$$100\,y + 409600 = 8192\,x$$

Fig. 6-23: Solving the equation

3. Solving the equation for x:

$$x = \frac{100}{8192}\,y + \frac{409600}{8192}$$

$$x = \frac{\boxed{100}}{\boxed{8192}}\,y + \boxed{50}$$

Factor
Summand
Divisor

Fig. 6-24: Solving the equation for x

6.3.3.9 Communication Type

PLC Handshake

Select the attribute PLC Handshake to inform the controller that the values of the **subsequent controller variables of the current screen** are to be changed.

To do this, you must:

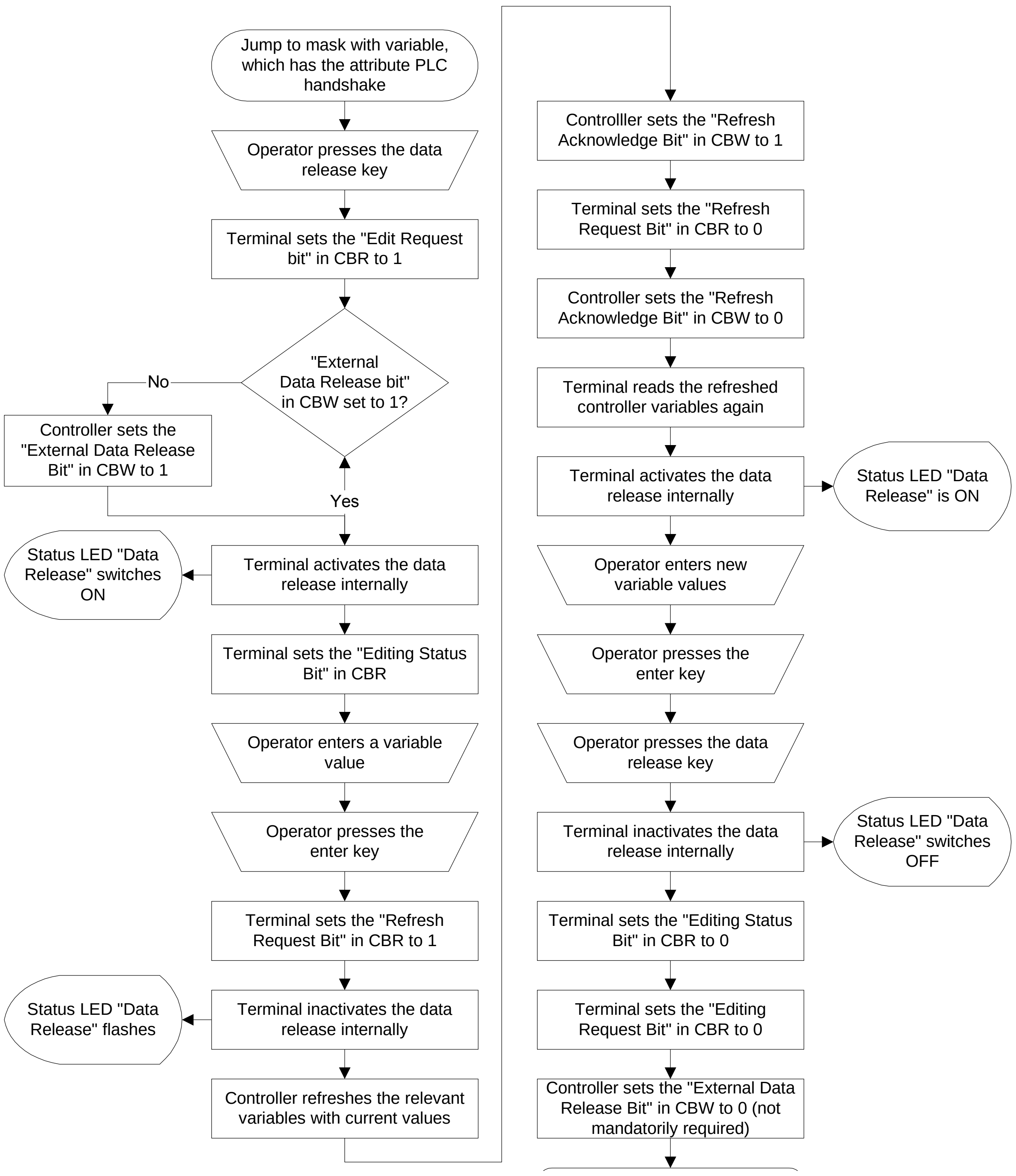
- Create a variable for the Read Coordination byte AND
- Create a variable for the polling area.

☞ See chapter “Read Coordination Byte“ on page 6-244.
See chapter “Write Coordination Byte“ on page 6-247.

The attribute PLC Handshake allows you to:

- Create your own recipe management system
- Inform the controller that a specific variable value will be changed

The PLC handshake process runs as follows:



Jump to mask with variable, which has the attribute PLC handshake
Operator presses the data release key
Terminal sets the "Edit Request bit" in CBR to 1
"External Data Release bit" in CBW set to 1?
No
Controller sets the "External Data Release Bit" in CBW to 1
Yes
Status LED "Data Release" switches ON
Terminal activates the data release internally
Terminal sets the "Editing Status Bit" in CBR
Operator enters a variable value
Operator presses the enter key
Terminal sets the "Refresh Request Bit" in CBR to 1
Status LED "Data Release" flashes
Terminal inactivates the data release internally
Controller refreshes the relevant variables with current values
Controlller sets the "Refresh Acknowledge Bit" in CBW to 1
Terminal sets the "Refresh Request Bit" in CBR to 0
Controller sets the "Refresh Acknowledge Bit" in CBW to 0
Terminal reads the refreshed controller variables again
Terminal activates the data release internally
Status LED "Data Release" is ON
Operator enters new variable values
Operator presses the enter key
Operator presses the data release key
Terminal inactivates the data release internally
Status LED "Data Release" switches OFF
Terminal sets the "Editing Status Bit" in CBR to 0
Terminal sets the "Editing Request Bit" in CBR to 0
Controller sets the "External Data Release Bit" in CBW to 0 (not mandatorily required)

Fig. 6-25: Flow diagram for PLC handshake

With Enter

The operator must press the Enter key to transfer the value of the variables from the operating device to the controller.

With +, – or Enter

Each time the operator presses the Plus and Minus keys, he transfers the incremented or decremented value to the controller. If the operator uses the 0 to 9 keys to enter the value, he must then press the Enter key.

For Each Change (Upon any modification)

The operator can change the value of a variable only with the Plus and Minus keys. The changed value is transferred to the controller each time you select the Plus and Minus keys.

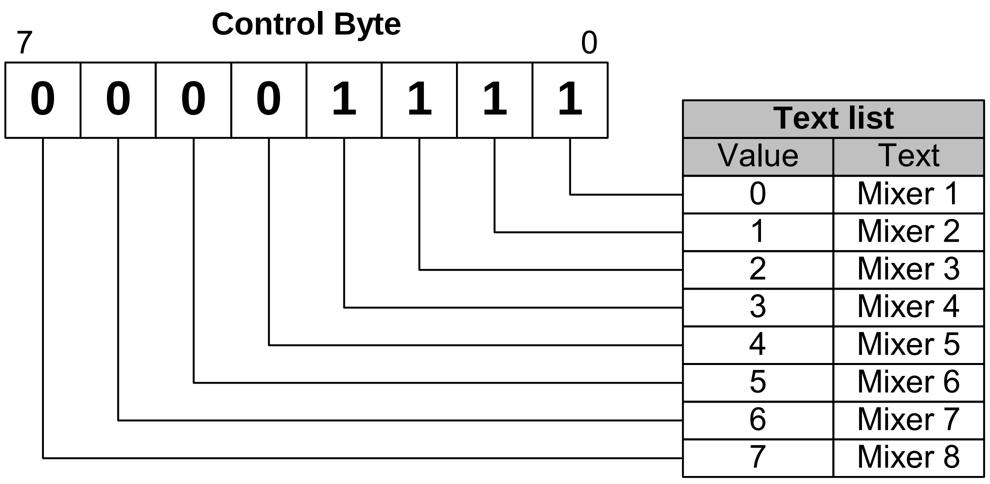
6.3.3.10 Access Type

Regular (Normal)

Use the access type **Regular** for accessing selection text, or selection image variables for projects that do not use any variant options. The system then displays for selection all entries in a text list or an image list.

Selective

Use the access type Selective to only display the selection texts or selection images that are “released“ using a controller variable. Each bit of the controller variable represents an entry in the text or image list.



Text list	
Value	Text
0	Mixer 1
1	Mixer 2
2	Mixer 3
3	Mixer 4
4	Mixer 5
5	Mixer 6
6	Mixer 7
7	Mixer 8

Fig. 6-26: Selective access

You can use the control byte depicted in the **Selective Access** image to only display the first four entries in the text list.

Enter the name of the controller variable in the property window of the text list or image list, respectively.

The controller variable is only read once, when the operating device is being initialized!

6.3.3.11 Variable Type

Standard

The standard type is the variable type most frequently used to display decimal numbers. The maximum length depends on the data type. The significance of the displayed digits increases from right to left. There are no blanks between the digits.

Significance	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}
Displayed 123,45D	0	1	2	3	4	5

BCD Number

A BCD number must be saved in the controller in BCD format. The operating device can interpret and display up to eight digits. The significance of the displayed digits increases from right to left. There are no blanks between the digits. The value can be displayed with leading zeros.

Significance	10^4	10^3	10^2	10^1	10^0
Displayed 1234D	0	1	2	3	4

6.3.4 Background Image

Background images are screen elements that are overlaid by every other element contained in the screen. You can display as many background images as you wish in a screen.

Select an image that has already been created using the **Images** function as a background image and assign another name and different attributes to the image for this function.

If background images are superimposed over one another, you can define the link principle by which the pixels are to be overlaid. You can use the SET, OR and XOR link options (depending on the terminal type).

With **SET** , the rear image is overlaid by the front image in such a way that the overlaid part of the lower image is no longer visible.

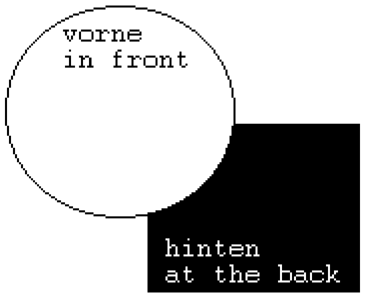


Fig. 6-27: Background image, SET-linked

Using **OR**, a link is created between the pixels where the logic is >inclusive or <.

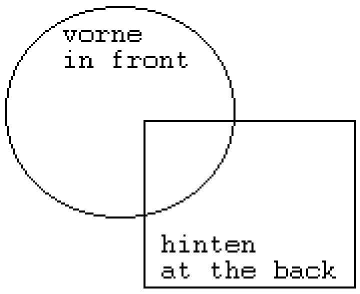


Fig. 6-28: Background image, OR-linked

Using **XOR**, a link is created between the pixels where the logic is >exclusive or<.

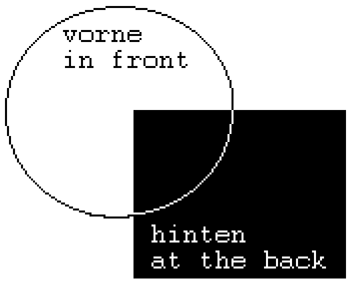


Fig. 6-29: Background image, XOR-linked

The layer number determines whether an image can overlay other images. The image with the lowest layer number is displayed in the foreground and is linked to the image with the next highest layer number. The link is always created in accordance with the settings of the background image with the smaller layer number.

6.3.5 Buttons

Buttons are graphic areas that are linked with a specific function. When you press a button, the preconfigured function is activated. This only applies to operating devices equipped with a touch screen. On operating devices with a full graphics display, buttons can only be used to display images, variables or texts.

A button can be broken down into the following:

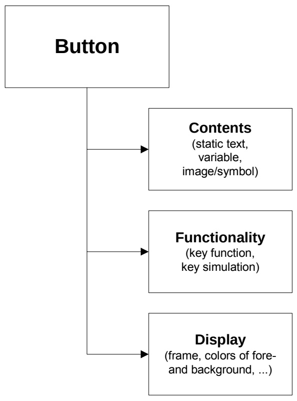

Button
Contents
(static text, variable, image/symbol)
Functionality
(key function, key simulation)
Display
(frame, colors of fore- and background, ...)

Fig. 6-30: Button

The button content, the functions and the representation (display frame, other attributes) can, for the most part, be programmed independently of each other.

6.3.5.1 Content of Buttons

A button can contain a static text, a text field, a variable, an image/symbols or nothing at all.

Buttons that have no content are displayed without a frame and are transparent screen elements. (Application: Transparent buttons are superimposed on a background image, for example, a plant overview to map “hot areas“ on an image. When you select this area, a specific action is carried out, for example, the system opens another screen).

6.3.5.2 Creating a Button

To create a button:

1. Click the Button icon in the toolbar.

Fig. 6-31: Tool symbol for button

2. While holding down the left mouse button, draw a frame on the screen and then release the left mouse button.

The buttons wizard opens and helps you by issuing a prompt for all the necessary parameters for a button.

Content:

- Empty: Button displays no content and is thus transparent. You can not have an empty button displayed with a frame or color.

- Text: Static text is displayed in the button. This text is edited in the Static Text tab.
- Text field: A piece of text (of multiple lines) is displayed on the button. This text is edited in the Text Field tab.
- Variable: A variable is displayed on the button (all variable and representation types are possible). If you select this option, the Variable Reference tab is shown as well as the Screen Variable tab.
- Image/symbol: An image (or symbol) is displayed on the button. This is edited in the Picture/Symbol tab.

Function:

- None: The button does not have a function.
- Pushbutton: Used to change screens or write a value to a system or controller variable.
- Switch: Used to write a value to a system or controller variable.
- Key simulation: Used to simulate any key in the system.
- Editor: Used to activate the edit screen for the input variable. This option is only available for selection if Variable was selected under Contents. You must then configure an input variable.

Position + Extension:

Enter the position and expansion (height and width) for the button in pixels.

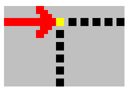
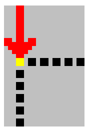
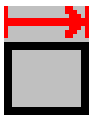
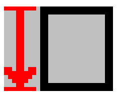
Symbol	Meaning
	Distance from the left border of the screen
	Distance from the top border of the screen
	Width of the button
	Height of the button

Fig. 6-32: Position and expansion for a button

Released condition:

Select colors for the foreground and background elements for when the button is not pressed.

Select the frame that should appear around the button in the same way.

If you are unable to select a frame, then no frame image is available. In this event, you can copy a frame image from the CD with the programming software or create your own bitmap.

Pressed condition:

Select colors for the foreground and background elements for when the button is pressed. Proceed in the same manner as described for released state.

Button background:

Check the checkbox **transparent** to represent the button transparency. Only buttons with the **Editor** function can be represented transparency. Thanks to this you have the possibility of showing a background image through the button.

Signal tone:

Choose the action or state that should trigger a beep tone.

Also specify the duration of the beep tone. You can manually enter the value in the input box or use the arrow keys to change the value in steps. The unit used to specify how long the tone should sound is seconds.

Dynamic attributes:

Click the **Dynamic attributes**button to assign various attributes to the button, depending on particular values.

Alignment of the variable:

To align the variable inside the button field check the apropriate checkboxes for horizontal and vertical alignment.

6.3.5.3 Functions of Buttons

A button can trigger the following functions:

- Open another screen
- Write a value (byte) to a PLC or system variable when you press the button
- Write a value (byte) to a PLC or system variable when you release the button
- Simulate any key
- Generate a free tone
- Activate or open the Editor for an input variable
- No action

Pushbuttons

As a pushbutton, the button can initiate a screen change or influence the value of a variable.

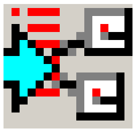
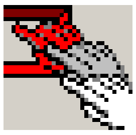
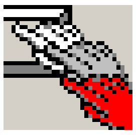
Symbol	Bedeutung
	Select the name for the screen to be used for a **screen change**.
	Enter the **value of the variable** when the button is **pressed** and type in the name of the variable.
	Enter the **value of the variable** when the button is **released** and type in the name of the variable.

Fig. 6-33: Pushbutton functions for button

The values for the variables must comply with the following table.

Configurable Val-ues	Default Value

0 to 255	Pressed = 1
	Released = 0

Fig. 6-34: Variable values for the Pushbutton function

You can also specify an access level for the button.

Enter a value from 1 to 255 for the access level and activate password protection for the button. This ensures that the function associated with the button is only executed once you have entered the correct pass-word.

Configurable Val-ues	Default Value
0 to 255	0 (password protection inactive)

Fig. 6-35: Access level

If you program a button with a screen change and a simultaneous vari-able change, this can trigger **critical** machine statuses that can no longer be stopped from the following screen!
Never use this combination in connection with manual machine opera-tions (e.g. teach-in).

Switch

The Switch function is comparable to that of a current surge relay (ELTACO). When the button is pressed, the other status is always restored.

Specify the following:

- Name of the status variable whose value is to be influenced
- Value of the variable when you press the button and
- Value of the variable when you release the button
- Access level for the button

Configurable Values	Default Value

0 to 255	Pressed = 1
	Released = 0

Fig. 6-36: Variable values for the Switch function

Enter a value from 1 to 255 for the access level and activate password protection for the button. This ensures that the function associated with the button is only executed once you have entered the correct password.

Configurable Values	Default Value
0 to 255	0 (password protection inactive)

Fig. 6-37: Access level

Key Simulation

The button takes over the function of a key.

You can simulate the following keys:

- 0 to 9
- A to Z
- Special characters
- Control keys
- Keys with special functions

Enter a value from 1 to 255 for the access level and activate password protection for the button. This ensures that the function associated with the button is only executed once you have entered the correct password.

Configurable Values	**Default Value**
0 to 255	0 (password protection inactive)

Fig. 6-38: Access level

Static Text for Buttons

In the **Text** box, enter the text that should appear on the button.

Choose the font, in which the text should be displayed, in the**Font** section.

To save a new font in the list of fonts available for selection, click the **New font**button.

Various attributes are available for the text depending on the touch screen panel type. Select any of the attribute check boxes to combine them as desired.

Text Field Properties

In the **Text** input box, enter the text you would like to have displayed on the button. The text will be automatically wrapped to the next line when you reach the right-hand frame of the button. A word is not split in this case but moved to the next line entirely instead. If words are too long, they must be split manually.

Blue arrows beside the input box indicate that the vertical expansion of the button is insufficient for displaying the entire text.

You can check how it will appear in the **Preview** box.For this purpose, select the **Display preview** check box.

Align (orientation) the text as follows:

- Left align
- Center
- Right align

Select the **Underline** check box to underline the text fully.

Select a font in the **Font** area.

Proceed as follows:

1. Click the button.
2. Highlight a font in the list.

To add a new font to the list of fonts, click the **New font** button.

Screen Variables for Buttons

In the **Representation Type** area, select how the variable values should be displayed. To configure the type more in more detail, click the **Edit Type** button.

If you have selected numerical or textual representation, you can choose a font under the **Representation** area.

To add a new font to the list of fonts, click the **New font** button.

Select the **Underline** check box to have the variable value or text fully underlined.

Select an existing help screen from the **Help Screen** area to provide the operator with help on the variable.

In the **Attribute** area, assign an access level to the variable. Given that several variables can appear in one screen, you can specify the order in which they are edited. This means that the cursor moves from one variable to another in this order.

The user presses the button to navigate in the corresponding list. For one-line selection text variables, pressing on the left half of the variable will result in a decrementation and, on the right, an incrementation. For multiple-line selection text variables, or selection image variables in general, pressing on the upper half of the variable results in a decrementation, and on the lower half, an incrementation.

One-line selection text

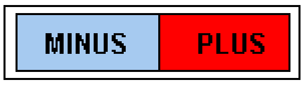


Fig. 6-39: Button with horizontal layout

Multiple-line selection text and selection image

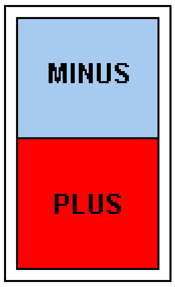


Fig. 6-40: Button with vertical layout

If you configure an input variable of type **selection text** or **selection image**, the Enter button is automatically created (prompt appears); that is, no edit screen is linked for these two types.

Variable Reference for Button

Select a variable from the tree view. Proceed as follows:

1. Double-click the folder to expand it and see the directories beneath.
2. Double-click the variable entry.

A red flag now marks the variable.

Use the following procedure to create a new controller variable:

1. Click the **Controller variables** folder.
2. Select **New PLC variable** from the context menu.
3. Enter the variable name and address of the variable into the dialog that appears.
4. Click **OK** to confirm your entries.

Note: You can not exit this page if you select a system variable that is in conflict with the settings specified in the **Button** tab.

You can not exit this page if you select a system variable that is in conflict with the settings specified in the **Button** tab.

Image/Symbol for Button

Select an image or symbol to be displayed on the button in the **Image / Symbol** area. The preview window on the right-hand side will help you to make this selection. To use the preview, select the **Display preview** check box.

To resize the button within the screen without changing the image:

Drag a sizing handle with the mouse to change the dimensions of the button as desired.

To resize the button in the screen and to resize the image at the same time:

Hold the Control key down and drag a sizing handle with the mouse.

6.3.5.4 Representation of Buttons

Attributes for displaying a button are:

- Foreground and background color for the pressed and released states
- Position and dimension of the button
- Frame for the pressed and released states

Note:

- When you program an input variable in the same way as for a keyboard-operated operating device, a button is automatically generated around the variable.
- Buttons can only overlap with background images.
- Buttons in tables that contain a variable have no frame and have one line.
- Buttons in input and output screens are not allowed to contain input variables.
- When you program a button with an input variable of the type Selection Text or Selection Image, the **Enter** button is automatically created (on request). In other words, no input or output screen is linked for both of these types. The user presses the button to navigate in the corresponding list. If the selection text variable only has one line, the variable value is decremented when you press the left half of the button, and incremented when you select the right half. For multiline selection text variables and for selection image variables in general, the variable value is decremented when you press the top half of the button, and incremented when you press the bottom half.

The following image depicts a button that has a horizontal layout.

This type is activated for selection text that only has one line.

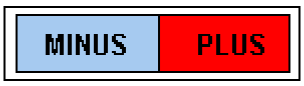


Fig. 6-41: Button with horizontal layout

The following image depicts a button that has a vertical layout.

This type is activated for selection texts and selection images with several lines.

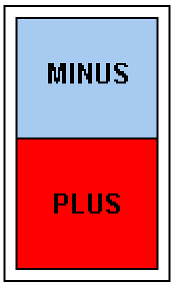


Fig. 6-42: Button with vertical layout

6.3.5.5 Frames for Buttons

The frame for a button is created as an image that is saved in the programming software as an image. This image is then made available by the programming software, and you can use it for button frames.

In the following example, a frame is created that depicts a button when it is not pressed (released state).

The image for this frame looks like this:

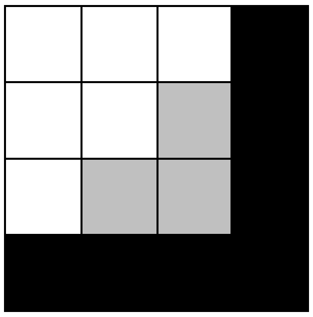


Fig. 6-43: Image for a basic frame

This image is made up of four subareas.

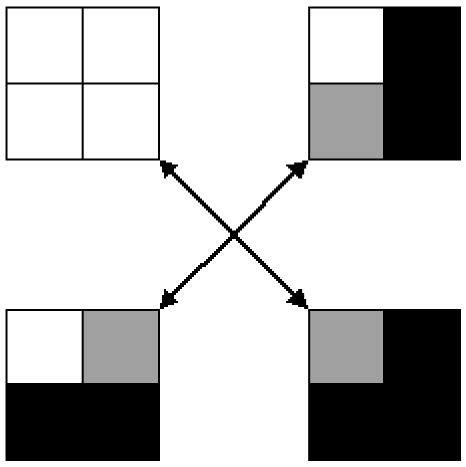


Fig. 6-44: Image split into four areas

The programming software automatically splits an image into these four areas.

The pixels for the edges are then determined and inserted a number of times, depending on the dimension of the button.

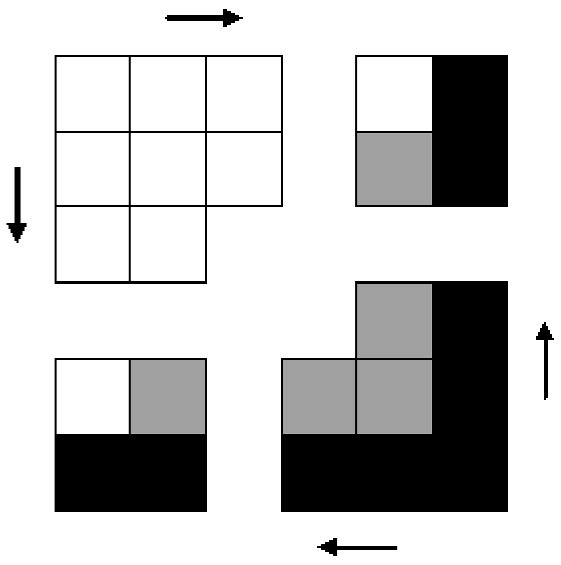


Fig. 6-45: Determining and expanding frame edges

In this context, only the top left and bottom right corners are taken into account. For the following image, each frame edge has been expanded by two pixels.

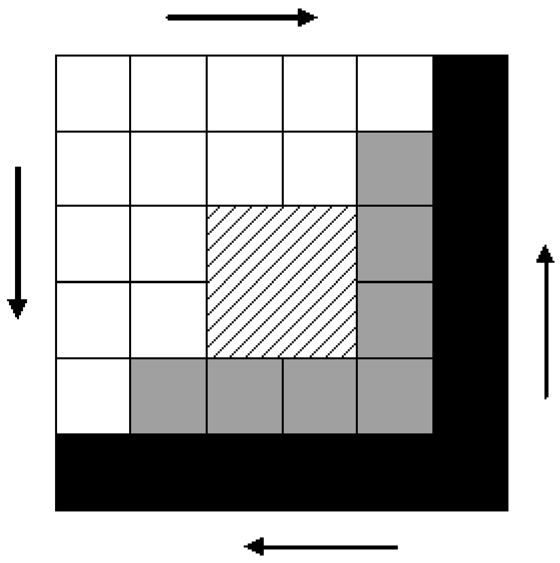


Fig. 6-46: Button: Final result

The arrows in the image illustrate the directions in which the pixels are inserted for the frame edges.

The hatched area depicts the button’s usable area. Texts, variables, and images are displayed here. You can assign a background color to this area.

You can use the formats bit map (BMP), device-independent bit map (DIB), Windows Metafile (WMF) or Enhanced Metafile (EMF) to create images.

6.3.6 Set of Curves (Graph)

You use sets of curves to graphically display the values that are recorded by one or more data loggers.

There are two wizards to help you program the representation of the sets of curves (graph).

Axis scale wizard: This wizard generates a set of curves with a single-color background.

Bitmap loader wizard: This wizard generates a set of curves with any background image of your choice.

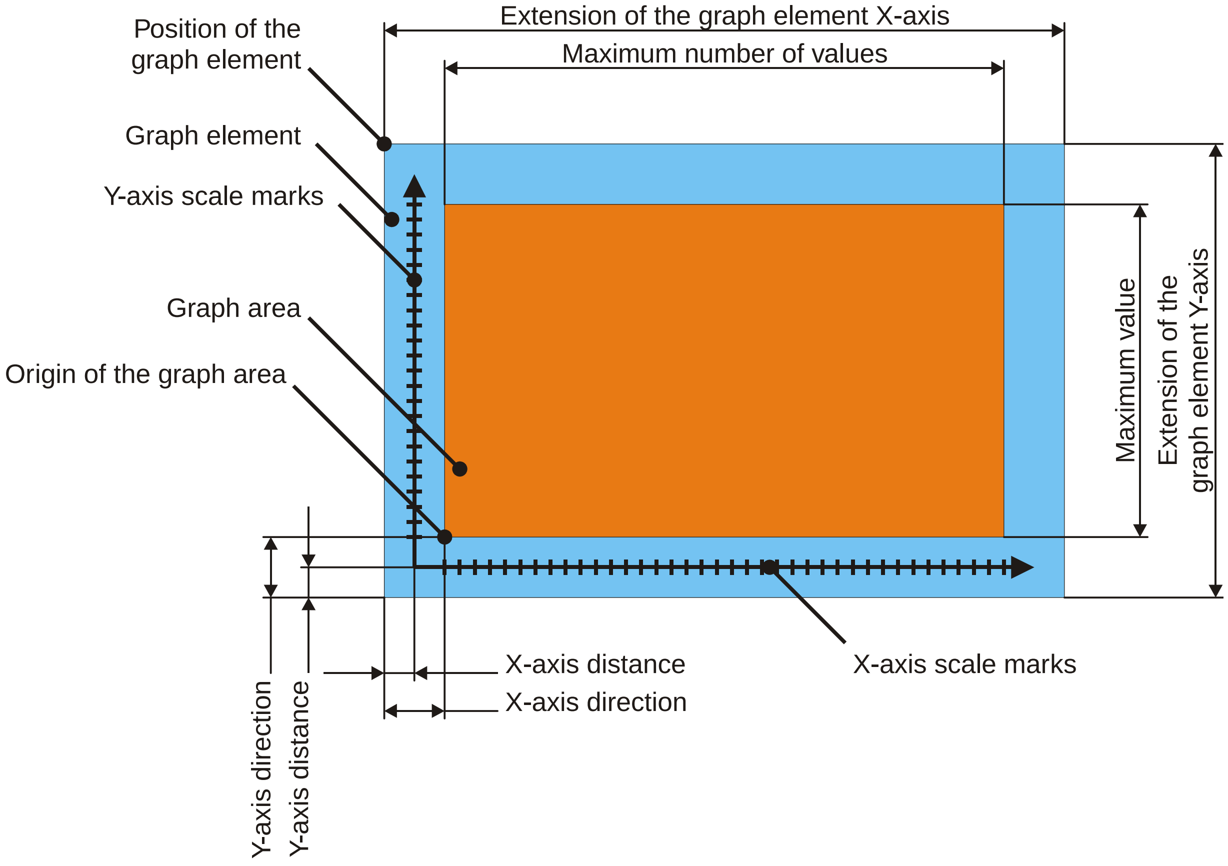


Fig. 6-47: Structure of a set of curves

6.3.6.1 Name and Recorder Type for Data Logger

You can program four independent data loggers.

The data loggers record cyclical or event-driven values between 0 and 254 from the controller. These values are displayed graphically in a set of curves.

For each data logger, you assign a name and a byte address in the controller.

You specify a recording type for each data logger:

- **Plotter continuous** (single values): The plotter moves over the output area and, in doing so, outputs the values (like an oscilloscope).
- **Plotter static** (single values): The curve is drawn continuously from the left or right edge (like an ECG plotter).
- **Flash light** (all values): The values for the curve are read as a snapshot from the controller and displayed in full (copy of all data at a point in time).

The number of data points must be specified for the horizontal direction.

You can also select the display orientation.

The number of data records per channel determines the horizontal extension of the set of curves area!

6.3.6.2 Address and Trigger for Data Recorder

The data to be recorded are kept in the controller in a variable and is polled by the operating device in accordance with a specific criterion.

Enter the name of a controller variable into the **Variable** field of the **Address** area.

Click the folder icon to selecta controller variable from the Variable dialog or to create a new controller variable in this dialog. Click the properties icon to display the properties of this variable.

- The variable address must be a byte address!
- The values of the data logger must be in the range of 0 to 254!

In the **Maximum number of channels** field, select the number of curves you want to be output.

In the **Number of data records per channel** field, specify how many data records are to be recorded in the x direction.

In the **Direction** area, select the direction in which the record process is to be carried out.

In the **Trigger** area, choose whether a value is to be recorded if an external event occurs (External Event) or at regular time intervals (Timer Event). Determine the time intervals by specifying a value in the range of full seconds, minutes or hours.

6.3.6.3 Axis Scale Wizard

You can use the axis scale wizard to define how the sets of curves are to be displayed. To start the wizard, click the **Start** button.

To do so, specify the following parameters:

- Position of the set of curves element within the screen
- Extension of the set of curves element in the X-direction
- Extension of the set of curves element in the Y-direction
- Size of the area for the set of curves
- Origin for the set of curves area
- Color of the curves
- Color of the background
- Maximum value

- Tick marks (grid) of X-axis
- Position of the Y-axis
- Number of measured values
- X-axis scale (grid)
- Position of the X-axis

6.3.6.4 Axis Scale Wizard, Colors of Graphs and Background

Select a color for each graph (curve), the position index of the plotter, the background of the curve, and the frame.

The position index of the plotter is displayed as a vertical line and shows the current position of the plotter along the X-axis.

The frame is displayed provided that the graph area (area for the set of curves) is smaller than the entire graph (set of curves) element.

6.3.6.5 Axis Scale Wizard, Geometry and Grid

To display a **Grid** (grid composed of dots), you must select the relevant check box and choose a **color** that contrasts well against the background color.

In the **Grid** (tick marks) field, specify the spacing between the grid points. In the **Extension of graph element** area, you can modify the size of the field, if necessary, in which the graph (set of curves) including the frame, scales and legend is to be displayed..

In the Frame area, enter the distance of the set of curves (graph) element from the bottom and left edge of the set of curves area.

Icon	Function
📈	Enter the distance of the graph area from the left edge of the graph element.
📈	Enter the distance of the graph area from the bottom edge of the graph element.

Fig. 6-48: Geometry and Grid parameters

6.3.6.6 Axis Scale Wizard, Legend

To display a legend for the individual sets of curves, you must select the **Display** check box.

You can enter a legend text with a maximum length of 15 characters for each set of curves.

You can use one of the following options to position the legend within the graph element.

- **Top** positions the legend across the top end of the graph (set of curves) element.
- **Bottom** positions the legend across the bottom end of the graph (set of curves) element.
- **Right** positions the legend on the right edge of the graph element.

6.3.6.7 Axis Scale Wizard, X-Axis

In the **X-axis in the Graph Element Area**, select the **Add** check box to display a scale along the x-axis.

Select the **Grid** (tick marks) check box to show tick marks within the scale. The distance between individual tick marks can be specified in the field next to the appropriate check box. Select a line thickness and line color for the representation of the scale.

Icon	Function
↦	Enter the distance of the scale from the left edge of the graph element.
↥	Enter the distance of the scale origin from the left edge of the graph element.
↔	Enter the length of the scale. Add the space between the scale and the graph area to the length to ensure that the scale extends to the right edge of the graph element.
←	Click the button to display an arrow at the left end of the scale.
—	Click the button to display the ends of the scale without an arrow.
→	Click the button to display an arrow at the right end of the scale.

Fig. 6-49: Parameters for the X-axis scale

6.3.6.8 Axis Scale Wizard, Y-Axis

In the **Y-axis in the Graph Element Area**, select the **Add** check box to display a scale along the x-axis.

Select the **Grid** (ticks) check box to show tick marks within the scale. The distance between individual tick marks can be specified in the field next to the appropriate check box. Select a line thickness and line color for the representation of the scale.

Icon	Function
	Enter the distance of the scale from the left edge of the graph element.
	Enter the distance of the scale origin from the left edge of the graph element.
	Enter the length of the scale. Add the space between the scale and the graph area to the length to ensure that the scale extends to the right edge of the graph element.
↓	Click the button to display an arrow at the bottom end of the scale.
|	Click the button to display the ends of the scale without an arrow.
↑	Click the button to display an arrow at the upper end of the scale.

Fig. 6-50: Parameters for the Y-axis scale

6.3.6.9 Bitmap Loader Wizard

Into the **Frame** area, enter the distances of the graph area from the left and bottom edge of the background image.

Icon	Function
	Enter the distance of the graph area from the left edge of the background image.
	Enter the distance of the graph area from the bottom edge of the background image.

Fig. 6-51: Parameters for the background image

In the **Import background image** area, click the folder icon to browse for a background image.

Choose a separate color for each curve, the position index of the plotter and the background image. To do this, click the arrow beside the field for the color.

Click the **Arrange Graph to the Center** button to center the graph area within the background image. The values in the "Frame" area are automatically adjusted.

6.3.7 Recipe Field

The Recipe field classifies the area in a screen used to display recipes.

To create a recipe field, carry out the following steps:

1. Select the **Recipe Field** icon in the toolbar and in the screen select the area where recipes will be displayed.

This area is displayed as a rectangle. The recipe field is marked with the letter R on the left edge of the rectangle.

2. You can use the sizing handles to change the height of the area. You can not change the width of the area.

You can change the recipe field parameters by selecting **Recipe field parameters** from the context menu.

See chapter “Working with Recipes“ on page 6-112.

6.3.7.1 Recipe Field, Parameters

Select the name of the recipe for which you want to set up a recipe field.

Just below, enter the height for the recipe field.

The **Font** area specifies the font used to display all elements of a recipe field on the operating device. To display all elements of a recipe field in another font, click the **New Font** button.

6.3.8 Table Field

The Table field classifies the area in a screen used to display values in a table.

To create a table field, carry out the following steps:

1. Select the **Table Field** icon in the toolbar and in the screen select the area where the table will be displayed.

This area is displayed as a rectangle. The table field is marked with the letter T on the left edge of the rectangle.

2. You can use the sizing handles to change the height of the area. You can not change the width of the area.

To display variable values in the table, create a variable frame in the table field. Specify the address for the variable and select the representation type.

Variables displayed in a button that has a frame can not be dragged with the mouse to the table field. Frames of buttons can not be displayed in tables.

Example:

You have 256 elements you want to be displayed in a table which has four columns. The operating device has screen with 20 lines. 16 of these lines are to be used to display elements. Therefore, create a table field with a height of 16. Enter 64 (16 lines x 4 elements) for the number of table elements.

6.3.8.1 Table Field, Parameters

Specify how many lines the table field will have and how many elements can be displayed in this table field.

The **Font** area specifies the font used to display all elements of the table on the operating device. To display all elements of the table in another font, click the **New Font** button.

6.3.9 Message Field

The message field refers to the area in a screen that is used to display messages.

To create a message field, carry out the following steps:

1. Select the **Message Field** icon in the toolbar and in the screen select the area where messages will be displayed.

This area is displayed as a rectangle. The message field is marked with the letter M on the left edge of the rectangle.

2. You can use the sizing handles to change the height of the area. You can not change the width of the area.

See chapter “Working with Messages“ on page 6-126.

6.3.9.1 Message Field, Parameters

In the **Message System** area, determine whether messages of the parallel message system or the serial message system will be displayed in the current message field.

Specify a value for the **height of the message field**. A message field can have a maximum height of 60 lines.

In the **Representation of Message** area, you can specify data that can be changed when the operating device is running.

Global settings:

If this parameter is active, default settings from the system parameters are used for the message system.

Message group:

The system outputs the group identifier before the message text.

Message number:

The system outputs the message number before the message text.

Message date:

The system outputs today’s date before the message text. In the date, the year can either be output as two digits or four digits. The value of the date is frozen with the message.

Time of message:

The system outputs the time before the message text. The value of the date is frozen with the message.

The maximum number of lines per message specifies that only the number of lines entered here will be displayed. The standard value is the maximum value of 255 lines.

In the Time Period area, you can specify the time period from which messages will be displayed.

Chronicle:

The system displays all messages.

Old list:

The system only displays acknowledged messages that do not have the status Disappeared.

In the Group Assignment area, you can select whether you would like to display message groups in the current message field, and if you would, specify the corresponding groups. Next to the group number, the system also displays the group identifier that you set up. Select the check box next to the group number, to select the required group. Any number of combinations are possible. If you do not select any check box, the system can display all groups.

The Font area specifies the character set used to display all elements of a message field on the operating device. To display all elements of a message field in another font, click the **New** button.

6.3.10 Creating System Icons / Button Display

If you are configuring table, recipe or message fields for touch-sensitive terminals, you need to generate additional buttons for navigation in the field. The buttons assume the functions of the Cursor Up, Page Up, Cursor Down and Page Down keys.

Colors:

The colors set for the released button (idle state) and pressed button (pressed state) apply to all generated buttons.

Frames:

the frames configured for the released button (idle state) and pressed button (pressed state) are used in all generated buttons.

6.3.11 Creating Navigation Buttons

If you are configuring table, recipe and message fields for touch-sensitive operating terminals, the fields require the Cursor Up, Page Up, Cursor Down and Page Down navigation buttons. For message fields, you additionally need the Enter and Clear buttons.

Under "Image/Symbol for Keys", select the relevant images or symbols acting as button contents for the relevant buttons.

6.3.12 Output Variables

Output variables are numeric or alphanumeric memory content from the connected controller. The variable values are requested from the controller if required, and displayed at the program location using the corresponding representation type.

6.3.12.1 Once-Off and Cyclic Output Variables

Pure output variables are transferred once from the controller when the screen is being called-up, and are displayed in the screen. Outputting the variable only once helps improve communication performance, and can be used for all variables, such as setpoint values, constants, and parameters that rarely or never change. All output variables can be displayed as scaled or formatted.

Cyclical output variables are used to display actual values and values that continuously change while a screen is being output.

You specify the cycle time with the polling time. This means that you know at this stage how often the display of the actual values will be refreshed.

The scaling and formatting of cyclical output variables, in particular of decimal numbers as floating point numbers, requires a corresponding computing time, and as a result the data is not output in real time.

The more cyclical data is transferred, the longer the reaction time to new values from the controller.

For these applications, select cycle times > 500 ms.

To improve the performance of transfer to the controller, use data types identically and ensure that the address ranges of a screen are as continuous as possible.

6.3.12.2 Formatted Output

You can format a numeric variable value to suit an output area.

Formatting consists of:

- Field length
- Fractional digits
- Positive values only
- Display leading zeros

The field length determines the entire length of the output value, including signs, decimal points, and fractional digits.

The number of fractional digits gives the operator the impression that a value has been divided, however, in reality no value has been divided. However, the variable value must exist in the controller in a correspondingly high resolution.

Example:

In the controller, the value of a length is stored as a word.The range of values is between 0 and 65535. The following settings are made for display:

- Decimal number
- Output
- Only positive
- Field length = 6 (5 + decimal point)
- Fractional digits = 2 (absolute)

The display area is between 0,00 and 655,35. If the check box Only Positive is not selected, the display area changes. The value is displayed with a sign. You must specify an additional position in the field length for the sign. The following data is required:

- Decimal number
- Output
- Field length = 7 (5 + decimal point + sign)
- Fractional digits = 2 (absolute)

The display range is then between –327.68 and +327.67.

6.3.13 Input Variables

When displaying input variables for the first time in the operating device, the system uses the same approach as for one-off output variables (output variables that are output only once). This also applies to scaling, which works from the controller's viewpoint.

Input variables are processed by editors in the operating device.

6.3.13.1 Plausibility Check

The system carries out a plausibility check for all input variables. During this check, it compares the value entered with the range limits stored in the variable list.

If the limits are not adhered to, the system issues one of the following terminal messages:

- 'Value too large' or
- 'Value too small'

The incorrect value is not written to the controller. If an error occurs, the previously valid value is retained.

To prevent the above-mentioned terminal messages from appearing, you must delete them in the programming software. When you do this, the following applies:

- If the value is exceeded, the value of the upper limit is entered
- If the value is not reached, the value of the lower limit is entered

6.3.14 Dynamic and Static Attributes

For the screen elements Static Text, Text Field, and Variable, you can assign 255 ranges of values for dynamic attributes.

Dynamic attributes change the display of a text or variable value in the operating device, based on a variable or control variable value.

The system displays the values for the upper and lower limit in a list box. In the same line, it displays the attributes for values that are within the limits. You can not enter overlapping value ranges!

1. Enter the values for the upper and lower limit under the list box.
2. Select the corresponding attributes in the relevant check boxes.
3. Assign the attributes to the range of values. The range of values and its attributes are simultaneously entered in the list box.

The dynamic attributes are either derived directly from the value of a variable or from the value of a corresponding control variable.

For variables in input and output screens, the entry of a control variable is optional.

In general, no control variables are permitted in recipes. Here, the dynamic attributes can only be derived from the value of the variables.

For texts in input/output screens, a control variable is always required to control the attributes.

You can not assign dynamic attributes for texts in recipes.

To assign dynamic attributes, carry out the following steps:

1. Click one of the lines in the list box.
2. Enter limit values into the appropriate fields.
3. Select the corresponding attributes.
4. Enter the name of the control variable, if needed.
5. Click the Assign button.

You can display up to 25 objects with dynamic attributes in a screen.

6.3.14.1 Global

Variables with the attribute Global:

- Appear in all languages for a project
- in all screens with the same name
- and in the same position

If you change the parameters of these variables, the changes apply to all screens with the same name and to all languages for the project.

6.3.14.2 Underline

You can assign the (dynamic) attribute **Underline** to variables and static texts.

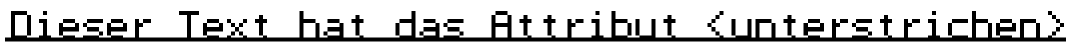

Dieser Text hat das Attribut <unterstrichen>

Fig. 6-52: Text with the Underline attribute

6.3.14.3 Inverse

You can assign the (dynamic) attribute **I**nverse to variables and static texts. This format is particularly suitable if you want to emphasize the variable that is currently selected.

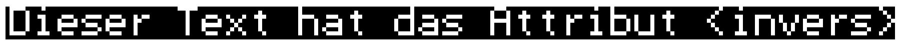

Dieser Text hat das Attribut <invers>

Fig. 6-53: Text with the Inverse attribute

6.3.14.4 Flashing

You can assign the (dynamic) attribute **Flashing** to variables and static texts. Note that an element that is assigned this attribute is displayed in the strikethrough format and not as flashing text.

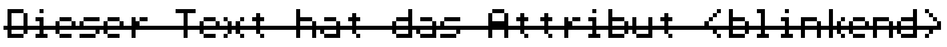

~~Dieser Text hat das Attribut <blinkend>~~

Fig. 6-54: Text with the Flashing attribute

6.3.14.5 Invisible

You can assign the dynamic attribute **Invisible** to variables and static text, to ensure that they do not appear below or above specific controller values.

The attribute **Invisible** is only evaluated together with a control variable. In all other cases, the attribute **Invisible** results in an error message during compilation, and no terminal file is generated.

Static texts and one-off variables (variables that are output only once) with the attribute **Invisible** are not output.

Cyclical variables with the attribute **Invisible** are overwritten with blanks. This erases any existing obsolete value on the screen.

If background images and cyclical variables are being used simultaneously, the background image is not updated!

Regarding input variables, note that the operator can not enter the range of values for the attribute **Invisible**. If an input value is in the area of the attribute **Invisible**, the edit process is not started. The operator then has no possibility to change the value again.

6.3.14.6 Non-Editable

You can assign the dynamic attribute **Non-Editable** to variables, to ensure that they can not be changed below or above specific controller values.

You can only use the **Non-Editable** attribute in conjunction with a control variable.

6.3.14.7 Foreground

You can assign the (dynamic) attribute 'Foreground' to variables and static text, to ensure that they are (when values are below or above specific controller values) are displayed with a specific color.

6.3.14.8 Background

You can assign the (dynamic) attribute 'Foreground' to variables and static text, to ensure that they are (when values are below or above specific controller values) are displayed in front of a specific background.

6.3.14.9 Attribute Priorities

1. First, the attributes of the variable or text set in the normal dialog box for the screen element are used.
2. If a control variable exists, its value and the range of values definition are used to define the dynamic attribute.
3. If no control variable exists, the value of the PLC variables and the range of values definition are used to determine the dynamic attribute. (Not in the case of static texts).
4. If value-specific attributes were defined in the text list for selection texts, these attributes are used.

6.3.14.10 Variable Selection

Double-click a variable folder to open it.

The variables are then positioned one below the other.

You can select a variable by double-clicking it. The dialog is then closed and you are returned to the previous window.

6.3.14.11 Font

The Font attribute determines the font in which the characters are displayed in a screen.

Note:

- Only some operating devices can display fonts up to any size.
- The names of the fonts can not be changed.
- You can select a separate font for each static text in a screen.
- The system displays all messages using the same font.
- The system displays all elements in a recipe using the same font.

6.3.15 Aligning Selected Elements to the Grid

Elements in a screen are normally aligned to the grid whose grid spacing is defined by the width of the characters and numbers. When a pixel-oriented graphic display is used, you can also move the elements in a screen pixel-by-pixel.

The **Align selected elements** function can be used to undo the pixel-oriented positioning of elements!

Use the following procedure:

1. Select the elements in the screen you want to realign to the grid.
2. Select **Align selected elements** from the context menu.

You can now move the selected elements according to the grid spacing defined by the characters.

6.3.16 Aligning Selected Elements

You can align the elements in a screen either in relation to each other or at the edge of the screen. At the same time, they are aligned in horizontal and/or horizontal direction. Only the selected elements are taken into account during this process.

Elements are aligned to:

Select whether you want the elements to be aligned to the border or to the selection boundary.

Selection boundaries are imaginary lines of outer points of the selected outermost elements in vertical and horizontal direction.

The following figure shows a square, circle and triangle as selected elements.

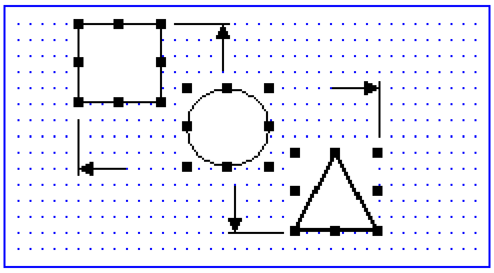


Fig. 6-55: Selection boundary

In horizontal direction, the left edge of the square is the selection boundary to the left; to the right it is the right corner of the triangle. In vertical direction, the top edge of the square is the selection boundary to the top; to the bottom it is the bottom edge of the triangle.

Horizontal:

Select which of the following options you want to apply to the horizontal direction of the selected elements:

- Retain current position
- Left justification
- Centering or
- Right justification

Vertical:

Select which of the following options you want to apply to the vertical direction of the selected elements:

- Retain current position
- Top justification
- Centering or
- Bottom justification

6.4 Working with Libraries

Libraries correspond to data bases where you can put-in screen objects which fit to a certain operating device. You put the objects into the library or from the library into the screen using drag and drop.

The context menu of libraries offers the following menu items:

- New library
- Open library
- Save library
- Close library

New library:

Generates a new library.

Open library:

Opens an already existing library.

Save library:

Saves the topical library.

Close library:

Closes the topical library.

See chapter “Create Project Folder, Template or Library (Templates)“ on page 5-4.
See chapter “Create Project Folder, Template or Library (Place to Store)“ on page 5-4.
See chapter “Create Project Folder, Template or Library (Terminal Type)“ on page 5-5.
See chapter “Create Project Folder or Template (Protocol Type)“ on page 5-6.

6.5 Working with Edit Screens

When you are configuring touch screen terminals, a specific editor is available for every input variable. You can launch the editor by pressing the input variable. Five different types of edit screens are available (depending on the variable type):

- Decimal (to edit decimal number variables)
- Hexadecimal (to edit hexadecimal number variables)
- Binary (to edit binary variables)
- Increment
- Alphanumerical (to edit alphanumerical variables)

For each variable, the appropriate editor is automatically used.

When a free display area in the edit screen is pressed, the editor is closed and the previous screen is displayed again. However, the data is not adopted in this case.

You can adopt the data by clicking the output area of the variable (editor display). The editor is then also closed.

To simplify the configuration process, you can use the editors provided as a template.

In your edit screen, you should include one button for each key to simulate and one button to display the variable value.

Note:

- To save memory space, only around 70% of the screen surface should be used up by edit screens. You can position the edit screen by dragging the red boundary lines to the corresponding positions. The edit screen will always appear at this position.
- Make sure that buttons located in edit screens do not contain input variables since this would result in an editor again.

The lists below contain the key functions you should insert into the corresponding editor screen at minimum.

Decimal:

- Cursor Left
- Cursor Right
- + (Plus)
- - (Minus)
- . (Decimal point)
- Home
- Help
- Clear
- 0 to 9

Hexadecimal:

- Cursor Left
- Cursor Right
- + (Plus)
- - (Minus)
- Home
- Help
- Clear
- 0 to 9
- A to F

Binary:

- Cursor Left
- Cursor Right
- + (Plus)
- - (Minus)
- Home
- Help
- Clear
- 0 and 1

Increment:

- Cursor Left
- Cursor Right
- +
- -
- Home
- Help
- Clear

Alphanumeric:

- Cursor Left
- Cursor Right
- + (Plus)

- - (Minus)
- . (Decimal point)
- Home
- Help
- Shift
- Clear
- 0 to 9
- a to z

If you configure a Shift button, the following numbers are superimposed:

- 0 with =
- 1 with !
- 2 with "
- 3 with ?
- 4 with $
- 5 with :
- 6 with &
- 7 with \8 with (
- 9 with)

6.6 Working with Scripts

Scripts are structured programs that are processed procedurally by an interpreter at runtime. You can use scripts to implement your own ideas, thus allowing you to derive greater benefit from your applications.

Scripts are started by events or cyclically. These types of events are

- Opening a screen
- Exiting from an editor screen

Scripts can not be used to start other scripts.

Since scripts are procedurally processed, the processing of an individual script is completed once the last script line has been processed. However, you can use control structures to create loops, whose conditions may not ever be fulfilled, within a script. Although the operating device is not blocked as a result, it may display incorrect variable values.

You can perform the following actions in scripts:

- Influence the values of variables (reading and writing)
- Use control structures to read conditions.

Within scripts, you can use variables, which are classified as follows.

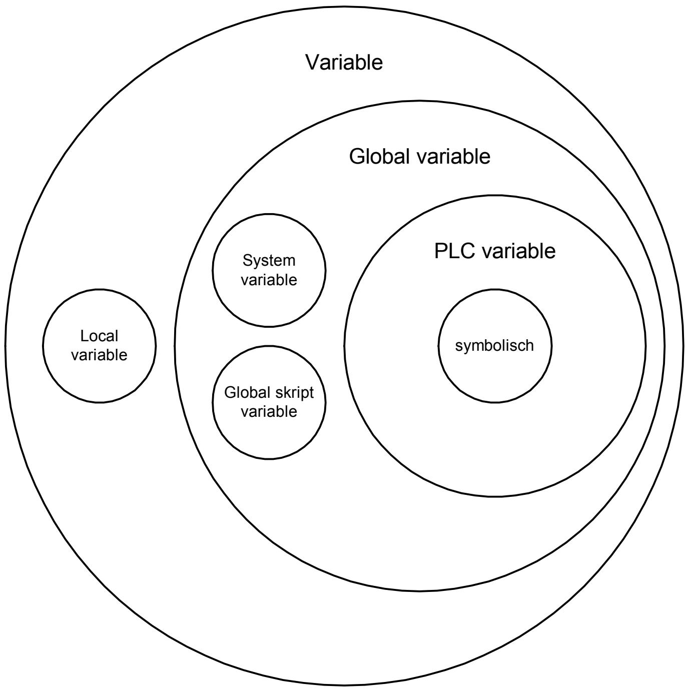


Fig. 6-56: Classification of variables

Depending on the variable type, you must declare the variable at another position. Locations for the declaration include the script itself or the script variable list. Only system variables must not be declared, as their data types are known to the system.

Type	**Declaration location**
Local script variable	In the script
Global script variable (retentive)	Variable list for script variables
System Variable	None (implicit)
Controller variable	Variable list for controller variables

Fig. 6-57: Variables for script processing

To create a script, start the script editor in the programming software. This editor provides support for the structured writing of scripts by

- Automatically suggesting completions
- Automatically setting tabs
- Displaying script elements using different colors and
- Marking errors.

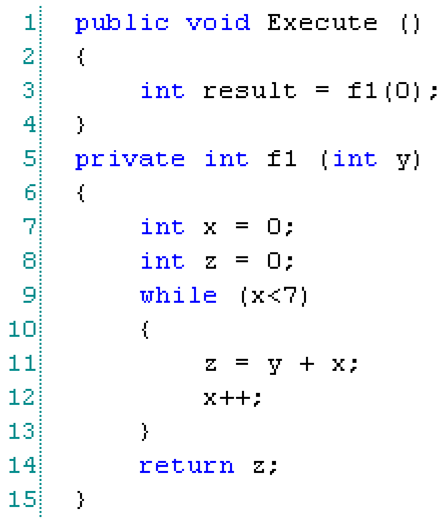
If you notice an error in the script, you can receive additional information by using the mouse cursor to navigate to the marked area.

The content of a script is restricted to elements that are permissible in the specified namespaces. This means that you only receive scripts that are executable on the relevant hardware platform.

The following chapters describe the methods, which are divided into classes, that you can use within the namespaces. Depending on the class, these are local methods (project) or global methods (firmware). Global methods are used to access system resources (system variables).

Scripts have a specific structure. The use of tab stops provides you with a better overview of the content.

Structure of a script:



```
 1  public void Execute ()
 2  {
 3      int result = f1(0);
 4  }
 5  private int f1 (int y)
 6  {
 7      int x = 0;
 8      int z = 0;
 9      while (x<7)
10      {
11          z = y + x;
12          x++;
13      }
14      return z;
15  }
```

Fig. 6-58: Sample script

1 Declaration of function with parameter transfer ()
2 Opening parenthesis for the following script content
3 Assignment of the results of f1 into the variable "result". The f1 method is transferred to the parameter (0).
4 Closing parenthesis for the script content
5 Declaration of the f1 method and transfer of the parameter (0) into the variable (int y). The "int“ declaration causes the method to return an integer value.
6 Opening parenthesis for the contents of the f1 method
7 Declaration of the x variable with value assignment
8 Declaration of the z variable with value assignment
9 Start the control structure with the query parameter (x<7)
10 Opening parenthesis for the contents of the control structure
11 Assignment of the result of adding the y and x variables into the z variable
12 Operation (increment operation) executed at the x variable
13 Closing parenthesis for the contents of the control structure
14 Return of the value of z to the calling method
15 Closing parenthesis for the contents of the f1 method

Declaration:

Each method (script) must begin with the declaration. The declaration specifies the range of validity in which the method is used, whether parameters are transferred and the name of the method.

Syntax	Function
public	Methods that are started by an event
private	Methods that are started by other methods

Fig. 6-59: Ranges of validity

Parameters are either transferred to the method or returned by the method. The return of a parameter is initiated with the "return“ keyword.

Syntax	Function
public void	This method does not return any parameter
public bool public int public uint public double public string	A parameter with the specified data type of this method is returned
private void f1 (bool y) private void f1 (int y) private void f1 (uint y) private void f1 (double y) private void f1 (string y)	A parameter of the specified data type is transferred to this method and written to the y variable

Fig. 6-60: Parameter transfers

You can choose any name for a method but you must pay attention to upper and lower-case spelling. However, a method that is called by an event must have the name "Execute"! (see line 1 of the sample script)

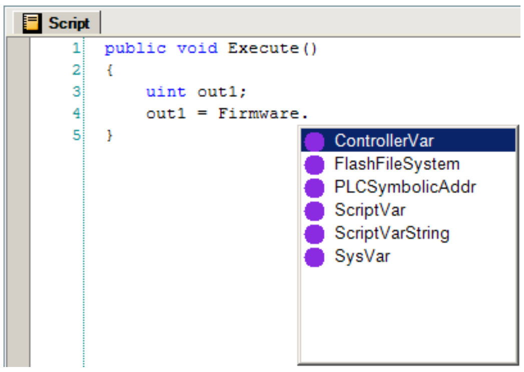
6.6.1 Script Editor

Create your scripts either with the internal script editor of the programming system or any external editor.

To overtake scripts edited by means of an external editor import its contents via the clipboard into the internal script editor.

The internal script editor has the advantage to offer automatically completion of certain text. This will reduce your work and also produces less typing errors. The following automatically completions are offered:

Classes names after namespaces:

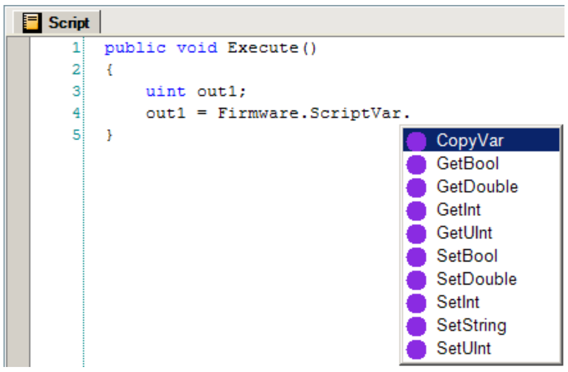


```
public void Execute()
{
    uint out1;
    out1 = Firmware.
}
```

ControllerVar
FlashFileSystem
PLCSymbolicAddr
ScriptVar
ScriptVarString
SysVar

Fig. 6-61: Automatically offer for classes names

Methodes names after classes names:

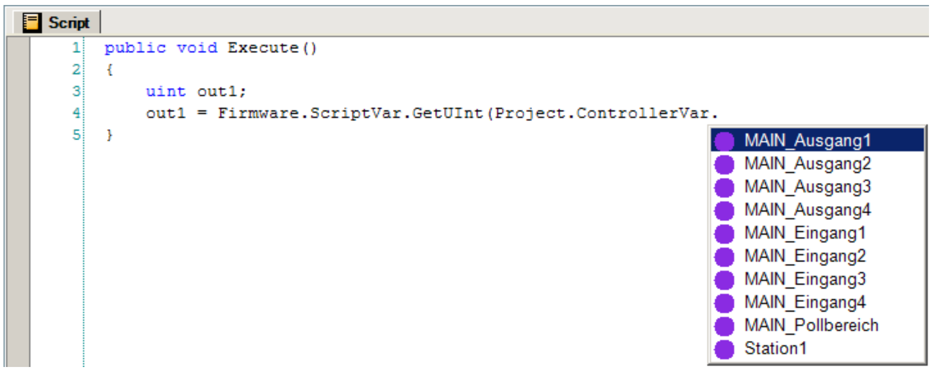


```
public void Execute()
{
    uint out1;
    out1 = Firmware.ScriptVar.
}
```

CopyVar
GetBool
GetDouble
GetInt
GetUInt
SetBool
SetDouble
SetInt
SetString
SetUInt

Fig. 6-62: Automatically offer for methods names

Variables names after methods names:



```
public void Execute()
{
    uint out1;
    out1 = Firmware.ScriptVar.GetUInt(Project.ControllerVar.
}
```

MAIN_Ausgang1
MAIN_Ausgang2
MAIN_Ausgang3
MAIN_Ausgang4
MAIN_Eingang1
MAIN_Eingang2
MAIN_Eingang3
MAIN_Eingang4
MAIN_Pollbereich
Station1

Fig. 6-63: Automatically offer for variables names

6.6.2 Namespaces

You can use the following namespaces within the script:

- Firmware - for script elements that access elements, directly or indirectly, that are provided by the firmware.
- Project - for all other script elements

6.6.3 Classes

You can use the following classes within the namespaces:

- ControllerVar - provides access to native controller addresses
- FileAccess - proviedes access to files
- FlashFileSystem - provides access to the flash file system
- PLCSymbolicAddr - provides access to symbolic controller addresses (only for BRC-Symbolic)
- ScriptVar - provides access to script variables, local and global
- ScriptVarString - provides access to string-type script variables
- SysVar (firmware) - provides access to system variables

6.6.4 Methods

Methods are like program functions. To receive the value of a system variable or a string in a script, you must call a method that returns the requested value. Additional information, for example, whether the method was successfully applied, can also be returned along with the requested value. In this case, the return value could either be true(1) or false (0).

Depending on the class, you can use different methods.

6.6.4.1 Methods for ControllerVar Class

With this class, you access the controller variables from the script. The controller variables must already be present in the variable list for controller variables.

The variable names must be conform to the following restictions.

- Only the characters a to z, A to Z, 0 to 9 and _ are permitted.
- A variable name must not begin with a number (e.g. 2Stations).
- Each variable name must be unambiguous.

If a variable name fails to meet the restrictions, the programming software automatically changes it.

Examples:

Original Name	Converted Name in the Script	Comment
Machine	Project.ControllerVar.Machine	Unchanged
!Caution!	Project.ControllerVar._Caution_	Symbol is changed
Overvoltage	Project.ControllerVar.Overvolt-age	Unchanged
Overvoltage	Project.Controller-Var.Overvoltage_2	Already defined, made unambigu-ous by _2
dwordarray[20]	Project.Controller-Var.dwordarray_20_	Symbols are changed
2Stations	Project.ControllerVar._Stations	Number at the be-ginning is changed

Fig. 6-64: Conversion of variables names

In the event of an error an error code is written to the return variable.

- the two high bytes contain the code of the COMMUNICATION ER-ROR,
- the two low bytes contain the subcode of the COMMUNICATION ER-ROR.

The code/subcode depends on the communication protocol that is cur-rently used.

GetBool(return variable, controller variable)

Reads the value of a controller variable from the data type bool.

Method	Return type
GetBool	bool
Return value = Value of the variable	

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint

GetInt(return variable, controller variable)

Reads the value of a controller variable as a signed integer with 32 bits.

Method	**Return type**
GetInt	int
Return value = Value of the variable	

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint

GetUInt(return variable, controller variable)

Reads the value of a controller variable as an unsigned integer with 32 bits.

Method	**Return type**
GetUInt	uint
Return value = Value of the variable	

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint

GetDouble(return variable, controller variable)

Reads the value of a controller variable as an IEEE floating point number with 8 bytes.

Method	**Return type**
GetDouble	double
Return value = Value of the variable	

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint

GetString(return variable, destination script variable, controller variable)

Copies a string from a controller variable into a destination script variable.

Method	**Return type**
GetString	uint
Return value = Length of the string (number of characters)	

Parameter	**Type**
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
Controller variable	uint

Depending on the controller and controller communication setting, the string is coded as a 8 bit ASCII or 16 bit UNICODE string!

SetBool(return variable, controller variable, value)

Writes a boolean value into the controller variable.

Method	**Return type**
SetBool	void

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint
Value	bool

SetInt(return variable, controller variable, value)

Writes a signed integer with 32 bits into the controller variable.

Method	**Return type**
SetInt	void

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint
Value	int

SetUInt(return variable, controller variable, value)

Writes an unsigned integer with 32 bits into the controller variable.

Method	**Return type**
SetInt	void

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint
Value	uint

SetDouble(return variable, controller variable, value)

Writes an IEEE floating point number with 8 bytes into the controller variable.

Method	**Return type**
SetInt	void

Parameter	**Type**
Return variable (global script variable)	uint
Controller variable	uint
Value	double

SetString(return variable, target controller variable, source script variable, counter)

Copies the string from the script variable to the controller variable. The counter parameter contains the number of characters to be written.

The return variable contains the length of the string that is read (number of characters).

Method	**Return type**
SetString	uint

Parameter	**Type**
Return variable (global script variable)	uint
Target controller variable	uint
Source script variable (global script variable)	uint
Counter	uint

☞ Depending on the controller and controller communication setting, the string is coded as a 8 bit ASCII or 16 bit UNICODE string!

6.6.4.2 Methods for FileAccess Class

With this class you access files from the script. With this class you can:

- open files.
- close files.
- write data of different types into files.
- read data of different types from files.

In the event of an error an error code that is analogue to the Win API function GetLastError() is written to the return variable.

FileOpen(return variable, script variable, mode)

Opens a file with the name stored in the script variable. With mode you declare the type of access.

The return variable contains the file handle for the opened file.

Method	**Return type**
FileOpen	uint

Parameter	**Type**
Return variable (global script variable)	uint
Script variable (global script variable)	uint

Mode
r = read
w = write
a = append
t = text
b = binary
c = enable commit flag
n = reset commit flag
	string

FileClose(return variable, file handle)

Closes the file with the file handle of the **FileOpen** method.

Method	**Return type**
FileClose	uint

Parameter	**Type**
Return variable (global script variable)	uint
File handle	uint

FileReadBool(return variable, target script variable, file handle)

Reads a boolean value from the file with the file handle of the **FileOpen** method. The target script variable contains the data.

Method	Return type
FileReadBool	uint
Return value >0 = Number of read data in byte Return value 0 = Error	

Parameter	Typ
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

FileReadInt(return variable, target script variable, file handle)

Reads an integer value from the file with the file handle of the FileOpen method. The target script variable contains the data.

Method	Return type
FileReadInt	uint
Return value >0 = Number of read data in byte Return value 0 = Error	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

FileReadUint(return variable, target script variable, file handle)

Reads an unsigned integer from the file with the file handle of the **FileOpen** method. The target script variable contains the data.

Method	**Return type**
FileReadUint	uint
Return value >0 = Number of read data in byte Return value 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

FileReadDouble(return variable, target script variable, file handle)

Reads an value of the type double from the file with the file handle of the **FileOpen** method.

Method	**Return type**
FileReadDouble	uint
Return value >0 = Number of read data in byte Return value 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

FileReadString(return variable, target script variable, file handle, counter)

Reads a string from the file with the file handle of the **FileOpen** method. The counter parameter contains the number of characters (16 bit) to read without termination. Enter a "0" for the counter parameter to read a string with "\0" termination.

Method	**Return type**
FileReadString	uint
Return value >0 = Number of read characters (without termination) Return value 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint
Counter	uint

FileReadOneByte(return variable, target script variable, file handle)

Reads a byte from the file with the file handle of the **FileOpen** method.

Method	**Return type**
FileReadOneByte	uint
Return value >0 = Number of read data in byte Return value 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

FileWriteBool(return variable, source script variable, file handle)

Writes a booelan value into the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written. The return variable contains the number of written data.

Method	**Return type**
FileWriteBool	uint
Return value >0 = Number of read data in byte Return value 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

FileWriteInt(return variable, source script variable, file handle)

Writes an integer value into the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	**Return type**
FileWriteInt	uint
Returnvalue = Number of written data in byte	

Parameter	**Type**
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

FileWriteUint(return variable, source script variable, file handle)

Writes an unsigned integer value into the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	Return type
FileWriteUInt	uint
Returnvalue = Number of written data in byte	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

FileWriteDouble(return variable, source script variable, file handle)

Writes a value of the type 0x83 FLOAT64 into the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	Return type
FileWriteDouble	uint
Returnvalue = Number of written data in byte	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

FileWriteString(return variable, source script variable, file handle, counter)

Writes a string into the file with the file handle of the **FileOpen** method. The counter parameter contains the number of characters (16 bit) to be written either with or without termination. Enter a "0" for the counter parameter to write a string with "\0" termination to the file.

Method	**Return type**
FileWriteString	uint
Returnvalue = Number of written data in byte	

Parameter	**Type**
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint
Counter	uint

FileWriteOneByte(return variable, source script variable, file handle)

Writes a byte into the file with the file handle of the **FileOpen** method.

Method	**Return type**
FileWriteOneByte	uint
Returnvalue = Number of written data in byte	

Parameter	**Type**
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

6.6.4.3 Methods for FlashFileSystem Class

With this class, you access the FlashFile system from the script. This allows you to store Drive parameters from the controller to the Flash-File system of the small operator terminal and to write them back to the controller.

To store Drive parameters from the controller onto the terminal, you need to call up the following functions in sequence:

1. ResetFFS
2. InitSaveDriveParam
3. SaveDriveParam
4. CloseBackupDriveParam

To transfer Drive parameters from the terminal onto the controller, you need to call up these functions in sequence:

1. InitRestoreDriveParam
2. ReadBytesToRestore
3. RestoreDriveParam

In the event of an error an error code is written to the return variable.

- the two high bytes contain the error code,
- the two low bytes contain the error subcode.

The error codes / subcodes are decribed for each method.

ResetFFS(return variable)

Locks the interface for the application and then erases the memory of the FlashFile system.

Method	**Return type**
ResetFFS	uint
Returnvalue >0 = OK Returnvalue 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint

Error code	**Error Subcode**	**Description**
255	3	Wrong Flash
	6	Error Erasing Flash

Fig. 6-65: Return variable for method ResetFFS

InitSaveDriveParam(return variable, station address)

Initializes the backup of the Drive parameters which are stored in controller at the station address.

Method	**Return type**
InitSaveDriveParam	uint
Returnvalue >0 = Number of bytes to save Returnvalue 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Station address	uint

Error code	**Error Subcode**	**Description**
255	1	Wrong Length
	2	File Write Error
	3	File Open Error
	4	File Close Error
0	0	No Data to Save

Fig. 6-66: Return variable for method InitSaceDriveParam

SaveDriveParam(return variable, station address, start element, number of elements)

Reads a block of elements (a maximum of 1400) - starting from the number of the start element - which are located at the station address of the controller.

If more than 1400 elements are to be read, it might be necessary to start this method several times in order to obtain the complete set of parameters.

Method	**Return type**
SaveDriveParam	uint
Returnvalue >0 = Operation OK Returnvalue 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint

Parameter	Type
Station address	uint
Start element	uint
Number of elements	uint

Error code	Error Subcode	Description
255	1	Wrong Length
	2	File Write Error
	3	File Open Error
	4	File Close Error
	5	Insufficient Memory

Fig. 6-67: Return variable for method SaveDriveParam

CloseBackupDriveParam(return variable, station address)

Terminates the transfer of Drive parameters stored at the station address of the controller and opens the interface for the application.

Method	Return type
CloseBackupDriveParam	uint
Returnvalue 1 = Operation OK Returnvalue 0 = Error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint

Error code	Error Subcode	Description
255	3	File Open Error
	4	File Close Error

Fig. 6-68: Return variable for method CloseBackupDriveParam

InitRestoreDriveParam(return variable, station address)

Initializes the transfer of Drive parameters to the station address in the controller.

Method	Return type
InitRestoreDriveParam	uint
Returnvalue 1 = Operation OK Returnvalue 0 = Error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint

Error code	Error Subcode	Description
255	3	Wrong Flash

Fig. 6-69: Return variable for method InitRestoreDriveParam

ReadBytesToRestore(return variable)

Reads the number of bytes to be written to the controller.

Method	Return type
ReadBytesToRestore	uint
Returnvalue >0 = Number of data in byte Returnvalue 0 = Error	

Parameter	Type
Return variable (global script variable)	uint

Error code	Error Subcode	Description
255	1	Wrong Length
	2	File Write Error
	3	File Open Error
	4	File Close Error
0	0	No Data to Save

Fig. 6-70: Return variable for method ReadBytesToRestore

RestoreDriveParam(return variable, station address, start element, number of elements)

Reads a block of elements (a maximum of 1400) - starting from the number of the start element - which are located at the station address of the controller.

If more than 1400 elements are to be written, it might be necessary to start this method several times in order to transfer the complete set of parameters.

Method	**Return type**
RestoreDriveParam	uint
Returnvalue >0 = Number of data in byte Returnvalue 0 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Station address	uint
Start element	uint
Number of elements	uint

Error code	**Error Subcode**	**Description**
255	1	Wrong Length
	2	File Write Error
	3	File Open Error
	4	File Close Error
	5	File Seek Error
	6	File Length Error

Fig. 6-71: Return variable for method RestoreDriveParam

6.6.4.4 Methods for PLCSymbolicAddr Class

With this class, you access - from the script - individual controller variables using symbolic addresses.

Supported are single accesses to a controller variable and single accesses to a field of a controller variable.

You store the access address in a script variable (address script variable; e.g. 10,5,S-0-0001.5). You can then use this script variable to access the controller variable. If the action is successful, the return value contains a 0 (zero).In the event of an error, the error number of the communication server is transferred as a return value.

Using the Element Number parameter, you specify whether you are accessing a single variable (0) or a field (0x8000 | element number).

GetListLength(return variable, address script variable, station address)

Determines the length of a field variable.

For this purpose, a read access is performed to element 0 of the field with the type 0x09 C-STRING.

Method	**Return type**
GetListLength	uint

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint

GetBool(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x01 bool with a length of 1 byte.

Method	**Return type**
GetBool	bool

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

GetInt(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x04 INT32 with a length of 4 bytes.

Method	**Return type**
GetInt	int

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

GetUInt(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x07 UINT32 with a length of 4 bytes.

Method	**Return type**
GetUInt	uint

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

GetDouble(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x83 FLOAT64 with a length of 8 bytes.

Method	**Return type**
GetDouble	double

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

GetString(return variable, target script variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x09 C-STRING.

Method	**Return type**
GetString	uint
Returnvalue = Length of the string (number of characters)	

Parameter	**Type**
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

SetBool(return variable, address script variable, value, station address, element number)

Writes a value into a variable of the type 0x01 bool with a length of 1 byte.

Method	**Return type**
SetBool	void
Returnvalue = TRUE	

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	bool
Station address	uint
Element number	uint

SetInt(return variable, address script variable, value, station address, element number)

Writes a value into a variable of the type 0x04 INT32 with a length of 4 bytes.

Method	**Return type**
SetInt	bool
Returnvalue = TRUE	

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	int
Station address	uint
Element number	uint

SetUInt(return variable, address script variable, value, station address, element number)

Writes a value into a variable of the type 0x07 UINT32 with a length of 4 bytes.

Method	**Return type**
SetUInt	bool
Returnvalue = TRUE	

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	uint
Station address	uint
Element number	uint

SetDouble(return variable, address script variable, value, station address, element number)

Writes a value into a variable of the type 0x83 FLOAT64 with a length of 8 bytes.

Method	**Return type**
SetDouble	bool
Returnvalue = TRUE	

Parameter	**Type**
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	double
Station address	uint
Element number	uint

SetString(return variable, source script variable, address script variable, value, station address, element number)

Writes a value into a variable of the type 0x09 C-STRING.

Method	**Return type**
SetString	uint
Returnvalue = Length of the written string (number of characters)	

Parameter	**Type**
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

6.6.4.5 Methods for ScriptVar Class

GetBool(script variable)

GetInt(script variable)

GetUInt(script variable)

GetDouble(script variable)

Reads the value of a script variable from the respective data type.

Method	**Return type**
GetBool	bool
GetInt	int
GetUInt	uint
GetDouble	double

Parameter	**Type**
Script variable (global script variable)	uint

SetBool(script variable, value)

Method	Return type
SetBool	bool

Parameter	Type
Script variable (global script variable)	uint
Value	bool

SetInt(script variable, value)

Method	Return type
SetInt	int

Parameter	Type
Script variable (global script variable)	uint
Value	int

SetUInt(script variable, value)

Method	Return type
SetUInt	uint

Parameter	Type
Script variable (global script variable)	uint
Value	uint

SetDouble(script variable, value)

Writes a value to a script variable of the respective data type.

Method	Return type
SetDouble	double

Parameter	Type
Script variable (global script variable)	uint
Value	double

SetString(script variable, string)

Writes a string to a script variable.

Method	Return type
SetString	uint

Parameter	Type
Script variable (global script variable)	string
String	"String"

CopyVar(target script variable, source script variable)

Copies the value of a script variable to another script variable.

Both variables must be of the same data type.

Method	Return type
CopyVar	uint

Parameter	Type
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint

DoubleToString(return variable, double value, fractional digits, target script variable)

Converts the double value to a string. The string is written to the target script variable.

Method	Return type
DoubleToString	void
Returnvalue 0 = OK Returnvalue -1 = Error	

Parameter	Type
Return variable (global script variable)	uint
Double value	double
Fractional digits	uint
Script variable (target) (global script variable)	uint

Error code	Description
-1	Double value is larger than destination memory

Fig. 6-72: Return variable for method DoubleToString

StringToDouble(return variable, source script variable)

Converts the string contained in a script variable into a double value. The return variable contains the double value.

Method	Return type
StringToDouble	double
Returnvalue 0 = OK Returnvalue -1 = Error	

Parameter	Type
Return variable (global script variable)	uint
Script variable (source) (global script variable)	uint

Error code	Description
-1	String contains an illegal character

Fig. 6-73: Return variable for method DoubleToString

IntToString(return variable, integer value, target script variable)

Converts the integer value into a string. The string is written to the target script variable.

Method	Return type
IntToString	void
Returnvalue 0 = OK Returnvalue -1 = Error	

Parameter	Type
Return variable (global script variable)	uint
Integer value	int
Script variable (target) (global script variable)	uint

Error code	Description
-1	Integer value is larger than destination memory

Fig. 6-74: Return variable for method IntToString

StringToInt(return variable, source script variable)

Converts a string contained in a script variable into an integer value. The return variable contains the integer value.

Method	**Return type**
StringToInt	int
Returnvalue 0 = OK Returnvalue -1 = Error	

Parameter	**Type**
Return variable (global script variable)	uint
Script variable (source) (global script variable)	uint

Error code	**Description**
-1	String contains an illegal character

Fig. 6-75: Return variable for method StringToInt

6.6.4.6 Methods for ScriptVarString Class

AppendString(script variable, string)

Appends a string to a script variable.

Method	**Return type**
AppendString	uint
Returnvalue >0 = Length of the string Returnvalue 0xFFFFFFFFu = Error	

Parameter	**Type**
Script variable (global script variable)	string
String	"String"

AppendVar(target script variable, source script variable)

Appends the string from the source script variable to the end of the target script variable.

Method	**Return type**
AppendVar	uint
Returnvalue >0 = Length of the string Returnvalue 0xFFFFFFFFu = Error	

Parameter	**Type**
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint

CopySubstring(target script variable, source script variable, start position)

Reads the remaining string starting from the specified start position of the source script variable and inserts it at the end of the string in the target script variable.

Method	**Return type**
CopySubstring	uint
Returnvalue >0 = Length of the string Returnvalue 0xFFFFFFFFu = Error	

Parameter	**Type**
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint
Start position	uint

CopySubstringLength(target script variable, source script variable, start position, string length)

Reads the string of the specified string length starting from the specified start position of the source script variable and inserts it at the end of the string in the target script variable.

Method	**Return type**
CopySubstringLength	uint
Returnvalue >0 = Length of the string Returnvalue 0xFFFFFFFFu = Error	

Parameter	**Type**
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint
Start position	uint
String length	uint

Length(script variable)

Determines the length of the string in a script variable.

Method	**Return type**
Length	uint
Returnvalue >0 = Length of the string Returnvalue FFFFFFFF = Error	

Parameter	**Type**
Script variable (global script variable)	uint

CopyString(target script variable, start position, string)

Overwrites the contents of the target script variable with the string as of the given position.

Method	**Return type**
CopyString	uint
Returnvalue >0 = Length of the string Returnvalue FFFFFFFF = Error	

Parameter	**Type**
Script variable (target) (global script variable)	uint
Start position	uint
String	"String"

GetChar(script variable, position)

Reads a character from the specified position of a script variable.

Method	**Return type**
GetChar	uint
Returnvalue >0 = Length of the string Returnvalue FFFFFFFF = Error	

Parameter	**Type**
Script variable (global script variable)	uint
Position	uint

6.6.4.7 Methods for SysVar Class

With this class you access system variables from the script.

In the event of an error an error code is written to the return variable.

Error code	Description
40	System variable is not supported by the current firmware

Fig. 6-76: Return variable for all methods of SysVar class

GetUInt(return variable, system variable)

Passes the value of a system variable to a script variable.

Method	Return type
GetUInt	uint
Returnvalue 0 = OK Returnvalue >0 = Error	

Parameter	Type
Return variable (global script variable)	uint
System variable	uint

SetUInt(return variable, system variable, value)

Passes a value to a system variable.

Method	Return type
SetUInt	bool
Returnvalue 0 = OK Returnvalue >0 = Error	

Parameter	Type
Return variable (global script variable)	uint
System variable	uint

GetString(return variable, target script variable, source system variable)

Passes the value of a system variable of the type "String" to a script variable.

Method	Return type
GetString	bool
Returnvalue 0 = OK Returnvalue >0 = Error	

Parameter	Type
Return variable (global script variable)	uint
Script variable (target) (global script variable)	uint
System variable (source)	uint

6.6.5 Data Types for Scripts

Variables can be used with the following data types:

Type	Range
bool	false to true
int	32-bit integer with a sign from -2147483648 to 2147483647
uint	32-bit integer without a sign from 0 to 4294967295
double	64-bit with a single precision in the IEEE754 format

Fig. 6-77: Data types for script variables

6.6.6 Keywords

The script interpreter works with the following keywords:

Word	Description
bool	Indicator for boolean data type
break	Interrupts the script
double	Indicator for double data type
else	Keyword for IF control structure
false	Value for a boolean expression (0)
if	Keyword for IF control structure

Fig. 6-78: Keywords for scripts

Word	Description
int	Indicator for INT data type
private	Local function within a script
return	Terminates a method
true	Value for a boolean expression (1)
void	Return type for a method which does not return a value
while	Keyword for WHILE control structure

Fig. 6-78: Keywords for scripts

6.6.7 Operators

The script interpreter processes the following operators.

Operator	Description
+	Addition (int, uint, double)
-	Subtraction (int, uint, double)
*	Multiplication (int, uint, double)
/	Division (int, uint, double)
<<	Shift to the left (int, uint - second parameter must be 'int'!)
>>	Shift to the right (int, uint - second parameter must be 'int'!)
~	Bit-by-bit complement (int, uint)
^	Bit-by-bit XOR (int, uint, bool)
&	Bit-by-bit AND (int, uint)
|	Bit-by-bit OR (int, uint)
%	Modulo (remainder after a division) (int, uint)
!	Negation (for "bool" only)
&&	Logical AND (for 'bool' only)
||	Logical OR (for 'bool' only)
=	Assignment (all data types)
(typ)	Cast assignment (converts a data type)
()	Specification of the order of operations in an expression (all data types)

Fig. 6-79: Script operators

6.6.8 Comparison Operators

The script interpreter processes the following comparison operators.

Operator	Function
<	less than (int, uint, double)
>	greater than (int, uint, double)
==	is equal to (bool, int, uint, double)
!=	not equal to (bool, int, uint, double)
<=	less than or equal to (int, uint, double)
>=	greater than or equal to (int, uint, double)

Fig. 6-80: Comparison operators for scripts

☞ The comparison operators ==, <= and >= should not be used with the data type "double" since rounding errors occur for this data type.

6.6.9 Control Structures

You can use the following control structures for scripts.

Word	Syntax

if	if (boolean expression) statement
	OR
	if (boolean expression) statement
	else statement

while	while (boolean expression) statement

Fig. 6-81: Control structures for scripts

Example for usage of "if":

```
public void Execute()
{
 f1(1);
}
private void f1(int y)
{
 int z;
 int x=2*y;
 if (x>10)
 {
 z=1;
 }
 else
 {
 z=2;
 }
}
```

If the value of x is greater than 10, the value "1" is written to the variable z. If x is smaller than 10, the value "2" is written.

Example for usage of "while":

```
public void Execute ()
{
 f1(0);
}
private void f1(int y)
{
 int x = 0;
 int z = 0;
 while (x<7)
 {
 z = y + x;
 x++;
 }
}
```

If the value of x is smaller than 7, variable z is offset against the starting value of the transferred value (0) plus the value of variable x. The value of x is incremented for each looping process.In this example, variable z will assume the values 0,1, 2, 3, 4, 5, 6.

6.6.10 Local Variables

Local script variables are placed on the stack of the script interpreter. These variables are not retentive and can only be used within the method.They are capable of passing the value of a local script variable to a local script method as a parameter. Local script variables can not directly be displayed in images. The lifetime of local script variables is limited to the execution time of the method.

Example for a local script variable (localVar1):

```
public void Execute()
{
 int localVar1;
 localVar1 = 10;
}
```

6.6.11 Global Variables

Global script variables are:

- Global script variables
- PLC variables and
- System variables.

You can access these variables using methods. The values of these variables can be displayed and edited in screens.

Enter global script variables into the script variable list.

Example for a global script variable (intVar1):

```
public void Execute()
{
}
private void f1 ()
{
 Firmware.ScriptVar.SetInt(Project.ScriptVar.intVar1, 33);
}
```

The memory for global script variables is limited to the following values:

Device	**Memory**
VCP01	16 KB
VCP02 to VCP 25	32 KB

Fig. 6-82: Memory for global script variables

Global script variables are only retentive if the operating device is equipped with a buffered RAM.

System variables are internally known to the system and are not declared separately.

Example for a system variable (RtcSec):

```
// Writing the value of the system variable into the local variable "Sec-
onds"public void Execute()
{
 uint Seconds;
 Seconds = Firmware.SysVar.GetUInt(Firmware.ScriptVar.MyReturncode,
Firmware.SysVar.RtcSec);
}
```

6.7 Working with Recipes

Various logically related variables can be organized into units known as recipes. Unlike screen variables, recipe variables are not transferred to the controller immediately after being entered, but are stored in the operating device as data sets. These data sets are protected against power failure. The data sets can be loaded to the controller as a unit as and when required.

The maximum number of recipes that can be created at programming time is 250. For each recipe, up to 250 data sets can be created. The data sets can either be created at programming time and be stored in the operating device's Flash memory together with the project or can be entered online on the operating device and are then stored in the battery-backed RAM.

You must copy data sets stored in the Flash memory to the RAM first before you can edit them. Data sets that have been edited remain in the battery-backed RAM.

Example:

Settings of a machine for manufacturing various products

Variable	**Value**	**Unit**
Material	ST37-3	
Feedrate	25,00	mm/s
Setpoint Value Axis 1	43,5	mm
Setpoint Value Axis 2	56,30	mm
Cutting Angle	30	°
Cutting Speed	110	mm/s

Fig. 6-83: Recipe for the product 'clamp'

Variable	**Value**	**Unit**
Material	X20Cr13	
Feedrate	20,00	mm/s
Setpoint Value Axis 1	45,6	mm
Setpoint Value Axis 2	51,20	mm
Cutting Angle	45	°
Cutting Speed	76	mm/s

Fig. 6-84: Recipe for the product 'shaft'

The variables Material, Feedrate, Setpoint Value Axis 1, Setpoint Value Axis 2, Cutting Angle and Cutting Speed can be organized into the recipe "Machine Settings for Products".

The variables Feedrate, Setpoint Value Axis 1 and Setpoint Value Axis 2 are defined as floating point numbers or fixed point numbers. The variable Cutting Angle is defined as an integer and the variable Material as a selection text (coded text).

The values for manufacturing the products Clamp and Shaft must be stored as data sets. Whenever another product is to be manufactured, the data set of the product to be manufactured next can be loaded into the controller.

The following check list contains all of the elements that are required and useful for creating and handling a recipe with data sets:

- The recipe itself (texts and variables)
- Data sets with data set number, data set name and variable offset
- I/O screens for the recipe
- Recipe field in the screen
- Recipe buffer (address for the data area in the controller)
- Variable Data Set Number for Transfer from operating device

- Variable Recipe Number for Transfer from operating device
- Variable Data Set Number for Request from controller
- Variable Recipe Number for Request from controller
- System variables:

System Vari-able	**Linked to**	**Description**
SelectDSNr	Selection Text/Decimal Number	Display/Select Data Set Number
SelectDSName	Selection Text Variable	Display/Select Data Set Names
DestDSNr	Positive Decimal Number	Destination Data Set Number for Copy Pro-cess
DSCopy	Softkey / Selection Text Variable	Activate 'Copy Data Set'
DSDelete	Softkey / Selection Text Variable	Delete Data Set
DSDownload	Softkey / Selection Text Variable	Load Data Set in Controller
DSDnload-Break	Softkey / Selection Text Variable	Stop Data Set Transfer
DSDnloadState	Selection Text Variable	Display Transfer Status
ActDSName	Alphanumeric Variable	Enter Name for RAM Data Set
Selec-tRezeptNr	Selection Text/Decimal Number	Display/Select Recipe Number
TabPgUp	Softkey	Page Up
TabPgDn	Softkey	Page Down
Break	Softkey	Cancel Input
LoadDSName	Selection Text Variable	Display Name of Last Data Set Transferred
StartSave	Softkey / Selection Text Variable	Data Set Transfer from Operating Device to PC
SaveState	Selection Text Variable	Display Transfer Status
StartRestore	Softkey / Selection Text Variable	Data Set Transfer from PC to Operating De-vice
RestoreState	Selection Text Variable	Display Transfer Status
RestoreLineNr	Positive Decimal Number	Display Current Transfer Line
StartRezPrint	Softkey / Selection Text Variable	Print Active Data Set
RezPrintState	Selection Text Variable	Display Printer Status
StartUpload	Softkey / Selection Text Variable	Data Set Transfer from Controller to Operat-ing Device
UploadDSNr	Positive Decimal Number	Destination Data Set Number for Upload
UploadState	Selection Text Variable	Display Transfer Status

Fig. 6-85: System variables for recipes

6.7.1 Structure of a Recipe

A recipe comprises a maximum of 255 variables. In addition, up to 255 explanatory texts can be programmed. The variables and texts can be spread out over a maximum of 255 lines (with each line stretching across the entire width of the screen). A help text can be programmed for every variable.

The recipe is displayed in a recipe field, within an I/O screen, that extends over the entire width of the screen. The height of the recipe field can be as small as one line or as large as the entire height of the screen. The Cursor keys can be used to scroll through long recipes in the recipe field.

All one-line display formats can be used for recipe variables. Multiple-line formats can not be used (for example, multiple-line selection fields, tables, etc.). In addition, neither variables nor texts can be displayed with the zoom option.

6.7.2 Working with Recipes and Data Sets

The majority of the operations described below refer to the active data set. In order to activate a data set, first select the recipe to which it belongs and then the data set itself. How to select recipes and data sets is explained in the next two sections.

6.7.2.1 Selecting a Recipe

Each recipe is assigned a number from 1 to 250 when the recipes are programmed.

You can select a recipe as follows:

- By means of a fixed assignment between the recipe and a screen. This means, that whenever you open the corresponding screen, the recipe field will contain the recipe that was specified when programming was carried out. If you do not permanently assign a recipe to a screen with a recipe field during the programming phase, the last recipe that was processed appears when the screen is opened.
- By means of the system variable **SelectRezeptNr**. You can edit the system variable using any Editor. It is a good idea, however, to use a selection text (coded text) and assign meaningful recipe names to each recipe number.

See chapter “SelectRezeptNr“ on page 6-202.

6.7.2.2 Selecting a Data Set

Data sets can be assigned both a number from 1 to 250 and a name.

You assign the data set numbers and names when the data sets are created, in other words either when programming is carried out for the data sets stored in the Flash memory or on the operating device in the case of data sets stored in the RAM. The maximum data set name length is 15 characters. Data set names need not necessarily be unique (though it is recommended that they are).

You can select a data set as follows:

- Select a new recipe. The associated data set with the lowest number is then selected for the new recipe automatically.
- By means of the system variable **SelectDSNr**. You can edit this system variable only as a selection text. In this case, only the numbers of those data sets that are available for the active recipe are displayed.
- By means of the system variable **SelectDSName**. You can edit the system variable only as a selection text. In this case, only the names of those data sets that are available for the active recipe are displayed.

See chapter “SelectDSNr“ on page 6-199.
See chapter “SelectDSName“ on page 6-200.

6.7.2.3 Copying a Data Set

You can only copy the active data set. To do so, write the number of the destination data set to the system variable **DestDSNr** and then write the value 1 to the system variable **DSCopy**.

The following conditions must be fulfilled in order for the data set to be copied successfully:

- The number of the destination data set must be in the range of 1 to 250.
- There must not already be a data set with the same number for the active recipe (unless **DSCopy**is set to 3).
- The active data set can not be edited at the same time.
- There must be enough free RAM on the operating device.

If any of these conditions is not satisfied, the copy process is not carried out and a corresponding terminal message is issued.

The destination data set becomes the active data set after it has been copied.

After it has been copied, the name of the destination data set consists merely of blanks. You can use the system variable **ActDSName** to change the name.

See chapter “DestDSNr“ on page 6-200.
See chapter “DSCopy“ on page 6-200.
See chapter “ActDSName“ on page 6-201.

6.7.2.4 Deleting a Data Set

You can only delete the active data set. To do so, you need to write the value 1 to the system variable **DSDelete**.

The following conditions must be fulfilled in order for the data set to be deleted successfully:

- The active data set can not be edited at the same time.
- The data set must be stored in the RAM.

If any of these conditions is not satisfied, the delete process is not carried out and a corresponding terminal message is issued.

After the deletion, the data set with the lowest number in the current recipe becomes the active data set.

See chapter “DSDelete“ on page 6-201.

6.7.2.5 Modifying a Data Set

The active data set can be modified, providing it is stored in the RAM.

To change the contents of a data set, the variables must be edited in the recipe window. Note, however, that the new values are not written in the data set as soon as the Enter key is pressed, but are first stored in a temporary buffer.

The Data Release key must then be pressed in order to enter them into the data set. If the new data is not to be entered, the system variable **Break** can be set to 1 to discard the contents of the buffer. For ease of use, you might want to program one of the softkeys or a specific button to the system variable **Break**.

You can not select another data set until the buffer contents has either been accepted or discarded.

If the controller changes to a different screen while a data set is being modified, or if the external data release is canceled again before you press the Data Release key, the buffer contents will likewise be discarded.

The modified data set is not transferred to the controller automatically.

An explicit command from you or the controller is necessary first.

☞ See chapter “Break“ on page 6-210.
See chapter “Transfer Single Data Set from Operating Device to Controller“ on page 6-229.

6.7.3 Data Set Transfer to/from Controller

You can load the data sets in the operating device to the controller. You can also load (any changed) data sets from the controller to the operating device. In this context, the data set transfer is always initiated by the operating device, but only when the controller has activated the corresponding release (Data Set Download Release bit in the Write Coordination byte).

Before a transfer is performed, the communication partner must be advised of the recipe number and data set number.

In the **Transfer from Terminal** area, enter the name of the variable for the recipe and data set number. To do this, click the button beside the input box. The operating device automatically enters the recipe number and data set number into this variable before data are transferred to the controller.

Tutorial

6.7.3.1 Transfer to the Controller (Operator-Controlled)

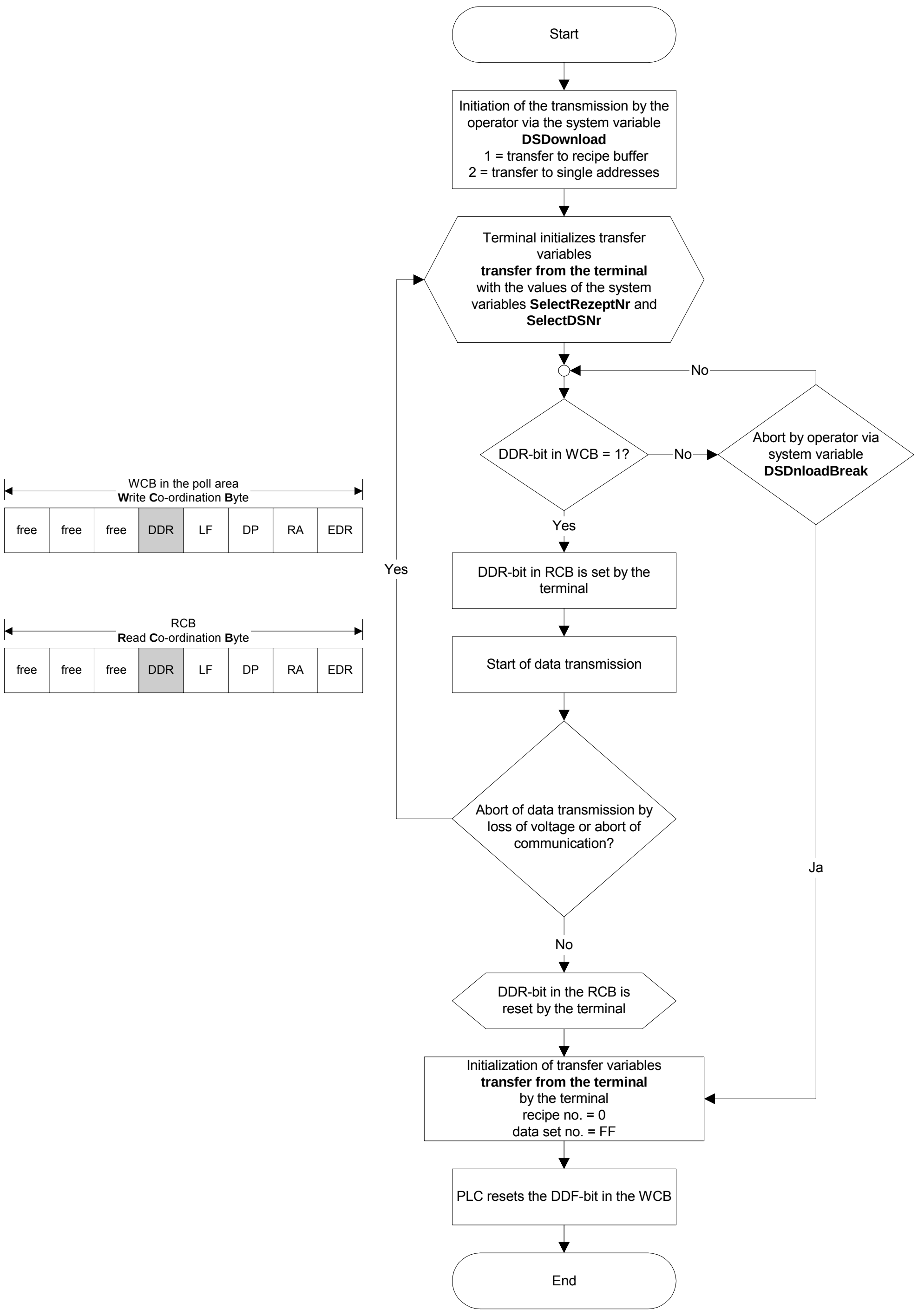

Start
Initiation of the transmission by the operator via the system variable **DSDownload** 1 = transfer to recipe buffer 2 = transfer to single addresses
Terminal initializes transfer variables **transfer from the terminal** with the values of the system variables **SelectRezeptNr** and **SelectDSNr**
No
DDR-bit in WCB = 1?
No
Abort by operator via system variable **DSDnloadBreak**
Yes
WCB in the poll area **W**rite **C**o-ordination **B**yte
free free free DDR LF DP RA EDR
Yes
DDR-bit in RCB is set by the terminal
RCB **R**ead **C**o-ordination **B**yte
free free free DDR LF DP RA EDR
Start of data transmission
Abort of data transmission by loss of voltage or abort of communication?
Ja
No
DDR-bit in the RCB is reset by the terminal
Initialization of transfer variables **transfer from the terminal** by the terminal recipe no. = 0 data set no. = FF
PLC resets the DDF-bit in the WCB
End

Fig. 6-86: Data transfer to the controller (operator-controlled)

6.7.3.2 Transfer to the Operating Device (Operator-Controlled)

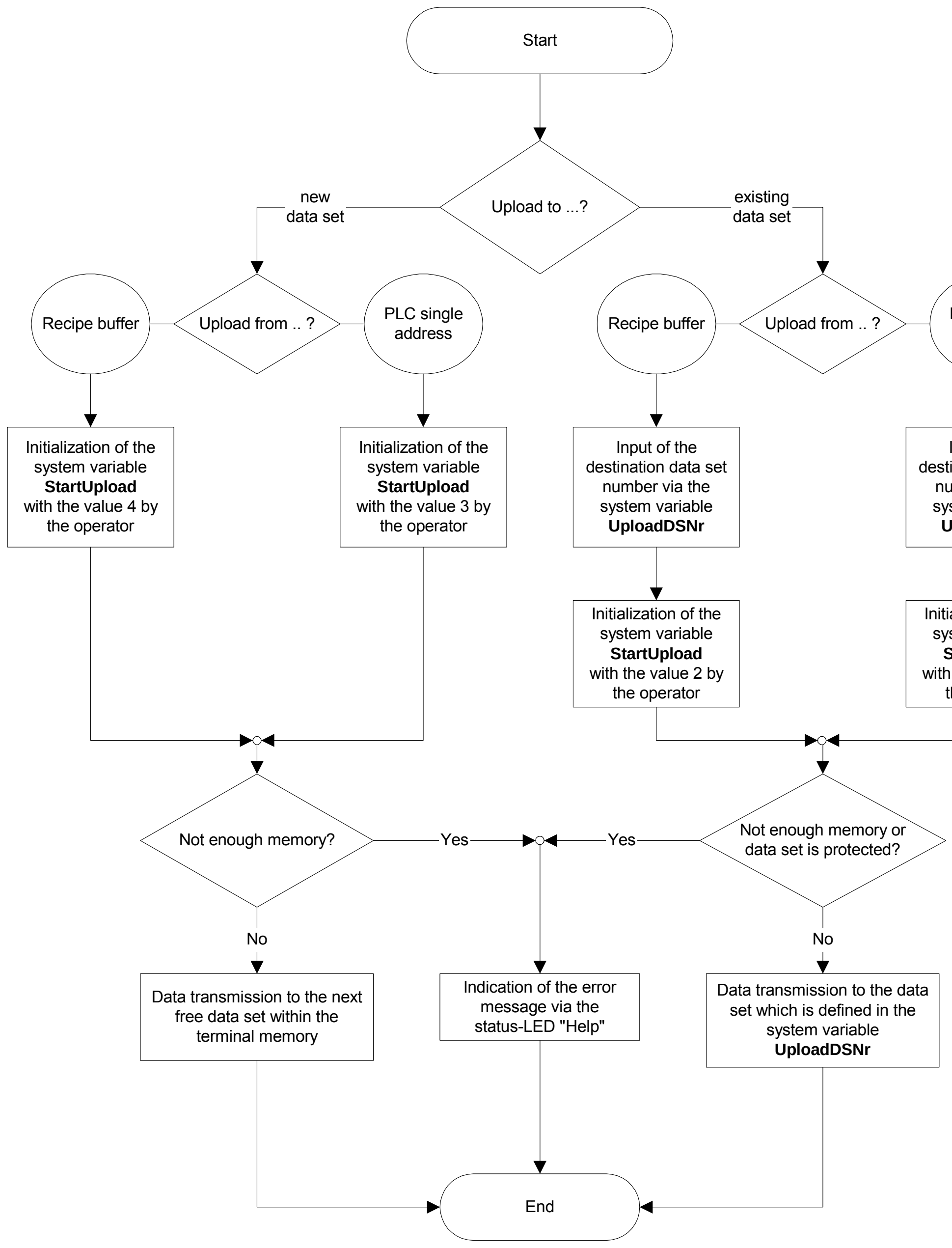

Start
Upload to ...?
new data set
existing data set
Recipe buffer
Upload from .. ?
PLC single address
Recipe buffer
Upload from .. ?
PLC single address
Initialization of the system variable **StartUpload** with the value 4 by the operator
Initialization of the system variable **StartUpload** with the value 3 by the operator
Input of the destination data set number via the system variable **UploadDSNr**
Input of the destination data set number via the system variable **UploadDSNr**
Initialization of the system variable **StartUpload** with the value 2 by the operator
Initialization of the system variable **StartUpload** with the value 1 by the operator
Not enough memory?
Yes
Yes
Not enough memory or data set is protected?
No
No
Data transmission to the next free data set within the terminal memory
Indication of the error message via the status-LED "Help"
Data transmission to the data set which is defined in the system variable **UploadDSNr**
End

Fig. 6-87: Data transfer to the operating device (operator-controlled)

6.7.3.3 Transferring Data Sets to / from a PC

This function is available for all operating terminals with a SER2 serial interface.

It is possible to transfer data sets to or from a PC via the SER2 interface, in order to back up the data sets stored in the small operator terminal, to process the data or to supply the small operator terminal with new data sets.

It is also particularly important to back up the data sets if a new application description is loaded in the small operator terminal, as all the data sets in the RAM are then deleted. If the recipe structure remains unchanged, however, they can be reloaded into the small operator terminal again after the terminal file has been loaded. If changes have been made to the structure of any of the recipes (number of variables, position of the variables in the data set buffer, etc.), only the data sets of the other, unchanged recipes can be reloaded into the small operator terminal .

The data sets are transferred in a format that can be edited with a text editor.

The parameters for the SER2 interface can be freely configured by means of the corresponding system variables. Make sure that the same parameters are defined on the PC-end.

Start the recipe data set transfer tool to transfer recipes and data sets.

To start the recipe data set transfer tool (transfer: terminal to/from file):

1. Select **Transmit recipe data sets** from the **Tools** menu.

To start the recipe data set transfer tool (transfer: terminal to/from open project):

1. Open the branch "Recipes" in the project tree.
2. Select a recipe.
3. Select **Transmit recipe data sets** from the **Tools** menu.

See chapter “Tools Menu, Transmit Recipe Data Sets“ on page 5-19.

6.7.3.4 Transfer to a PC

The transfer of data sets to the PC is initiated by writing a value to the system variable StartSave. The number of data sets that are transferred depends on the value that is written to the system variable. The following are valid values:

System variable value = 1: Only the active data set is transferred.

System variable value = 2: All of the data sets of the active recipe are transferred.

System variable value = 3: All of the data sets of all recipes are transferred.

The process can be monitored by the operator with the aid of the system variable SaveState.

6.7.3.5 Transfer from a PC

The operating device is placed to the Ready-to-Receive state when the system variable StartRestore is set to 1. The data sets can then be sent by the PC. The operating device recognizes the end of the data set transfer automatically by analyzing the data it has received. It then returns to its normal state.

To cancel the Ready-to-Receive state again without receiving data, the value of the system variable StartRestore must be changed to 2.

The system variable RestoreState indicates whether or not the terminal is ready to receive.

If a formatting error is detected in the received data, a system message to this effect is output and the receive process is terminated. The position of the formatting error can be located, at least approximately, with the aid of the system variable RestoreLineNr. This system variable contains the number of the last line to have been received.

Data sets can only be stored in the operating device if their structure is still identical to the data set structure specified for the corresponding recipe in the application description. This can be checked by the operating device on the basis of a version number (see Structure of Data Set File). If a data set which is found to be invalid is received, it is rejected and a system message to this effect is output. The receive process is not terminated, however.

If a data set with the same number as the transferred data set is already stored in the Flash Eprom, the newly received data set is rejected without any warning to the operator.

If a data set with the same number as the transferred data set is already stored in the RAM, a parameter setting in the received data (see Structure of Data Set File) determines whether or not the existing data set is overwritten. If it is not supposed to be overwritten, and another data set with the same number already exists in the operating device, the newly received data set is similarly rejected without any warning to the operator.

6.7.3.6 Structure of a Data Set File

The data sets transferred to the PC are generally stored in a file.

If this file is only used for backup purposes, the operator does not necessarily be familiar with its structure. In this case, the file can merely be transferred back to the operating device unchanged when it is needed.

If the data are to be processed further, for example, within the scope of production data acquisition, the user should understand the structure of the file.

All of the data in the data set file are represented by a simple language specifically developed for this purpose.

The following are elements of this language:

Key words:

S + two further letters. They normally appear at the beginning of a line. Example: SDW or SFA

Decimal number:

Any number of the digits 0-9, preceded by a negative sign when required. Example: 999 or -1234567

Hexadecimal number:

H + any number of the digits 0-9 or letters A-F or a-f. Example: H999 or H123abCD4

Hexadecimal string:

C + any even number of the digits 0-9 or letters A-F or a-f.

Example: C12 or CAAFF33

ASCII string:

Any string of characters enclosed between two backslash characters (\) .

Example: \This is one ASCII string\

Comment:

Any string of characters enclosed between two dollar signs ($). Comments can be inserted in the data set backup file at any position and can stretch across several lines.

Example: $This is a comment$

Any number of separators (blanks, tab characters or line feed characters) can be placed between these language elements.

The above-mentioned language elements are used to create a file with the following structure:

- Start of file identifier
- Any number of data sets
- End of file identifier

A data set consists of:

- Data set header
- Any number of data set variables

- End of data set identifier

Start of File Identifier	
Key	SFA
Parameter	none (date and time are output by the operating device as a comment)

Fig. 6-88: Start of file identifier

End of File Identifier	
Key	SFE
Parameter	none

Fig. 6-89: End of file identifier

Data Set Header	
Key	SDK
Parameter	Recipe number, data set number, data set name (as an ASCII string), data set size in bytes, recipe version number, write-over identifier

Fig. 6-90: Data set header

Data Set Variables	
Key	SDW
Parameter	Offset of the variables in the recipe, variable size in bytes, value of the variables (as a hexadecimal string)

Fig. 6-91: Data set variables

End of Data Set Identifier	
Key	SDE
Parameter	none

Fig. 6-92: End of data set identifier

Explanations:

Recipe version number

On creating or changing the recipe description in the programming software, this version number is increased automatically whenever the structure of the data sets has changed. To be able to load a data set from the PC to the operating device, the downloaded version number and the version number stored in the operating device for the recipe involved must match. The downloaded data set will not be stored if the version numbers do not match.

Write-over identifier:

The value 1 means that the downloaded data set is to overwrite any data set with the same number that may already exist in the operating device. The value 0 means that the downloaded set is to be rejected if a data set with the same number already exists. Only those data sets can be overwritten that are not stored in the Flash memory, i.e. that were loaded into the operating device together with the project.

6.7.3.7 Printing Data Sets

The data set print process can be started from both the operating device and the controller. To be able to initiate a print process from the operating device, either the system variable **StartRezPrint** must be placed into a screen or a softkey must be assigned accordingly.

The active data set is printed via the SER1 interface when a 1 is written to the system variable.

Writing the value 2 to the same system variable will cancel the print process.

A heading including the recipe number, data set number and data set name will be printed at the beginning of each data set.

The status of the print process can be displayed through the system variable **RezPrintState**.

To be able to control a print process from the controller, the data set number and recipe number must be entered into the appropriate variables first. You then start the print job by writing the value **7FF8H** to the address of the serial message channel.

A value of 0 (zero) in the variable for the recipe number (for request from the operating device) will indicate that the data set is being printed.

If another print job is currently being printed so that the printer can not print the specified data set, the value 255 will be written to the variable for the recipe number (for request from the operating device).

6.7.3.8 Memory Requirement for Data Sets

The RAM in the operating device that is not required by the system (approximately 110000 bytes) is used to store messages as well as data sets that have been stored in the RAM.

The size of the message buffer is configurable. Each message takes up 24 bytes. This makes a total of 12000 bytes for the default message buffer size (500 messages), so that a further 98000 bytes are available for storing data sets.

Space is also needed to store the data set name and management information (additional 28 bytes per data set).

Example:

If the data set size is programmed as 22 bytes, a total of

98000 / (22 + 28) = 1960

data sets can be saved in the RAM (message buffer size: 500). Other, fixed programmed data sets can also be stored in the Flash Eprom.

6.8 Working with Messages

You can specify general parameters and parameters for the serial and the parallel message system.

General Parameters

You can enter a message number to display a message directly. The message number also indicates its priority. The lowest message number has the highest priority and the highest message number has the lowest priority.

All messages that have a higher priority than the message number specified here are handled using a special procedure in the operating device when they appear. These messages are indicated by flashing of the status LED in the direct selection key of the message screen or are signaled by a terminal message.

Using the Size of Message Buffer, you define how many messages can be stored in the operating device. Specify the maximum number of messages to be managed by the operating device.

You can have the messages displayed in indented format. To do so, specify the number of characters by which the lines are to be indented after the first line. The value you enter here refers only to the display of messages in the operating device's message field.

The same options are available for outputting messages to the logging printer.

You can choose from four variants for outputting to the printer.

1. Print the entire message, SER2 reserved exclusively for message output
2. Formatted printout, SER2 reserved exclusively for message output
3. Print the complete message, SER2 reserved only temporarily for message output
4. Formatted printout, SER2 reserved only temporarily for message output

Serial Message System

From the controller, you can **erase all acknowledged messages** if you write the bit pattern **E216h** to the controller address **Delete Messages** and write the control code **7FF5h** to the serial message channel.

If you want to **delete all messages**, you need to write the control code **7FFEh** to the serial message channel. If you want to use this function of the operating device, you must assign a variable for deleting messages.

Parallel Message System

For the parallel message system, you must enter a Variable for Status Messages as the start address of the data area where the messages are stored in the controller in bit-coded form.

You can also specify a name for a Variable for Acknowledging Status Messages of the same size.

You also define the number of bytes to define the Size in Bytes of the area for the status messages in the controller.

You can define a maximum of 256 bytes for this area.

By entering the polling time, you also specify the interval at which the operating device reads the data area of the status messages from the controller.

For the polling time, you can enter values from 0 to 25.5 seconds.

The active messages are displayed in an I/O screen with a message field for parallel messages. The status messages can be sorted according to various criterion.

6.8.1 Internal Messages

Internal messages are all messages that are generated by the operating system. A distinction is made between terminal messages and error messages. The user (programmer) can not influence the generation of these messages.

6.8.2 System Icon

The system icon is used to display and retrieve terminal-internal information (e.g. terminal messages). It can be configured for touch-sensitive terminals. The system icon replaces the function of the help status LED and is used to display pending terminal messages. The pending terminal message is acknowledged by pressing the system icon.

To configure a system icon:

1. Open the relevant screen.
2. Press the right mouse button and select the option **Generate system icon**.
3. A wizard is displayed assisting you in specifying the colors and the frame for the icon.
4. When finished, press **Finish**.

Two buttons will then be displayed (top left of the screen):

- one button with the system icon function and
- and one button with a "?" which is used as a help key to switch to the terminal message screen.

Note:

When working with system icons, configure one icon for each terminal message. In addition, define a **default image** for the system icon. To do this, go to the **Touch parameters** of the operating device properties.

Repeat the following steps for each terminal message (or use a project template:

- Select a terminal message.
- Click **Touch terminals**in the properties window.
- There, enter the system icon.

Number	Brief Descriptions	System icon
1	Wrong format	
2	Value too large	
3	Value too small	
4	Replace battery	
5	Message overflow	
6	New message	
7	Message buffer full	
8	Invalid screen no	

Fig. 6-93: Terminal messages and icons

Number	**Brief Descriptions**	**System icon**
9	Invalid message no.	
10	Print log invalid	
11	Interface in use	
12	Invalid password	
13	Password unchanged	
14	Overvoltage	
15	Data set protected	
16	Illegal data set	
17	Data set unknown	
18	Data set memory full	
19	Data set active	
20	Data set transfer	
21	Password missing	
22	Editing mode active	
23	Data set file error	
24	Data set format	
25	Number invalid	
26	Loop-through active	
27	No data set address	

Fig. 6-93: Terminal messages and icons

Number	Brief Descriptions	System icon
28	Recipe unknown	
29	Data set download	
30	Scanner error	
31	Print log unknown	
32	Language switching error	
33	Flashcard information	
34	New application	NEW S3
35	Format error	
36	Script error	

Fig. 6-93: Terminal messages and icons

6.8.3 Suppressing the Display of Terminal Messages

You can prevent terminal messages from being displayed by deleting the corresponding text. The entry of the terminal message in the project management function remains existent.

Example:

Terminal message 7 - "Message buffer full" is to be suppressed. Older messages or messages with a lower priority are to be overwritten.

Delete the terminal message text in the project management function.

By suppressing the display of this terminal message, the user agrees that incoming messages automatically overwrite the oldest messages or those with the lowest priority once the message buffer is full.

6.8.4 Error Messages

The messages listed here are displayed by the operating system in English. The size of the texts has been chosen in such a way that they can be displayed on every operating device.

The text output can not be suppressed and the texts can not be modified. The term "error message" is used because the terminal does not operate in accordance with the true meaning of the standard mode while these messages are displayed. In addition to true system errors, various conditions and processes are also described.

COMMUNICATION ERROR

COMMUNICATION ERROR	
CODE	X
SUBCODE:	X
RETRIES:	XXX

This message is generated for all types of protocol and interface errors. The error codes (CODE X) and SUBCODE (X) are protocol-specific and are listed in the respective description in the chapter on controller and bus connections. The connection with the communication partner has been interrupted. RETRIES displays the number of unsuccessful attempts to establish a connection. This number is incremented while the device is running. The number of retries depends on the protocol that is being used.

ADDRESS ERROR

```
ADDRESS ERROR
```

This message may be displayed during a download. The S3 file addresses physical addresses in the operating device. The transmission is aborted as soon as invalid addresses are detected during this process. The starting address of the invalid line in the S3 file is specified in hexadecimal format.

FLASH MEMORY FAILURE

```
FLASH MEMORY FAILURE
```

Is displayed during a download if the Flash Eprom can not be programmed. This message indicates that the application memory is defective. The starting address of the invalid line in the S3 file is specified in hexadecimal format.

CHECKSUM ERROR

```
CHECKSUM ERROR
```

Error during transmission of the application description. The error has either occurred during the serial transmission or the S3 file contains invalid lines or no valid S3 file has been transmitted. Recompile the application description and attempt to retransmit.

BYTECOUNT OVERFLOW

```
BYTECOUNT OVERFLOW
```

Error during transmission of the application description. An error was detected in the S3 file of the application description. More bytes were received in one of the transmission lines than specified in the byte count.

FORMAT ERROR

```
FORMAT ERROR
```

Transmission format of the application description contains errors. The output file used has not been generated by this programming system. The transmitted file did not contain S0, S3 or S7 lines, no S3 format was used.

TURN POWER OFF

```
1. TURN POWER OFF
2. RESET DIP-SW 4
OTHERWISE ALL FLASH-
DATA WILL BE LOST !!
```

The user mode switch S4 was at the "on" position when the supply voltage for the operating device was switched on. The Flash data will be retained if the following instructions are complied with. Switch the device off, set S4 to "off" position, switch device on - data will be retained and the device will function as before. If S4 is set to the off-position while power is on - data will be lost, the device switches to the download mode!

DIFFERENT ID-No BETWEEN TERMINAL AND PROJECT

```
DIFFERENT ID-No
BETWEEN TERMINAL
AND PROJECT.
TERMINAL   ID-No:     8.0
PROJECT    ID-No:     7.0
```

The version of the programming system and the operating system in the operating device are not compatible. This error occurs if the wrong operating system version was selected for compilation of the application description. The two program versions must match.

DIFFERENT DRIV VERS

```
DIFFERENT DRIV VERS
           EPROM      FLASH
VERSION    XXX        XXX
REVISION   XXX        XXX
```

The protocol driver loaded via the programming system and the operating device’s operating system do not match. The two program versions must match.

NONE DEFAULT PARAMETERS ON SERIAL PORT X2 USED

```
! ! ! ! ! WARNING ! ! ! ! !
NONE DEFAULT PARA-
METERS ON SERIAL
PORT X2 USED
```

The parameters of the interface SER1 (X2) were modified. To achieve an operational connection, both communication partners must be set to the new parameters. This message is used for informational purposes if the connection to the communication partner can not be established.

NO PROTOCOL-DRIVER IN PROJECT FOUND

```
! ! ! ! ! ERROR ! ! ! ! !
NO PROTOCOL-DRIVER
IN PROJECT FOUND
```

The operating system can not find a protocol driver in the application description loaded. Select a protocol, recompile the application description and activate another download.

PLC TYPE MISMATCH BETWEEN TERMINAL AND PROJECT

```
! ! ! ! ! ERROR ! ! ! ! !
PLC TYPE MISMATCH
BETWEEN TERMINAL
AND PROJECT
```

The protocol selected in the programming system when creating the application description and the operating device's hardware are not compatible. For example, the Interbus protocol driver has been loaded to a device with standard interfaces.

KEYBOARD ERROR

```
KEYBOARD ERROR
PLEASE RELEASE KEY
```

A self-test is performed and the keyboard is checked when the operating device is switched on. Make sure no keys are pressed during this process. Please follow the request. If this message is issued when no key is pressed, it indicates that the keyboard is defective!

INITIALIZING MESSAGE BUFFER

```
INITIALIZING
MESSAGE BUFFER
```

When the operating device is switched on, all messages in the operating device are sorted. This initialization process requires a certain length of time based on the number of stored messages. The message is always generated, but is only displayed for a very short time period or is not visible at all.

ERASE FLASH EPROM

```
ERASE FLASH EPROM
```

Is displayed while the application memory is being erased. All of the programmed data are erased at this point.

FLASH IS ERASED

```
FLASH IS ERASED
FLASH XXX kBYTE
HFXXXXXX
```

This message appears after the delete process is completed. Interface SER2 (X3) is initialized for download operating mode.

DOWNLOAD 1

```
DOWNLOAD 1
FLASH XXX kBYTE
```

The operating device indicates that it is ready for a download with a baud rate of 19200 Bd via interface X3. A new project can now be loaded or new interface parameters for the transfer can be exchanged.

DOWNLOAD 2

```
DOWNLOAD 2
```

The operating device indicates that it is ready for a download with the new interface parameters. If no data are received within 20 s, the operating device will return to the DOWNLOAD 1 state.

AUTO REBOOT1

```
AUTO REBOOT1
```

The operating device will reboot after a few seconds.

INITIALIZING 1

```
INITIALIZING 1
CPU       XX         MHz
Flash     XXX        kBYTE
XXXXXXX              YYYYYYY
X                    Y
```

The operating device reports its parameters during the startup process:

- CPU frequency in MHz
- Size of Flash memory in Kbytes
- Version number XXXXXXXX
- Loaded PLC driver YYYYYYYY

IDENTIFY MEMORY-TYP

```
IDENTIFY MEMORY-TYP
```

The Flash memory type used is being identified.

HIGH VOLTAGE

```
! ! ! HIGH VOLTAGE ! ! !
```

The voltage applied to the operating device is too high! This message will not disappear until the specified supply voltage has been reached.

ERROR ASYNCHRONOUS SERIAL I/O UNIT 0

```
ERROR ASYNCHRONOUS
SERIAL I/O UNIT 0
```

Initialization of the serial interface (unit 0 or unit 1) failed.

KEYBOARD-MODUL

```
KEYBOARD-MODUL
HARDWARE-VERSION
NOT CONFORM TO
DRIVER-VERSION
```

The program level of the keyboard card and the current firmware are not compatible. Retrofit the operating device. The subcode specifies the level of the keyboard card.

FIRMWARE UPDATE SUCCESSFUL

```
FIRMWARE UPDATE
SUCCESSFUL
AUTO REBOOT
```

Indicates a successful update operation. The operating device reboots automatically.

SYSTEM ERROR

```
SYSTEM ERROR
CODE      :
SUBCODE   :
RETRIES   :
```

A fatal error has been encountered. If this error message is displayed, contact the Bosch Rexroth service in your vicinity. Before calling, make a note of the firmware and hardware version.

UNEXPECTED INTERUPT

```
UNEXPECTED INTERUPT
NR =
IP =
CALL HOTLINE
```

An unexpected interrupt has occurred. Contact the Bosch Rexroth service in your vicinity. Before calling, make a note of the interrupt number (NR) and the program counter number (IP).

FLASH NOT ERASEABLE

```
FLASH NOT ERASEABLE
```

Is displayed after the device has been switched on or prior to a download to indicate that the Flash Eprom can not be erased.

WRONG S3-FILE

```
WRONG S3-FILE
```

Is displayed at the beginning of a download to indicate that the S3 file is not the correct type for the operating device being used.

NO FLASH EPROM

```
NO FLASH EPROM
```

This message is displayed to indicate that no Flash supported by the programming algorithm can be detected.

FLASH CHECKSUM ERROR

```
FLASH CHECKSUM ERROR
```

The application description stored in the FLASH contains errors. This error may occur at the end of a transmission, e.g. if the transmission was incomplete or after a device, with a defective memory, is switched on.

TERMINAL-TYP IS XXXX

```
TERMINAL-TYP IS XXXX
```

An attempt has been made to load a S3 file which was intended for another device type. When this error occurs, the correct type for this operating device is displayed where "XXXX" appears. Recompile using this selection in the programming system.

MEMORY IS FLASH XXXK

```
MEMORY IS FLASH XXXK
```

An attempt has been made to load a S3 file which was created for a larger application memory. The amount of memory space requested by the S3 file and the memory available in the terminal do not match. When this error occurs, the memory size available in the device is specified, in Kbytes, where "XXX" appears. This value must be specified in the programming system when compiling.

FATAL ERROR

```
FATAL ERROR
CODE        :           XXXXX
SUBCODE     :           XXXXX
CALL HOTLINE
```

An error message that should never occur, but which exists nevertheless. The terminal's operating system generates this error if proper operation is no longer possible due to a lack of plausibility. To be able to reproduce the incident, we need to know the code and subcode number as well as the software versions of the operating system and programming software. Do not hesitate to call our hotline and we will help you.

FIRMWARE NOT CONFORM

```
FIRMWARE NOT CONFORM
TO HARDWARE
FIRMWARE    1
HW_VERS.    2
```

If this error message is displayed, contact the Bosch Rexroth service in your vicinity. Before calling, make a note of the firmware and hardware version. The operating system of the operating device switches into an endless loop to prevent damage to the device.

DATASET STORAGE FAILURE

```
DATASET STORAGE
FAILURE
```

A checksum error was detected when checking the memory areas of the recipe data sets. Either the battery or the RAM memory is defective.

6.8.5 External Messages

External messages are generated by the connected controller and forwarded to the operating terminal as information on the monitored process. The user can choose two separate message systems. Depending on the requirements, message transfers to the operating device can be either serial or parallel. This is regardless of whether the messages are process messages or fault messages.

Messages can consist of the message text and a scaled and formatted variable. Every variable type available in the system is valid.

The information in the message memory can be used for statistical evaluations. The message is assigned between the operating device and the controller by means of a message number. The associated texts and variable specifications are stored in the operating device together with the application description. The function of a message and its contents are determined by the user when the application description is created in the programming system.

All of the external messages are stored in the message memory in chronological order or in order of priority. You can optionally store parallel messages in the serial message memory to ensure that they are evaluated statistically as well. If the message contains a variable, its value will be frozen in the message memory.

6.8.5.1 Structure of an External Message

An external message is made up of the following:

- Message number from 1 to 9999
- Date
- Time
- Message text with up to 255 characters
- The values of up to two variables, from the time the message appears (only if available)

When a new project is being created, existing messages can be transferred individually or completely.

6.8.5.2 Message Number

For external messages, the message number also determines the priority of the message. The message with the number 1 has the highest priority, and the message with the number 9999 has the lowest priority. You do not have to assign continuous message numbers.

The assignment of the message numbers in the area for status messages always starts with 1.

From the system, you can also use status message texts in the serial message system.

Make sure that the serial and parallel message systems do not overlap! If you want both message systems to be independent of each other, make sure that the message numbers of the serial system start above the status messages.

☞ If you would like to program full-page message outputs, you must harmonize the message and screen numbers.
See chapter “Full-Page Message Output“ on page 6-147.

☞ See chapter “Full-Page Message Output“ on page 6-147.

6.8.5.3 Message Text and Variable

The text length must not exceed 255 characters, including a formatted variable. The programming system will not allow you to enter texts longer than this.

The standard size of all characters of the operating device specified in each case is permitted. Each message text can contain two output variables.

The output format of the variables is identical to the one-off output variables in input/output screens. In this way, for example, coded texts can be used to modify individual messages or to use them for several statuses.

You can change the output format of the message line during operation in a configuration screen for the message screen.

The same options exist for serial and parallel messages.

Example:

Complete message format:

No.	**Date**	**Time**	**Text 1**	**Variable 1**	**Text 2**	**Variable 2**
1234	**25.08.92**	**11:30:00**	**Temperature**	**285**	**°C at station**	**07**

Explanation of the message structure:

1234	4-digit message number
25.08.92	Date - is recorded when the message is detected in the operating device
11:30:00	Time - is recorded when the message is detected in the operating device
Temperature	Text 1 in front of variable 1
285	value of variable 1 at the time of message generation, is stored in the operating device
°C at station	Text 2 between variable 1 and 2
07	value of variable 2 at the time of message generation, is stored in the operating device

6.8.5.4 Size of the Message Memory

The maximum message memory size allows management of up to 3000 message entries. As the amount of data is considerable, a high performance level is required when sorting the messages, and during resorting and initialization.

As you usually do not require this many entries, you can set the maximum memory size for messages as needed. The basic setting for the message memory allows 500 entries.

When making the setting, take note that for example, you will need about 50 pages of paper to print an entire message memory containing 3000 messages.

The message buffer is output in a screen containing a message field. You can use the system variable **RepmanSortCrit** to define message sorting.

See chapter “Memory Requirement for Messages and Data Sets“ on page 6-151.
See chapter “RepmanSortCrit“ on page 6-173.

6.8.5.5 Message Sorting

You can optionally display messages in the message screen according to their time of arrival or according to their priority. The desired sorting option can be selected when the system is programmed.

If both possible message systems are used, it is possible to select the sorting settings separately. The settings are stored in the system variables **RepmanRepSortCrit** and **RepmanSortCritP**. You can use these system variable to change the settings during operation on the operating device (using a configuration screen, for example).

If you do not give the operator a configuration option, the preselected sorting settings will apply.

Sorting options for the serial message system:

0 - by priority

1 - by time of arrival (most recent first)

2 - by time of arrival (oldest first)

3 - by group

Sorting options for the parallel message system:

0 - by priority

1 - by time of arrival (most recent first)

2 - by time of arrival (oldest first)

3 - by group

6.8.5.6 Message Priority for Direct Display

The priority of a message is determined by its message number.

The higher the message number, the lower the message priority.

The value that represents the upper limit for the message number that is to be indicated on arrival by a flashing LED or by outputting a terminal message can be entered into the system parameters of the message system.

If you enter the value 0, you will not be notified of newly arrived messages!

6.8.5.7 Printing the Message Memory

This function is available for all operating terminals with a SER2 serial interface.

The memory contents of the serial and parallel message systems can be printed either in full or in part.

The entire contents of the message memory of the **serial** message system is printed if the system variable **PrintAllRep** is set to the value '1' (formatted printout) or the value '2' (full-length printout).

The entire content of the message memory of the **parallel** message system is printed if you press a softkey or button linked to the system variable **PrintAllState**.

To print the message memory in part, the messages to be printed must be selected in the message screen. This is done by pressing the **Data Release** key in the message screen (or a corresponding button) and selecting the messages in the message field using the **Cursor Up** and **Cursor Down** keys.

The print job is started by pressing a softkey (or button) linked to the system variable **BlockPrint** (prints visible part of the selected block) or **BlockPrintLong** (prints messages of the selected block in full length). The system variables can additionally be included in a configuration screen and be edited during operation.

6.8.5.8 Direct Call of the Message Screen

In the programming software, you can link a function key or button with a message screen. You can use this function key (the button must be available in each screen) to go from each screen to the message screen. As well as accessing the message screen from a selection menu, you can then also use the function key to access the message screen. The integrated LED of the function key then takes on the task of indicating when new messages have been received. In this case, the LED flashes when a new message has been received.

When you select the flashing function key, the system goes directly to the message screen. When you select the function key again, the system automatically returns to the previous screen. The usual flashing help key LED is not available when programming a function key.

6.8.5.9 Message Output Formats

The following information is available for each external message:

- Message Number
- Date
- Time
- Message text
- The values of up to two variables, from the time the message appears (only if available)

You can use different system parameters to influence message display in a message screen or on a printer. You can set these parameters during operation in a programmed configuration screen.

System variables are then used to select and deselect message elements.

Serial Messages	**Parallel Messages**	**Affects**
RepoutNr	RepoutNrP	Message Number
RepoutDate	RepoutDateP	Date
RepoutTime	RepoutTimeP	Time
RepoutAnzYear	RepoutAnzYearP	2-digit or 4-digit display of the year

Fig. 6-94: System variables for messages

You can select or deselect individual message elements to influence the length of a message line. These settings do not influence the information saved.

The following output variants are available for selection:

Complete message format:

No.	Date	Time	Text 1	Variable 1	Text 2	Variable 2
1234	**25.08.92**	**11:30:00**	**Temperature**	**285**	**°C at station**	**07**

Variants:

1234	**25.08.92**	**Temperature**	**285**	**°C at station**	**07**
1234	**11:30:00**	**Temperature**	**285**	**°C at station**	**07**
1234	**Temperature**	**285**	**°C at station**	**07**	
25.08.92	**11:30:00**	**Temperature**	**285**	**°C at station**	**07**
11:30:00	**Temperature**	**285**	**°C at station**	**07**	
25.08.92	**Temperature**	**285**	**°C at station**	**07**	
Temperature	**285**	**°C at station**	**07**		

6.8.5.10 Zooming Messages

Messages are displayed in a one-line format in the message screen for the sake of clarity. In order to display a longer message in its full length, the message must first be selected and then the Enter key pressed.

Line of the message screen on an operating device displaying 20 characters per line:

1234 25.08.92 11:30:00 Station 137

Zoomed view:

1234 25.08.92 11:30:00

Station 137 in the furnace has a

temperature of 285 °C

The zoomed view remains active for as long as you hold the Data Release key down. With smaller displays (for example, with 4 x 20 characters) only the message text is zoomed. The device type that is to be used must be considered when the text is programmed, to ensure the lines are wrapped correctly.

6.8.5.11 Acknowledging Messages

Message acknowledgment in the controller can be carried out by means of variables. Various editors or function keys (softkeys) are suitable for this purpose. The acknowledgment enables the controller to delete the message and initiate another verification.

6.8.6 Serial Message System

Two bytes are reserved in the cyclical polling area for the transfer of serial messages. These two bytes are referred to as the 'serial message channel'. The byte order depends on the selected data type of the polling area (see Polling Area). The controller writes a 16 bit message number in this message channel.

The operating device polls the entire polling area of the controller at cyclical intervals and transfers the serial message in the process.

Upon detecting a message (message number > 0), the operating device stores this message in the internal message memory and resets the serial message channel in the controller to zero (0). The value 0 indicates to the controller that the message has been picked up by the operating device. The polling time for the serial message channel is configurable.

The same procedure is used to address external screens and message screens. Whenever the number transmitted corresponds to a screen number, this screen is displayed. If a screen and a message text exist for this number, the screen (message screen, full-page fault message text) is displayed and the associated message text is entered into the message memory.

Make sure that the message number is always written to the serial message channel with a 16 bit command!
As a result of asynchronous processing of some data transfer protocols, evaluation of the message number may lead to problems if the message number has been entered with single-byte commands.

6.8.6.1 Full-Page Message Output

The full-page message is a combination of message processing and external screen selection.

For full-page message screen output, a screen and a message text must be programmed with the same number.

The controller calls up the 'external screen' through the serial message channel. When it is called up, the screen is displayed and the associated message text is entered into the message memory. As you can choose the display content freely, it is possible to implement a message screen, full-page error output or other content types.

To be able to return to the previous screen from here, at least one screen parameter must be programmed with the function 'previous screen'. Message screens can consist of several screens or complete structures for error recovery.

A separate, full-page help text can be configured for each full-page message.

6.8.6.2 Outputting Messages to a Logging Printer

This function is available for all operating terminals with a SER2 serial interface.

When serial messages are logged directly, the printer always runs synchronously. Every new message arriving via the serial message channel is printed immediately and is transferred to the message memory in parallel. Here, attention must be paid that the printer can only process one print job at one time. Every print request must be ended before any further print request is started by the system.

You can influence message output to a printer with the system variable **RepmanRepPrint**.

The settings that apply when the formatted type of printout is selected are the same as those selected for the display of messages in the message screen.

The settings for the printout can be changed during operation on the operating device.

As the output consists of a pure text file, the message can also be read by a host computer or a PC. With a further system variable **PrintAllRepLong**, the full length of the message can be output.

6.8.6.3 Erasing the Message Memory Externally

The internal message memory of the serial message system can be erased externally, that is from the controller. To do this, a symbolic variable name for the delete variable must be specified in the Message System option of the system parameters in the programming software. Two bytes are needed in the controller for the variable.

The operating device always checks the delete variable in the controller once it has received the delete sequence (write the control code **7FFE**$_H$ to the serial message channel). The internal message memory is erased when the delete variable contains the bit pattern **E216**$_H$. The delete variable increases protection against unintentional deletion.

If deletion is not required, you should reset the variable or specify no symbolic name in the programming software.

6.8.7 Parallel Message System (Status Messages)

The parallel message system supplements the serial message system. The messages are transferred in parallel and evaluated in the operating device. In this context, the current message status is compared with the previous status in the operating device. The messages that no longer exist are automatically deleted from the memory, and new messages are added to the memory. The current status of the messages can be output.

All messages have a date and time, to enable you to determine when a message appeared for the first time.

The length of the message buffer can not exceed 256 bytes.

Set the length in the system parameters for the message system in the programming software. Certain restrictions may apply to the length, depending on the protocol used.

Status messages are only retained in the message memory for the length of time they are reported by the controller. To set up a message system with acknowledgment, you must have the messages of the parallel message system written to the serial message memory. You must set the transfer of a message from the parallel to the serial message memory separately for each message.

Status messages can be transferred on a time- and/or event-controlled basis.

6.8.8 Settings for Status Messages

6.8.8.1 Size in Bytes

Enter the size of the parallel message system in bytes. You can transfer eight status messages per byte. The absolute size depends on the data type used (address of the variables). For example, the number of bytes is always rounded up for a word address.

The maximum size for the parallel message system is limited to 256 bytes.

Depending on the operating device, different storage quantities are available, which are also used differently for messages and data sets.

This is how you determine the memory requirement in your operating device:

	CPU in Operating Device	
	Z80	**32 Bit RISC**
Available Memory Space in Bytes	About 116000	About 116000 for FW version 1.07 or lower About 147000 for FW version 1.08 or higher
Memory Requirement per Message (=1 Memory Location)	24 Bytes	32 Bytes
Memory Requirement per Data Set without User Data	33 Bytes	44 Bytes

Fig. 6-95: Memory space / memory requirement

You must expect **three times the memory requirement** for the following message variants:

1. Messages containing 2 variables
2. Messages with 'Appear/Disappear'
3. Messages with acknowledgment ('Appear/Disappear' active).
4. Messages whose first variable has a size of greater than 4 bytes

The following table illustrates the memory use for 500 messages and a maximum number of data sets containing 22 bytes of user data:

	CPU in Operating Device	
	Z80	**32 Bit RISC**
Available Memory Space in Bytes	About 116000	About 116000 for FW version 1.07 or lower About 147000 for FW version 1.08 or higher
Memory Requirement per Message in Bytes	24 x 500 = 12000	32 x 500 = 16000
Remaining Memory Capacity in Bytes	104000	About 100000 for FW version 1.07 or lower About 12*4000 for FW version 1.08 or higher
Possible Number of Data Sets	104000 / (22+33) = **1890**	100000 / (22 + 44) = **1515** for FW version 1.07 or lower 1294000 / (22+44) = **1960** for FW version 1.08 or higher

Fig. 6-96: Memory use for 500 messages

You can use the following formula to determine the exact message buffer size:

G >= M1 + M3 + 20

G = Size of message buffer

B = Size of parallel message system in bytes

M1 = Number of messages which: require only 1 memory location, are entered in the message editor and have message numbers smaller than B x 8

M3 = Number of messages which: require 3 memory locations, are entered in the message editor and have message numbers smaller than B x 8

20 = Minimum size of message buffer

6.8.8.2 Polling Time

The polling time determines the intervals at which the variables for status messages are read again.

6.8.8.3 Variables for Status Messages

You must specify the memory address for the parallel message system as a symbolic variable in the system parameters for messages . All variable types that the controller can access bit-by-bit, and the operating device can access byte-by-byte or word-by-word are permitted.

Word	**High Byte**								**Low Byte**							
Byte	**2**								**1**							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Message no.	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Fig. 6-97: Structure of variables for status messages with 2 bytes

A bit set in a byte activates the corresponding status message in the operating device.

6.8.8.4 Variable for Acknowledging Messages

The variable for acknowledging messages has the same structure as the variable for the messages themselves.

Each bit set in a byte represents the acknowledgment of the corresponding message.

6.8.9 Memory Requirement for Messages and Data Sets

Depending on the operating device, different storage quantities are available, which are also used differently for messages and data sets.

	Z80-CPU	32 Bit RISC CPU
Available Memory Space in Bytes	About 116000	About 116000 for FW version 1.07 or lower About 147000 for FW version 1.08 or higher
Memory Requirement per Message (=1 Memory Location)	24 Bytes	32 Bytes
Memory Requirement per Data Set without User Data	33 Bytes	44 Bytes

Fig. 6-98: Memories in operating devices in comparison

You must expect three times the memory requirement for the following message variants:

1. Messages containing 2 variables
2. Messages with 'Appear/Disappear'
3. Messages with acknowledgment ('Appear/Disappear' active).
4. Messages whose first variable has a size of greater than 4 bytes

The following table illustrates the memory use for 500 messages and a maximum number of data sets containing 22 bytes of user data:

	Z80-CPU	32 Bit RISC CPU
Available Memory Space in Bytes	About 116000	About 116000 for FW version 1.07 or lower About 147000 for FW version 1.08 or higher
Memory Requirement for Message in Bytes	24 x 500 = 1200	32 x 500 = 16000
Remaining Memory Capacity in Bytes	104000	About 100000 for FW version 1.07 or lower About 1294000 for FW version 1.08 or higher
Possible Number of Data Sets	104000 / (22+33) = 1890	100000 / (22+44) = 1515 for FW version 1.07 or lower 1294000 / (22+44) = 1960 for FW version 1.08 or higher

Fig. 6-99: Memory use for 500 messages in comparison

You can use the following formula to determine the exact message buffer size:

G >= M1 + M3 + 20

G = Size of message buffer

B = Size of parallel message system in bytes

M1 = Number of messages which: require only 1 memory location, are entered in the message editor and have message numbers smaller than B x 8

M3 = Number of messages which: require 3 memory locations, are entered in the message editor and have message numbers smaller than B x 8

20 = Minimum size of message buffer

6.9 Working with Password Protection

Password protection prevents screens from being accessed and the data they contain from being altered without proper authorization. The protective function is available in every operating device. It is achieved by assigning access levels to screens and by using passwords.

Unless otherwise specified by the programmer, the access levels for all screens automatically default to the lowest level (=0). That means, no password is required to access screens with this access level.

Two authorization levels, referred to as the **edit level** and **view level**, are assigned to every password.

View level

View level means that the next screen can be viewed after the password is entered; but the values in it can not be edited.

Edit level

Edit level means that the screen can be viewed after the password is entered and the values in it can be edited.

The following rules apply to passwords:

- Access is permitted if the view level and edit level values are greater than or equal to the values specified for the access level.
- The edit level must be equal or less than the view level.
- The higher the values for the view level and edit level, the higher the degree of authorization.
- The valid range of values for the view level and edit level is 0 to 255.
- The default setting for both is 0.
- The authorization levels are automatically set to 0 if you enter an incorrect password.
- If you select the Data Release key for an edit level that is too low, no function is implemented when you select the key.

You can enter a password in all screens. The only special case is the setup screen. The system variable **ScrchgPasswd** is available for entry.

See chapter “ScrchgPasswd“ on page 6-196.

In the programming software, you can select the Password Editor, which allows hidden password entry on the operating device. An X then appears for each character you enter in the operating device.

Master Password

During programming, we advise you to ensure that at least one password, a master password, has the highest authorization level. The first password entered in the programming system is of particular significance as a master password. Unlike all other passwords, the master password can not be changed in the operating device. It also allows you to reset all changed passwords to the standard values entered in the programming software.

Example for using access levels:

Access level for screen 5 = 10

Access level for screen 6 = 20

Access level for screen 7 = 30

Password 4712 has the edit level = 15 and the view level = 25

The following accesses are possible after the password **4712** has been entered:

- Screen 5 will be displayed, editing of values is authorized.
- Screen 6 will be displayed, editing of values is **not** authorized.
- Screen 7 will **not** be displayed, editing of values is **not** authorized.

Startup screen

The access level for the startup screen is always 0.

Setup screen

The setup screen is an exception with regards to the password and external data release functions. Since no communication is taking place when the setup screen is displayed, the external data release function is not applicable.

To restrict access, passwords must be used!

By defining the die system variable **ScrchgPasswd** as the first editable variable in the setup screen, all further variables can be protected against unauthorized access. The view level does not apply when accessing the setup screen. Viewing is always permitted if a value less than or equal to 254 is selected for the access level of the setup screen.

The edit level for all variables of the setup screen, with the exception of the system variable **ScrchgPasswd**, is the same as that defined as the access level.

Access to the screen is always denied if an access level of 255 is defined for the setup screen. This means that the setup screen will no longer be displayed during initialization of the operating device and can therefore not be selected. However, all device-specific parameters can also be edited in any other screen. The new parameters become effective by restarting the operating device or with the system variable **Boot**.

6.9.1 Reactivate Password Protection

The access authorization for a screen or variable is reset when the following is carried out:

- The operating device is switched off and back on again.
- The wrong password is entered.
- A logical 1 is written to the PR bit of the Write coordination byte.
- The system variable **ScrchgResPasswd** is activated.
- The option **Reset Password** is selected in the screen parameters of the password-protected screen.

See chapter “ScrchgResPasswd“ on page 6-197.

6.9.2 Password Screen and Password Functions

You can create a password query screen. This screen will then appear when you try to go to a password-protected screen, and you have not already entered a password with sufficient authorization.

As soon as you enter a password with sufficient authorization in the password query screen, and select the Enter key, the system opens the screen previously selected. No restrictions apply to the other content (for example, texts, other variables, and softkeys) in the screen.

For each screen of the user interface, you can specify whether password protection will be activated after you exit the screen.

If the operator has not entered a valid password, it must be possible to exit the screen. You can program the **cursor key Home** to do this, for example.

If you do not create a password query screen, the operator must enter a password in screens specifically provided for this purpose.

You can deactivate password protection entirely by writing the value 1 to the system variable **PasswdInactive**.

The operating device then behaves as if each screen were created with an edit and view level of 0. The system variable is battery-backed, that is, the deactivation still has an impact after you switch on the operating device again.

See chapter “PasswdInactive“ on page 6-198.

6.10 Working with System Variables

You can use system variables to control the operating device’s internal functions.

You can display and change the value of system variables either in a screen or using any suitable representation type, function or softkey.

When you link a system variable with a function key or softkey, the following rules apply:

- Do not use the same key to link a screen change and a system variable.
- You do not have to link Set (1) and Reset (0) with the same key, except if you are setting up a jogging mode.

Do not add the names of system variables to the variable list! In the same way as you use the name of a system variable for a controller variable, the function is lost for the operating device.

6.10.1 Basic Functions

- IntEraseEprom (Delete Flash memory)
- MainVersion (Display firmware version)
- ComVersion (Display protocol version)
- UserVersion (Display project version)
- Boot (Trigger reboot)
- LcdContrast (Contrast setting display)
- LcdBackground (Normal/inverse display)
- LcdBackLight (Dim background lighting)
- TurnOnTemp (Switch LCD display on if a specific temperature is reached)
- OsLanguage (Language switching)
- IdentName (Project ID - name)
- IdentVersion (Project ID - version number)
- IdentDate (Project ID - date)
- IdentTime (Project ID - time)
- IdentCount (Project ID - counter)
- IdentRandom (Project ID - random number)
- ComErrorRetry (Number of communication errors)
- SerialNumber (Serial number)
- IPAddress (IP address of the operating device)
- BaseScreenDelay (Delay for base screen)
- BaseScreenActivate (Activate base screen function)

6.10.1.1 IntEraseEprom

Function		Deletes the project from the Flash memory and places the operating device into the download mode.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Deletes the project

6.10.1.2 MainVersion

Function		Displays the current firmware version.
Data type		Alphanumeric
Representation type		Alphanumeric, field length = 8
Configurable values		Format determined by the manufacturer.

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.1.3 ComVersion

Function		Displays the type and version number of the current protocol.
Data type		Alphanumeric
Representation type		Alphanumeric, field length = 8
Configurable values		Format determined by the manufacturer.

6.10.1.4 UserVersion

Function		Displays the project's version number.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		0 to 255

6.10.1.5 Boot

Function		Boots the operating device (system restart).
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Boot

6.10.1.6 LcdContrast

Function		Sets the contrast of LC displays.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		Depends on the operating device type.

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

☞ Adhere to the values for the upper and lower limits, as specified in the user manual for the relevant operating device.

6.10.1.7 LcdBackground

Function		Displays screens in inverted format on operating devices equipped with a LC display.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Not inverted
	1	Inverted

6.10.1.8 LcdBackLight

Function		Brightness of the backlighting of LC displays.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		Depends on the operating device type.

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

☞ Adhere to the values for the upper and lower limits, as specified in the user manual for the relevant operating device.

6.10.1.9 TurnOnTemp

Function		Temperature value at which the display is automatically switched on.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Display OFF
	1	Display ON

6.10.1.10 OsLanguage

Function		For multilingual projects, this variable is used for online language selection.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	First language
	N	n–th language

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.1.11 IdentName

Function		Displays the name of the current project (application ID).
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 13 characters

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.1.12 IdentVersion

Function		Displays the version of the current project (application ID).
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 5 characters

6.10.1.13 IdentDate

Function		Displays the date of the current project (application ID).
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 6 characters

6.10.1.14 IdentTime

Function		Displays the time of the current project (application ID).
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 6 characters

6.10.1.15 IdentCount

Function		Displays the counter value of the current project (application ID).
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 4 characters

6.10.1.16 IdentRandom

Function		Displays the current project's ending (application ID).
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 2 characters

6.10.1.17 ComErrorRetry

Function		Displays the number of communication errors.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to n	Number of communication errors

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.1.18 SerialNumber

Function		Displays the serial number.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Max. 64 characters

6.10.1.19 IPAddress

Function		Displays the toplical IP address of the operating device.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		0.0.0.0 to 255.255.255.255

6.10.1.20 BaseScreenDelay

Function		Delay until base screen (target screen) is displayed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	10 to 9999	Seconds

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.1.21 BaseScreenActivate

Function		Activates/deactivates the "time-controlled change to base screen" function.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Time-controlled screen change deactivated
	1	Time-controlled screen change activated

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.2 Communication SER1

- ComDataLenA (Number of data bits)
- ComParityA (Parity setting)
- ComStopBitsA (Number of stop bits)
- ComBaudrateA (Baud rate setting)
- ComHandshakeA (Handshake setting)
- ComDefaultA (Submit parameters)
- ComTimeout (Timeout setting)
- ComRetryTimeout (Retry time setting)
- ComSlaveNr (Network slave number)
- ComErrorCode (Memory for last error code)
- ComErrorSubcode (Memory for last error subcode)

6.10.2.1 ComDataLenA

Function		Sets the number of data bits for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	5 Bit
	1	6 Bit
	2	7 Bit
	3	8 Bit

6.10.2.2 ComParityA

Function		Sets the parity for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	No parity
	1	Odd parity
	2	Even parity

6.10.2.3 ComStopBitsA

Function		Sets the number of stop bits for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	1 Bit
	1	1,5 Bit
	2	2 Bit

6.10.2.4 ComBaudrateA

Function		Sets the baud rate for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	300 Baud
	1	600 Baud
	2	1200 Baud
	3	2400 Baud
	4	4800 Baud
	5	9600 Baud
	6	19200 Baud
	7	38400 Baud
	8	57600 Baud (operating devices with 386 CPU only)

6.10.2.5 ComHandshakeA

Function		Sets the handshake method for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	No handshake
	1	Hardware handshake (RTS/CTS)
	2	Software handshake (XON/XOFF)

6.10.2.6 ComDefaultA

Function		Activates the interface parameters for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Activates the interface parameters entered by the operator.
	2	Activates the interface parameters that were specified in the programming software.

6.10.2.7 ComTimeout

Function		Sets the timeout watchdog time for SER1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	No timeout watchdog (Initial state)
	1 to 65535	Timeout watchdog time in ms

6.10.2.8 ComRetryTimeout

Function		Sets the waiting time (delay) after which another connection setup is attempted for SER1. This time period allows to span the time period required for the PLC-specific power-up phase, thereby preventing error messages from being generated.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 65535	Waiting time in ms

6.10.2.9 ComSlaveNr

Function		Sets the slave number for an operating device connected to a network.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 255	Slave number

6.10.2.10 ComErrorCode

Function		Displays the last error code issued for a COMMUNICATION, SYSTEM, or FATAL error.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 99999	

☞ By inserting this system variable into the message text of a message, the error code will be stored in the message memory in addition to the message.

☞ See chapter “Serial Message System“ on page 6-147.

6.10.2.11 ComErrorSubcode

Function		Displays the last error subcode (low word) issued for a COMMUNICATION, SYSTEM, or FATAL error.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 99999	

☞ By inserting this system variable into the message text of a message, the error code will be stored in the message memory in addition to the message.

☞ See chapter “Serial Message System“ on page 6-147.

6.10.3 Error Statistics SER1

- ComParityCount (Parity error counter)
- ComOverrunCount (Overrun error counter)
- ComFrameCount (Protocol frame error counter)

6.10.3.1 ComParityCount

Function		Displays the number of parity errors for SER1. Is deleted at every download.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 65535	Number of parity errors

6.10.3.2 ComOverrunCount

Function		Displays the number of overrun errors.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 65535	Number of overrun errors

6.10.3.3 ComFrameCount

Function		Displays the number of framing errors.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 65535	Number of framing errors

6.10.4 Communication SER2

- ComDataLenB (Number of data bits)
- ComParityB (Parity setting)
- ComStopBitsB (Number of stop bits)
- ComBaudrateB (Baud rate setting)

- ComHandshakeB (Handshake setting)
- ComDefaultB (Enter parameters)

6.10.4.1 ComDataLenB

Function		Sets the number of data bits for SER2.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	5 Bit
	1	6 Bit
	2	7 Bit
	3	8 Bit

6.10.4.2 ComParityB

Function		Sets the parity for SER2.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	No parity
	1	Odd parity
	2	Even parity

6.10.4.3 ComStopBitsB

Function		Sets the number of stop bits for SER2.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	1 Bit
	1	1,5 Bit
	2	2 Bit

6.10.4.4 ComBaudrateB

Function		Sets the baud rate for SER2.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	300 Baud
	1	600 Baud
	2	1200 Baud
	3	2400 Baud
	4	4800 Baud
	5	9600 Baud
	6	19200 Baud
	7	38400 Baud
	8	57600 Baud (operating devices with 386 CPU only)

6.10.4.5 ComHandshakeB

Function		Sets the handshake for SER2.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	No handshake
	1	Hardware handshake (RTS/CTS)
	2	Software handshake (XON/XOFF)

6.10.4.6 ComDefaultB

Function		Activates the interface parameters for SER2.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Activates the interface parameters entered by the operator.
	2	Activates the interface parameters that were specified in the programming software.

6.10.5 Real-Time Clock

- RtcSec (Display/set seconds)
- RtcMin (Display/set minutes)
- RtcHour (Display/set hours)
- RtcDay (Display/set day)
- RtcMonth (Display/set month)
- RtcYear (Display/set year)
- RtcDayOfWeek (Display/set day of the week)
- RtcDateFmt (Display/set date format)
- RtcYear2000 (Display year with 4-digits)

☞ The values for the real-time clock can be set from the operating device and from the controller.

☞ See chapter “Date and Time Image“ on page 6-233.

6.10.5.1 RtcSec

Function		Sets the seconds of the real-time clock.
Data type		Numeric
Representation type		Decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number, bar
Configurable values	0 to 59	Seconds

6.10.5.2 RtcMin

Function		Sets the minutes of the real-time clock.
Data type		Numeric
Representation type		Decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number, bar
Configurable values	0 to 59	Minutes

6.10.5.3 RtcHour

Function		Sets the hours of the real-time clock.
Data type		Numeric
Representation type		Decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number, bar
Configurable values	0 to 23	Hours

6.10.5.4 RtcDay

Function		Sets the day of the date for the real-time clock.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 31	Number of days depends on the month. Invalid settings are corrected by the real-time clock next time when the date changes.

6.10.5.5 RtcMonth

Function		Sets the month of the real-time clock.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	1 to 12	

6.10.5.6 RtcYear

Function		Sets the year of the real-time clock.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 99	Only the year and decade are influenced.

6.10.5.7 RtcDayOfWeek

Function		Sets the day of the week of the real-time clock.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 6	Only for operating devices with TMP Z84 CPU or RISC–CPU
	1 to 7	Only for operating devices with 386 CPU

6.10.5.8 RtcDateFmt

Function		Sets the date format for the message output.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Europe DD MM YY
	1	USA MM DD YY
	2	Japan YY MM DD

6.10.5.9 RtcYear2000

Function		Sets a 4-digit year of the real-time clock.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 9999	

6.10.6 Serial Message System

- RepmanSortCrit (Set sorting criterion)
- ClearRepBuf (Clear message buffer)
- RepmanRepPrint (Output to a printer)
- RepoutNr (Output message number)
- RepoutDate (Output message date)
- RepoutTime (Output message time)
- RepoutAnzYear (Display year)
- RepoutRepText (Output message)
- RepoutRepText21 (Output message starting from 21st digit)
- RepoutRepText41 (Output message starting from 41st digit)
- RepoutRepText61 (Output message starting from 61st digit)
- RepmanQuitKey (Acknowledge message)
- RepmanChgMask (Screen change from message field)
- RepoutQuitText (Output current message acknowledgment)
- RepoutQuitText21 (Output message acknowledgment starting from 21st digit)
- RepoutQuitText41 (Output message acknowledgment starting from 41st digit)
- RepoutQuitText61 (Output message acknowledgment starting from 61st digit)
- RepoutQuitAnz (Number of unacknowledged messages)
- RepoutMarker (Position of message field)
- RepoutSelectGroup (Output messages of group only)
- RepoutSelectTime (Output messages in chronological order)
- RepoutGroup (Output message group)

6.10.6.1 RepmanSortCrit

Function		Defines the sorting criterion for message output.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	By priority of message number
	1	In order of arrival (most recent first)
	2	In order of arrival (oldest first)
	3	By group

6.10.6.2 ClearRepBuf

Function		Erases the memory for the serial messages.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Delete all messages from the message memory
	2	Delete only the acknowledged messages from the message memory

6.10.6.3 RepmanRepPrint

Function		Is used to have messages output to a printer.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Formatted output, the interface is used permanently.
	2	Complete output, the interface is used permanently.
	3	Formatted output, the interface is used temporarily.
	4	Complete output, the interface is used temporarily.

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

If you use the interface permanently for message output, it will not be possible to send any other print jobs to the printer.

If you use the interface temporarily for message output, messages will not be printed while other print jobs are being printed.

6.10.6.4 RepoutNr

Function		Allows you to output a message number along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.6.5 RepoutDate

Function		Allows you to output the date along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.6.6 RepoutTime

Function		Allows you to output the time along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.6.7 RepoutAnzYear

Function		Specifies how the date appears when the message is output.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Year with 2 digits
	1	Year with 4 digits

6.10.6.8 RepoutRepText

Function		Displays the most recent serial message. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.9 Repout RepText21

Function		Displays the most recent serial message beginning from the 21st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.10 RepoutRepText41

Function		Displays the most recent serial message beginning from the 41st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.11 RepoutRepText61

Function		Displays the most recent serial message beginning from the 61st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.12 RepmanQuitKey

Function		Simulates the function of the Acknowledge key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Function of the Acknowledge key active

You must edit the value of this variable with a function key or a button. You can not use any other input form.

6.10.6.13 RepmanChgScreen

Function		Lets you jump to the screen which is linked with the selected message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive

You must edit the value of this variable with a function key or a button. You can not use any other input form.

6.10.6.14 RepoutQuitText

Function		Displays the most recent unacknowledged serial message. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

If the operator acknowledges the displayed message, the system automatically displays the next unacknowledged message.

6.10.6.15 RepoutQuitText21

Function		Displays the most recent unacknowledged serial message beginning from the 21st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.16 RepoutQuitText41

Function		Displays the most recent unacknowledged serial message beginning from the 41st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.17 RepoutQuitText61

Function		Displays the most recent unacknowledged serial message beginning from the 61st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.6.18 RepoutQuitAnz

Function		Displays the number of messages that still need to be acknowledged.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		

6.10.6.19 RepoutMarker

Function		Indicates the current position of the messages within the message box.
Data type		Numeric
Representation type		Decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number, bar
Configurable values	0	The message with the highest priority is visible.
	1	Neither the message with the highest priority nor the message with the lowest priority is visible.
	2	The message with the lowest priority is visible.
	3	The message with the highest priority and the message with the lowest priority is visible.
	4	No message is visible within the message box.

6.10.6.20 RepoutSelectGroup

Function		Sets the group numbers whose messages are displayed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0x01h to 0x80h	Group number 1 to 8

Each group is represented by one bit of a byte. A logical '1' in bit 0 activates group1, in bit 1 activates group2 and so on.

The settings for a message field override the settings for this system variable! To prevent the settings in the operating device from being overridden, you must activate the Global Settings function for the message field.

6.10.6.21 RepoutSelectTime

Function		Specifies the time rule according to which messages are displayed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	1	All messages in chronological order.
	2	All acknowledged messages that do not have the attribute "Disappear".
	3	All messages that do not have the attribute "Acknowledged".

6.10.6.22 RepoutGroup

Function		Allows you to output a group number along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.7 Parallel Message System

- RepmanSortCritP (Set sorting criterion)
- RepoutNrP (Output message number)
- RepoutDate P (Output message date)
- RepoutTimeP (Output message time)
- RepoutAnzYearP (Display year)
- RepoutRepTextP (Output message)
- RepoutRepText21P (Output message starting from 21st digit)
- RepoutRepText41P (Output message starting from 41st digit)
- RepoutRepText61P (Output message starting from 61st digit)
- RepoutSelectGroupP (Output messages of group only)
- RepoutGroupP (Output message group)

6.10.7.1 RepmanSortCritP

Function		Defines the sorting criterion for message output.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	By priority of message number
	1	In order of arrival (most recent first)
	2	In order of arrival (oldest first)

The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.7.2 RepoutNrP

Function		Allows you to output a message number along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.7.3 RepoutDateP

Function		Allows you to output the date along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.7.4 RepoutTimeP

Function		Allows you to output the time along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.7.5 RepoutAnzYearP

Function		Specifies how the date appears when the message is output.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Year with 2 digits
	1	Year with 4 digits

6.10.7.6 RepoutRepTextP

Function		Displays the most recent parallel message. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.7.7 Repout RepText21P

Function		Displays the most recent parallel message beginning from the 21st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.7.8 RepoutRepText41P

Function		Displays the most recent parallel message beginning from the 41st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.7.9 RepoutRepText61P

Function		Displays the most recent parallel message beginning from the 61st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.7.10 RepoutSelectGroupP

Function		Sets the group numbers whose messages are displayed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 256	

Each group is represented by one bit of a byte. A logical '1' in bit 0 activates group1, in bit 1 activates group2 and so on.

The settings for a message field override the settings for this system variable! To prevent the settings in the operating device from being overridden, you must activate the Global Settings function for the message field.

6.10.7.11 RepoutGroupP

Function		Allows you to output a group number along with the message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.8 Printer Control

- StopPrint (Cancel print process)
- BlockPrint (Print selected block)
- PrintAllRep (Print serial messages)
- PrintAllState (Print parallel messages)
- BlockPrintLong (Print selected block in full length)

6.10.8.1 StopPrint

Function		Stops the current print process.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Stops the print process.

6.10.8.2 BlockPrint

Function		Starts to print the selected messages.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Starts the print process.

6.10.8.3 PrintAllRep

Function		Starts to print all of the serial messages.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Starts printing using the current formatting.
	2	Starts printing using all of the formatting options.

6.10.8.4 PrintAllState

Function		Starts to print all parallel messages.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Starts the print process.

6.10.8.5 BlockPrintLong

Function		Starts printing the selected messages using all of the formatting options.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Starts the print process.

6.10.9 Menu Control / Keys

- NewMask (Screen change)
- VarTablenR0 (Table numbering, beginning with 0)
- VarTablenR1 (Table numbering, beginning with 1)
- TabLeft (Moves to next column on the left)
- TabRight (Moves to next column on the right)
- TabPgUp (Page down)
- TabPgDn (Page up)
- Shift (Shift mode 1 for alphanumeric editor)
- ShiftCase (Shift mode 2 for alphanumeric editor)
- ShiftTouch (Display shift mode for touch screen panels)
- KeyCursLeft (Function of Cursor Left key)
- KeyCursRight (Function of Cursor Right key)
- KeyCursUp (Function of Cursor Up key)
- KeyCursDown (Function of Cursor Down key)
- KeyHome (Function of Cursor Home key)
- KeyHelp (Function of Help key)
- KeyDot (Function of the Dot key)
- KeyClear (Function of Delete key)
- Key0 (Function of key 0)
- Key1 (Function of key 1)
- Key2 (Function of key 2)
- Key3 (Function of key 3)
- Key4 (Function of key 4)
- Key5 (Function of key 5)
- Key6 (Function of key 6)
- Key7 (Function of key 7)
- Key8 (Function of key 8)
- Key9 (Function of key 9)
- KeyPlus (Function of the Plus (+) key)
- KeyMinus (Function of the Minus (-) key)
- KeyEnter (Function of the Enter key)
- KeyEdit (Function of the Data Release (Edit) key)

6.10.9.1 NewMask

Function		Changes to the screen with the indicated number.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	1 to 9999	Screen number

6.10.9.2 VarTablenR0

Function		Creates a continuous numbering in tables, beginning with 0.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to n	

6.10.9.3 VarTablenR1

Function		Creates a continuous numbering in tables, beginning with 1.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	1 to n	

6.10.9.4 TabLeft

Function		Is used to move to the left column of a table.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Move to left column

6.10.9.5 TabRight

Function		Is used to move to the right column of a table.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Move to right column

6.10.9.6 TabPgUp

Function		Is used to page up within a table.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Page up

6.10.9.7 TabPgDn

Function		Is used to page down within a table.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Page down

6.10.9.8 Shift

Function		Enables alphanumerical character input.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state, only numerical input possible
	1	Upper-case alphanumerical character input enabled
Key		Letters (Characters)
0		( ) ° 0
1		S T U 1
2		V W X 2
3		Y Z % 3
4		J K L 4
5		M N O 5
6		P Q R 6
7		A B C 7
8		D E F 8
9		G H I 9
Decimal point		: ? ! .
Plus		< = > +
Minus		\ * / –

6.10.9.9 ShiftCase

Function		Enables alphanumerical character input.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state, only numerical input possible
	1	Upper case and lower case alphanumerical character input enabled
Key		Letters (Characters)
0		( ) ° 0
1		S T U s t u 1
2		V W X v w x 2
3		Y Z % y x % 3
4		J K L j k l 4
5		M N O m n o 5
6		P Q R p q r 6
7		A B C a b c 7
8		D E F d e f 8
9		G H I g h i 9
Decimal point		: ? ! .
Plus		< = > +
Minus		\ * / –

6.10.9.10 ShiftTouch

Function		Displays the state of the shift mode.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	OFF
	1	ON

6.10.9.11 KeyCursLeft

Function		Simulates the key function of the Cursor Left key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Cursor Left active

6.10.9.12 KeyCursRight

Function		Simulates the key function of the Cursor Right key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Cursor Right active

6.10.9.13 KeyCursUp

Function		Simulates the key function of the Cursor Up key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Cursor Up active

6.10.9.14 KeyCursDown

Function		Simulates the key function of the Cursor Down key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Cursor Down active

6.10.9.15 KeyHome

Function		Simulates the key function of the Cursor Home key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Cursor Home active

6.10.9.16 KeyHelp

Function		Simulates the key function of the Help key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Help active

6.10.9.17 KeyDot

Function		Simulates the key function of the Decimal Point key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Decimal Point active

6.10.9.18 KeyClear

Function		Simulates the key function of the Clear key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Clear active

6.10.9.19 Key0

Function		Simulates the key function of the key '0'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 0 active

6.10.9.20 Key1

Function		Simulates the key function of the key '1'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 1 active

6.10.9.21 Key2

Function		Simulates the key function of the key '2'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 2 active

6.10.9.22 Key3

Function		Simulates the key function of the key '3'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 3 active

6.10.9.23 Key4

Function		Simulates the key function of the key '4'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 4 active

6.10.9.24 Key5

Function		Simulates the key function of the key '5'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 5 active

6.10.9.25 Key6

Function		Simulates the key function of the key '6'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 6 active

6.10.9.26 Key7

Function		Simulates the key function of the key '7'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 7 active

6.10.9.27 Key8

Function		Simulates the key function of the key '8'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 8 active

6.10.9.28 Key9

Function		Simulates the key function of the key '9'.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function 9 active

6.10.9.29 KeyPlus

Function		Simulates the key function of the Plus key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Plus active

6.10.9.30 KeyMinus

Function		Simulates the key function of the Minus key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Minus active

6.10.9.31 KeyEnter

Function		Simulates the key function of the Enter key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Enter active

6.10.9.32 KeyEdit

Function		Simulates the key function of the Data Release key.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Key function Edit active

6.10.10 Password

- ScrchgPasswd (Input password)
- ScrchgResPasswd (Delete password, reset authorization)
- ChangePasswd (Change password)
- FlashPasswd (Reset passwords)
- PasswdInactive (Disable password protection)
- ActViewLevel (Topical view level)
- ActEditLevel (Topical edit level)

6.10.10.1 ScrchgPasswd

Function		Variable for password input.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		11 characters

6.10.10.2 ScrchgResPasswd

Function		Deletes the currently entered passwordand resets the access authorization.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Delete password and reset access authorization

6.10.10.3 ChangePasswd

Function		Changes a password.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		11 characters

6.10.10.4 FlashPasswd

Function		Resets the passwords to the values specified in the programming software.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Reset passwords

Before you use this system variable, make sure to save the password with the highest-level access authorizations!

6.10.10.5 PasswdInactive

Function		Deactivates password protection.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Password protection active, inactive during initial initialization
	1	Password protection active, edit and view level = 255

The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.10.6 ActViewLevel

Function		Displays the current view level.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 255	

6.10.10.7 ActEditLevel

Function		Displays the current edit level.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 255	

6.10.11 Recipes

- SelectDSNr (Active data set number)
- SelectDSName (Active data set name, variant 1)
- DestDSNr (Target data set number)
- DSCopy (Copy data set)
- DSDelete (Delete data set)
- ActDSName (Active data set name, variant 2)
- SelectRezeptNr (Active recipe number)
- SelectRezeptName (Active recipe name)
- StateDSDelete (Status of delete process)
- LoadRezName (Name of last recipe loaded)
- DSDownload (Send data set to controller)
- DSDnloadBreak (Stop data set transfer)
- DSDnloadState (Monitor data set transfer to controller)
- LoadDSName (Name of last data set transferred)
- LoadDSNr (Number of last data set transferred)
- StartSave (Send data set to PC)
- SaveState (Monitor data set transfer to PC)
- StartRestore (Send data set from PC to terminal)
- RestoreState (Monitor data set transfer to terminal)
- RestoreLineNr (Line number of data set file, to terminal)
- StartRezPrint (Print data set)
- RezPrintState (Monitor data set print process)
- StartUpload (Read data set from controller)
- UploadDSNr (Number of destination data set)
- UploadState (Monitor data set read process)

6.10.11.1 SelectDSNr

Function		Number of the current data set.
Data type		Numeric
Representation type		Selection text, decimal number
Configurable values	0 to 250	

6.10.11.2 SelectDSName

Function		Name of the current data set.
Data type		Alphanumeric
Representation type		Selection Text
Configurable values		30 characters

6.10.11.3 DestDSNr

Function		Number of the destination data set.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	1 to 250	

6.10.11.4 DSCopy

Function		Copies the current data set to the destination indicated in DestDSNr.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Copy to destination in DestDSNr
	2	Automatically copy and search a free data set
	3	Copy to destination in DestDSNr and overwrite any data set existing

6.10.11.5 DSDelete

Function		Deletes the current data set.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Deletes the active data set and activates the first data set of the recipe.
	2	Deletes all data sets of the current recipe and activates the default data set of the recipe.

6.10.11.6 DSNew

Function		Generates a new data set with the name specified in the system variable. **DSDestNr contains no value:** Data set is assigned to the next free number. **DSDestNr contains a value:** Data set number is checked - if already assigned and not write-protected the data set is overwritten - if already assigned and write-protected the data set is assigned to the next free number.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.11.7 ActDSName

Function		Name of the current data set.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		30 characters

6.10.11.8 SelectRezeptNr

Function		Number of the currently active recipe.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		

☞ You can only enter the numbers of existing recipes. Invalid entries are ignored

6.10.11.9 SelectRezeptName

Function		Name of the current recipe.
Data type		Numeric
Representation type		Alphanumeric
Configurable values		The programming software provides the texts automatically (30 characters).

☞ You can only select the names of existing recipes. The programming software automatically generates a text list with the names of existing recipes and links it to this system variable.

6.10.11.10 StateDSDelete

Function		Displays the status of the data set delete process.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Delete inactive
	1	Delete active (the current data set of the current recipe is deleted)

6.10.11.11 LoadRezName

Function		Name of the last recipe transferred.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Up to 30 characters If the recipe was deleted after being transferred, a number of question marks '????' are displayed instead of the name.

6.10.11.12 DSDownload

Function		Loads the current data set to the controller.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Loads the content of the recipe buffer to the controller.
	2	Loads the content of the single variable to the controller.

6.10.11.13 DSDnloadBreak

Function		Ends the data set transfer currently in progress.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Stop data set transfer

6.10.11.14 DSDnloadState

Function		Displays the status of the data set transfer to the controller.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Data set transfer is requested, but not yet released by the controller.
	2	Data set transfer in progress

6.10.11.15 LoadDSName

Function		Name of the last data set transferred.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Up to 30 characters If the data set was deleted after being transferred, a number of question marks '????' are displayed instead of the name.

6.10.11.16 LoadDSNr

Function		Number of the last data set transferred.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		If the data set was deleted after being transferred, the number 0 (zero) is shown.

6.10.11.17 StartSave

Function		Loads data sets to the PC.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Transfers a single data set to the PC
	2	Transfers all data sets of a recipe to the PC
	3	Transfers all data sets to the PC
	4	Transfers all data sets to the CompactFlash card.

6.10.11.18 SaveState

Function		Displays the status of the data set transfer to the PC.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Transfers a single data set.
	2	Transfers all data sets of a recipe.
	3	Transfers all data sets in the operating device

6.10.11.19 StartRestore

Function		Controls the process of loading recipes and data sets to the operating device.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Activate ready-to-receive
	2	Stop transfer
	3	Transfer of all recipes and data sets in the **tesrez.bak** file located on the CompactFlash card.

6.10.11.20 RestoreState

Function		Displays the status of the data transfer to the operating device.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Data transfer in progress

6.10.11.21 RestoreLineNr

Function		Current line number in the data set file.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		1 to 255

6.10.11.22 StartRezPrint

Function		Starts printing a data set.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Start printing
	2	Stop printing

6.10.11.23 RezPrintState

Function		Displays the status of the data set print process.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Printing in progress

6.10.11.24 StartUpload

Function		Loads the data set which is currently active in the controller to the operating device.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Loads one variable at a time to the destination indicated in UploadDSNr.
	2	Loads variables as a block from the recipe buffer to the destination indicated in UploadDestNr.
	3	Loads one variable at a time and saves them automatically to a free data set. Terminal message 18 is displayed if no free data set is available.
	4	Loads variable as a block from the recipe buffer and saves them automatically to a free data set. Terminal message 18 is displayed if no free data set is available.

6.10.11.25 UploadDSNr

Function		Number of the destination data set for the upload.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	1 to 250	

6.10.11.26 UploadState

Function		Displays the status of the data set upload.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Upload active

6.10.12 Running Time Meters

- Counter1 to Counter8 (Status of running time meter)

6.10.12.1 Counter1 to Counter8

Function		Running time meter 1 to 8. The counter is incremented when the bit is set.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		0 to 4.294.967.295

☞ The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

☞ The function of the Running Time Meter is dependent on other parameters.

☞ See chapter “Working with Running Time Meters“ on page 6-225.

6.10.13 Loop-through Operation

- Pg2Sps (Enable loop-through operation)
- Pg2SpsState (Status of loop-through operation)

6.10.13.1 Pg2Sps

Function		Enables/disables the loop-through operation. **The loop-through operation is alternately activated/deactivated by the rising edge from 0 to 1!**
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Enable loop-through operation

☞ Provisions must be made to be able to use the loop-through operation with the PG protocol !

6.10.13.2 Pg2SpsState

Function		Displays the status of the upload process of the loop-through operation.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Request loop-through operation
	2	Loop-through operation is possible
	3	Loop-through operation active

6.10.14 Loadable Character Set

- ChrsetName (Current character set name)

6.10.14.1 ChrsetName

Function		Displays the name of the current character set.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		Default (character set NORMAL or ZOOM used for display)
		Character set name (user-created character sets used for display)

6.10.15 Maintenance (Service)

- User1 to User5 (Universal for users)
- LCDADCInput (Read AD converter)
- LCDDACOutput (Read DA converter)
- Break (Cancels the editor)
- StartCalibrationTouch (Calibrate touch sensor)
- StateCalibrationTouch (Display calibration status)
- MaskStartupTime (Screen buildup time)
- ScreenStartupComCnt (Number of communication requests)
- ComMeanTime (Transfer time)
- KeyResponseTime (Alteration time of variable)

6.10.15.1 User1 to User5

Function		For free use.
Data type		Alphanumeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		Any, up to 16 bit

The value of the variable is stored retentively. The stored value is automatically used again after a power failure.

6.10.15.2 LCDADCInput

Function		Current input value of the AD converter for contrast control.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 255	

6.10.15.3 LCDDACOutput

Function		Current input value of the DA converter for contrast control.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 255	

6.10.15.4 Break

Function		Cancels the current input process.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Cancel input process

The input values are not transferred to the controller!

6.10.15.5 StartCalibrationTouch

Function		Starts the calibration process for the touch-screen.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Inactive
	1	Starts the calibration process

Once you have set the system variable to the value 1, the next touch screen touches are used for calibration! You must set up the system variable **StateCalibrationTouch** to ensure that operators will know how to proceed.

See chapter “StateCalibrationTouch“ on page 6-211.

6.10.15.6 StateCalibrationTouch

Function		Displays the calibration status.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Touch to calibrate (the operator starts the calibration process with the next touch)
	1	Touch left pixel (the operator needs to touch a specific coordinate at the top left)
	2	Touch right pixel (the operator needs to touch a specific coordinate at the lower right)
	3	Calibration successful (calibration process complete)

For the user interface, we recommend that you create a text list or an image list, and that you present the instructions to the operator as selection text or a selection image.

6.10.15.7 MaskStartupTime

Function		Displays how much time (in milliseconds) has elapsed for screen buildup.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	---	Milliseconds

6.10.15.8 ScreenStartupComCnt

Function		Shows how many communication requests the terminal sends to the controller during screen buildup.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	---	

6.10.15.9 ComMeanTime

Function		Displays the mean transfer time (in milliseconds) for each communication request during screen buildup.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	---	Milliseconds

6.10.15.10 KeyResponseTime

Function		Shows how much time (in milliseconds) elapses to modify a variable in the controller.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	---	Milliseconds

You can use this system variable only with touch panels.

6.10.16 Editors

- EditInvers (displays editor in inverse mode)
- EditEnter (input behavior of editor)
- StatePerm (status of status LED for data release)
- EditUpperLimit (indicates the upper limit for input editor)
- EditLowerLimit (indicates the lower limit for input editor)
- ActEditNumber (indicates the variable number of the edit order)

6.10.16.1 EditInvers

Function		Displays the variable inverse while it is edited.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Not inverted
	1	Inverted

6.10.16.2 EditEnter

Function		Controls the cursor when the Enter key is pressed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Cursor changes to the next input variable
	1	Cursor remains at the current position

6.10.16.3 StatePerm

Function		Displays the status of the status-LED for the data release.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Status-LED OFF
	1	Status-LED ON
	2	Status-LED FLASHING

6.10.16.4 EditUpperLimit

Function		Indicates the upper limit of the input value for the current input variable.
Data type		Numeric
Representation type		Alphanumeric
Configurable values		

6.10.16.5 EditLowerLimit

Function		Indicates the lower limit of the input value for the current input variable.
Data type		Numeric
Representation type		Alphanumeric
Configurable values		

6.10.16.6 ActEditNumber

Function		Indicates the number of the current edit variable in the edit order.
Data type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Representation type		Alphanumeric
Configurable values		

☞ The number indicated by the system variable not always comply to the programmed number of the edit order.
Example:
Edit order is: 1, 2, 3 ActEditNumber indicates: 1, 2, 3
Edit order is: 7, 8, 9 ActEditNumber indicates: 1, 2, 3
Edit order is: 6, 8, 4 ActEditNumber indicates: 2, 3, 1

6.10.17 Help

- StateHelp (Status of status LED for help)
- SysMessage (System message number)
- SysQuitMessage (Acknowledge system messages)
- StatusText (Output message text)
- StatusText21 (Message text starting from 21st digit)
- StatusText41 (Message text starting from 41st digit)
- StatusText61 (Message text starting from 61st digit)
- ActVarLimit (Current limit value)

6.10.17.1 StateHelp

Function		Displays the status of the help status-LED.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Status-LED OFF
	1	Status-LED ON
	2	Status-LED FLASHING

6.10.17.2 SysMessage

Function		Displays the current terminal message.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1 to 29	Number of the terminal message

☞ To prevent a terminal message from being issued, you must delete the message text for the terminal message. This means, however, that the terminal message will not appear in any display forms.

6.10.17.3 SysQuitMessage

Function		Acknowledge the terminal message which is currently displayed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Status-LED OFF
	1	Status-LED ON
	2	Status-LED FLASHING

☞ For operating devices equipped with a keyboard, this function is permanently linked with the Help key. There are a number of options for acknowledging a terminal message for touch-screen-operated operating devices.

☞ See chapter “Buttons“ on page 6-40.

6.10.17.4 StatusText

Function		Displays the most recent parallel message. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.17.5 StatusText21

Function		Displays the most recent parallel message beginning from the 21st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.17.6 StatusText41

Function		Displays the most recent parallel message beginning from the 41st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.17.7 StatusText61

Function		Displays the most recent parallel message beginning from the 61st character. The message is displayed in accordance with the specified representation settings.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

6.10.17.8 ActVarLimit

Function		Displays current lower or upper limit value.
Data type		Alphanumeric
Representation type		Alphanumeric
Configurable values		

This terminal variable is by default included in the terminal messages **Value too large** and **Value too small** and displays the respective limits.

6.10.18 Print Logs

- SelectPrintLog (Select print log)
- StartPrintLog (Print print log)
- StatePrintLog (Printer state)
- PageNumber (Print log page)

6.10.18.1 SelectPrintLog

Function		Number of the currently selected print log.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 255	

6.10.18.2 StartPrintLog

Function		Starts to print the currently selected print log.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Start printing
	2	Stop printing

6.10.18.3 StatePrintLog

Function		Status of the current print process.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Ready
	1	Printing in progress
	2	Print process stopped by operator
	3	Error while printing

6.10.18.4 PageNumber

Function		Current page of the current print job. Can be combined with the representation type "Bar" to create a progress indicator.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		

6.10.19 Compact Flash Card

- CardFileName (File name display)
- CardApplicationMove (Start firmware update)
- CardFileError (Output software error messages)
- CFCardError (Output hardware error messages)

⚠ Small operator terminals of the VCP model can only use Compact Flash cards formatted as FAT16. By default, the Windows XP® operating system formats Compact Flash cards in FAT32 format! Make sure to change the settings to activate the FAT16 format when formatting a Compact Flash card using Windows XP®!

6.10.19.1 CardFileName

Function		Name of a file which you want to access in write mode or read mode. Enter the file name including the file extension. The file name can not exceed a length of 40 characters including the dot and the file extension!
Data type		Alphanumeric
Representation type		Selection Text
Configurable values		

You can display the content of the Compact Flash card using this system variable. The items for the text list are generated automatically.

6.10.19.2 CardApplicationMove

Function		Starts a firmware update from the Compact Flash card. The name of the S3 file must be entered in the system variable **CardFileName**.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Start firmware update

6.10.19.3 CardFileError

Function		Displays errors that occurred while using the Compact Flash card. The error number has different meanings depending on the type of operating device.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number

Configurable values	0	No error
	1	No Compact Flash card inserted.
	2	The specified file does not exist on the Compact Flash card or can not be read.
	3	The Compact Flash card is full or write-protected.
	4	The file already exists on the Compact Flash card.
	5	The file has the wrong data type (.S3 for application and Firmware, .TXT for data sets)
	6	The S3 file was generated for another operating device. Select a different S3 file or generate a new S3 file for the corresponding operating device.
	7	The S3 file is for an operating device equipped with a different memory size. Select a different S3 file or generate a new S3 file for the corresponding operating device.
	8	The Compact Flash card was detected.

6.10.19.4 CFCardError

Function		Displays errors that occurred while using the Compact Flash card.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	No error
	1	No Compact Flash card in device.
	2	The requested file does not exist on the Compact Flash card.
	3	The Compact Flash card is full or write-protected.
	4	The file already exists on the Compact Flash card.
	5	File has the wrong file extension.
	6	S3 file is for the wrong device type.
	7	S3 file is for the wrong memory type.
	8	The Compact Flash card was detected.

6.10.20 Set of Curves (Graphs)

- DataLogTrig
- DataLogClear

6.10.20.1 DataLogTrig

Function		Release a trigger event for a data logger.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Trigger for data logger 1
	2	Trigger for data logger 2
	3	Trigger for data logger 3
	4	Trigger for data logger 4

6.10.20.2 DataLogClear

Function		Erase the memory of a data logger
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Initial state
	1	Erase the memory of data logger 1
	2	Erase the memory of data logger 2
	3	Erase the memory of data logger 3
	4	Erase the memory of data logger 4

6.10.21 Image Parameters

- ScreenOffset (Variable for multiplex procedure)

6.10.21.1 ScreenOffset

Function		Functions as a multiplex variable. The value of this variable is transferred by the current screen. Each screen can have another screen offset value.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0 to 65535	

6.10.22 Script Processing

- SkriptId (ID number of current script)
- InstructionPointer (last value of instruction pointer)
- SkriptAktiv (turnes the script processing on/off)

6.10.22.1 SkriptId

Function		Displays the ID number of the script currently being executed.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values		

6.10.22.2 InstructionPointer

Function		Displays the last value of the instruction pointer of a virtual machine. This shows you, exactly where the script was terminated or interrupted.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	6	Script ended successfully

6.10.22.3 SkriptAktiv

Function		Turnes the script processing on or off.
Data type		Numeric
Representation type		Positive decimal number, alphanumeric, selection text, selection image, hexadecimal number, binary number
Configurable values	0	Script processing is OFF
Function	1	Script processing is ON

6.11 Working with Running Time Meters

Each operating device has 8 running time meters.

Control Byte

Each running time meter is assigned a bit in the control byte. Using the control byte, the controller can influence the running time meters in the operating device.

If bit X is set in the control byte when polling is carried out, the running time meter X is incremented. In each case, the value of the running time meter is stored in the system variable CounterX.

Bit	Counter	System Variable
0	1	Counter1
1	2	Counter2
2	3	Counter3
3	4	Counter4
4	5	Counter5
5	6	Counter6
6	7	Counter7
7	8	Counter8

Fig. 6-100: Control byte of the running time meter

Reset Byte

Each running time meter is assigned a bit in the reset byte. Using the reset byte, the controller can reset the running time meters in the operating device.

If bit X is set in the reset byte when polling is carried out, the running time meter X is reset to 0.

Bit	Counter	System Variable
0	1	Counter1
1	2	Counter2
2	3	Counter3
3	4	Counter4
4	5	Counter5
5	6	Counter6
6	7	Counter7
7	8	Counter8

Fig. 6-101: Reset byte of the running time meter

Polling Time

You use the polling time to specify the time intervals at which the operating device reads from the controller the variables for the control byte and the reset byte.

The running time meters are activated in the operating device as soon as you have entered a variable name for the control byte and specified a value for the polling time. If the polling time is 0 or if there is no address for the control byte, the Running Time Meter function in the operating device is off.

Transferring counter value

For each running time meter, you can enter a variable name in the controller. The operating device stores the value of the corresponding running time meter if the controller requests that the operating device to do so.

For this purpose, the controller writes the hexadecimal code **7FCF** into the serial message channel of the polling area.

For each variable, a 32-bit memory area must be available in the controller!

Example:

You want to set up a running time meter for a maintenance interval of 50 hours. The polling time for the counter = 60 seconds (the counter increases by one each minute).

System Variable	Counter1	
Representation type	Decimal number	
Format	Field length	4
	Fractional digits	1
	Only positive	
Scaling	Factor	1
	Divisor	6
	Addend	0

After 150 polling cycles, the operating device displays a value of 2.5 hours.150 / 6 + 0 = 25Using the format 'Fractional Digits=1', the value 25 is displayed as 2.5!

This example has a precision of +/- 6 minutes.

6.12 Working with Control Codes

You can use hexadecimal control codes to control special functions on the operating device. The control codes are transferred to the operating device using the polling area. The operating device interprets the control code and subsequently triggers the corresponding function.

The following functions can be requested by the controller:

Code	**Function**
7FC7	Delete data logger 1
7FC8	Delete data logger 2
7FC9	Delete data logger 3
7FCA	Delete data logger 4
7FCB	Trigger data logger 1
7FCC	Trigger data logger 2
7FCD	Trigger data logger 3
7FCE	Trigger data logger 4
7FCF	Write values of the running time meters to the controller
7FEx	Switch to another language (x = language number)
7FF2	Automatic data release for scanner module
7FF3	Reload event-controlled variable values
7FF4	Transfer single data set from the controller
7FF5	Delete acknowledged messages from serial message memory
7FF6	Cancel printing the print log
7FF7	Printing a print log
7FF8	Printing a data set
7FF9	Set clock in operating device
7FFA	Data set transfer from controller to operating device (block mode)
7FFB	Data set transfer from operating device to controller
7FFC	Send keyboard image to controller
7FFD	Data set transfer from controller to operating device (single mode)
7FFE	Erase serial message memory
7FFF	Refresh message system

Fig. 6-102: Control Codes

6.12.1 Delete Data Logger

You can use the following control codes from the controller, to have the operating device delete the data logger values.

Hexadecimal code: **7FC7h** deletes data logger 1

7FC8h deletes data logger 2

7FC9h deletes data logger 3

7FCAh deletes data logger 4

6.12.2 Trigger Data Logger

You can use the following control codes from the controller, to have the operating device trigger a data logger to log a new value.

Hexadecimal code: **7FCBh** triggers data logger 1

7FCCh triggers data logger 2

7FCDh triggers data logger 3

7FCEh triggers data logger 4

6.12.3 Write Values of Running Time Meters to Controller

You can use the following control code from the controller, to have the operating device pass the values of the running time meters to the controller.

Only the values of active running time meters are transferred.

Hexadecimal code: **7FCFh**

6.12.4 Switch to Another Language

You can use the following control code from the controller, to have the operating device switch to another language.

The number of the language is the least significant digit of the hexcode.

Valid hexadecimal values for the language number are **7FE0h** to **7FEFh**.

Valid decimal values for the language number in the Language Parameters dialog are 1 to 16.

Hexadecimal code: **7FExh**

Example 1: You want to load the language with the number 4.

Write the hexadecimal number 7FE3 to the address of the serial message channel.

Example 2: You want to load the language with the number 12. Write the hexadecimal number 7FEB to the address of the serial message channel.

6.12.5 Activating Recipe and Data Sets from the Controller

You can use the following control codes from the controller, to instruct the operating device to read the values for the recipe number and data set number from the controller, to make the recipe and the data set from its own memory available and to activate them.

Hexadecimal code: **7FF1h**

6.12.6 Automatic Data Release for Scanner Module

You can use the following control code from the controller, to have the operating device automatically read in the values from the connected scanner.

Hexadecimal code: **7FF2h**

6.12.7 Reload Event-Controlled Variable Values

You can use the following control code from the controller, to instruct the operating device to read all variable values from the controller again that are currently displayed in a screen and have the property Event-Controlled.

Hexadecimal code: **7FF3h**

6.12.8 Transfer Single Data Set from Operating Device to Controller

You can use the following control code from the controller, to have a single data set transferred from the operating device to the controller.

You must write the number of the data set to the variable defined for this purpose. In addition, you need to define the corresponding variables for the transfer buffers.

Hexadecimal code: **7FF4h**

6.12.9 Delete Acknowledged Messages from Serial Message Memory

You can use the following control code from the controller, to have all acknowledged messages of the operating device's serial message system erased.

In addition, the delete variable must contain the value **E216h**. This is to help avoid unintentional deletion. The delete variable is deleted in the system parameters for the serial message system.

Hexadecimal code: **7FF5h**

6.12.10 Cancel Printing the Print Log

You can use the following control code from the controller, to instruct the printer connected to the operating device to cancel the current print job for a print log.

Hexadecimal code: **7FF6h**

6.12.11 Printing a Print Log

You can use the following control code from the controller, to instruct the printer connected to the operating device to print the print log whose number was written to the variable defined for this purpose.

Hexadecimal code: **7FF7h**

The operating device will write one of the following four values back to the variable for the print log number to allow the print process to be monitored.

Value	Description
0	Print log printed with no errors.
1	Printing of the data set with the desired data set number is not possible
2	The selected print log does not exist.
3	Print process stopped.

Fig. 6-103: Return values from operating device

6.12.12 Printing a Data Set

You can use the following control code from the controller, to instruct the printer connected to the operating device to print the current data set.

Hexadecimal code: **7FF8h**

The operating device will write one of the following two hexcodes back to allow the print process to be monitored.

Value	Description
0x0h	Data set printout OK
0XFF	Printing of the data set with the desired data set number is not possible

Fig. 6-104: Return values from operating device

6.12.13 Set Clock in Operating Device

You can use the following control code from the controller, to have the operating device set the real time clock in the device as specified in a defined control word.

For the clock, the year can be set as two digits only.

Hexadecimal code: **7FF9h**

See chapter “Date and Time Image“ on page 6-233.

6.12.14 Data Set Transfer from Controller to Operating Device (Block Mode)

You can use the following control code from the controller, to have a data set transferred from the controller to the operating device. The data are transferred in block mode.

The number of the data set must be written to the variable defined for this purpose.

In addition, the corresponding variables for the transfer buffers must be defined.

Hexadecimal code: **7FFAh**

6.12.15 Data Set Transfer from Operating Device to Controller

You can use the following control code from the controller, to have the operating device transfer a data set from the operating device to the controller.

The number of the data set must be written to the variable defined for this purpose.

In addition, the corresponding variables for the transfer buffers must be defined.

Hexadecimal code: **7FFBh**

6.12.16 Send Keyboard Image to Controller

You can use the following control code from the controller, to have the current keyboard status transferred from the operating device to the controller.

Hexadecimal code: **7FFCh**

6.12.17 Data Set Transfer from Controller to Operating Device (Single Mode)

You can use the following control code from the controller, to have a data set transferred from the controller to the operating device. The data are read in single mode.

The number of the data set must be written to the variable defined for this purpose.

In addition, the corresponding variables for the transfer buffers must be defined.

Hexadecimal code: **7FFDh**

6.12.18 Erase Serial Message Memory

You can use the following control code from the controller, to have the entire message memory of the operating device's serial message system erased.

Hexadecimal code: **7FFEh**

6.12.19 Refresh Message System

You can use the following control code from the controller, to have the operating device load all new parallel messages.

This allows implementation of an event-controlled message system.

Hexadecimal code: **7FFFh**

6.13 Working with a Real-Time Clock in the Operating Device

Each operating device has a real time clock. You set the parameters of the real time clock in the system parameters. You use system variables to set the time, date, and weekday in the operating device, and insert these variables in any screen.

You can transfer the data for the real time clock to the connected controller cyclically or on request, or provide values from the controller to the real time clock on request.

To allow the values to be exchanged, you must agree on a variable in which the image of date and time is stored. Enter the name of this variable in the system parameters of the real time clock either for setting the real-time clock or for transferring the real-time clock to the controller.

6.13.1 Date and Time Image

The time and date image describes the structure of the array variables that must be defined for setting and updating the time.

The date and time image is exchanged in the BCD format. For the image, you require an array variable with up to 8 bytes.

The length of the array variable is based on the length of the year specified. The following table illustrates the image with a 4-digit year:

Address			**Content**
Address + 0	H	H	Century (00 to 99)
Address + 1	Y	Y	Year (00 to 99)
Address + 2	M	M	Month (01 to 12)
Address + 3	D	D	Day (01 to 31)
Address + 4	h	h	Hour (00 to 23)

Fig. 6-105: Image of date and time with a 4-digit year

Address			Content
Address + 5	m	m	Minute (00 to 59)
Address + 6	s	s	Seconds (00 to 59)
Address + 7	W	W	Weekday (0 to 6 or 1 to 7)

Fig. 6-105: Image of date and time with a 4-digit year

Address			Content
Address + 0	Y	Y	Year (00 to 99)
Address + 1	M	M	Month (01 to 12)
Address + 2	D	D	Day (01 to 31)
Address + 3	h	h	Hour (00 to 23)
Address + 4	m	m	Minute (00 to 59)
Address + 5	s	s	Seconds (00 to 59)
Address + 6	W	W	Weekday (0 to 6 or 1 to 7)

Fig. 6-106: Image of date and time with a 2-digit year

The byte for the weekday is independent of the calendar and always runs Modulo 6.

Create the names of the weekdays in a text list. To display the weekdays in the operating device, in any screen create the system variable **RTCDayofWeek** with the representation type **Selection Text**. Link this variable with the weekday text list.

You must order the names of the weekdays in the correct sequence. You can select any starting point.

Value	Text
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

Fig. 6-107: Text list for operating devices with a Z80-CPU or RISC-CPU

6.13.2 Setting the Real Time Clock from the Controller

To update the real time clock data in the operating device from the controller, you must firstly create a variable in which the controller will store the date and time image. Enter this variable in the system parameters for the real time clock in the field **Setup**.

Finally, write the control code **7FF9h** in the serial message channel. This instructs the operating device to read the date and time image once from the agreed variable.

6.13.3 Transferring the Real-Time to the Controller

To transfer the real time clock data from the operating device to the controller, you must firstly create a variable in which the operating device will store the date and time image. Enter this variable in the system parameters for the real time clock in the field **Update**.

Then specify a polling time with which you want the operating device to write data at cyclical intervals into the variable.

6.14 Working with the Help Function

For each screen and each input variable in the project, you can create a help screen, and link these screens with each other. If you do not create or link any help screens, the default help screen is displayed.

The help texts are always limited to the size of one single screen.

6.14.1 Help Screen for Screens

You can create a separate help screen for each screen.

You can link the help screen with the screen using the screen parameters.

If you are in a screen and data release has not been requested, the help screen appears for this screen for the length of time you press the Help key, or after you have pressed a button that has been programmed accordingly.

In order for the button to simulate the key function of the Help key, you must create the Help key using the **key simulation** function, and link it with the system variable **KeyHelp**.

See chapter “KeyHelp“ on page 6-192.

6.14.2 Help Screen for Input Variable

You can create a separate help screen for each input variable.

You can link the help screen with the variable using the variable parameters.

If you are in a screen that contains a variable and data release has been requested, the cursor must be located at the variable. In this case, the help screen appears for the length of time you press the Help key, or press the button that has been programmed accordingly.

The help screen for an input variable is specifically designed for specifying the permitted range of values for the current input variable.

In order for the button to simulate the key function of the Help key, you must create the Help key using the **key simulation** function, and link it with the system variable **KeyHelp**.

See chapter “KeyHelp“ on page 6-192.

6.14.3 Help Screen for Message Screens

You can only create one help screen for a screen that contains a message field. You can not call a help screen for any programmed input variables in the message screen.

If you are in a screen that contains a message field and data release has not been requested, the help screen appears for the length of time you press the Help key, or after you have pressed a button that has been programmed accordingly.

In order for the button to simulate the key function of the Help key, you must create the Help key using the **key simulation** function, and link it with the system variable **KeyHelp**.

See chapter “KeyHelp“ on page 6-192.

6.15 Working with Function Keys / Softkey Functions

Another important feature, in addition to the screens, are the function keys and their LEDs. Function keys are user-programmable. They can be used as direct selector keys to switch to another screen or as control keys for the machine. When used as control keys, the integrated LEDs provide feedback information.

Programming the function keys as direct selector keys allows fast, direct access to the screens as well as to entire menu structures.

In the programming system, the combination of direct selection and control function can be programmed for function keys and for softkeys. Only the press codes of the keys should be evaluated in this mode of operation. This is because, depending on the length of time the key is pressed and the nature of the assigned screen, the stop code may have already changed!

6.15.1 Direct Selector Keys

Direct selector keys are function keys programmed to directly call up a specific screen. Pressing this function key thereby allows you to directly change to another screen.

This change of screen is not possible if the data release has been requested (status LED in the Data Release key is flashing or lights up) in a screen without automatic data release.

Direct selector keys allow speedy and convenient operation.

6.15.2 Function Keys in the Controller

In addition to programming function keys as direct selector keys, they can be programmed to carry out a function in the controller. To do this, instead of assigning a screen change to a function key, assign it the symbolic name of a controller variable in the application description.

When you press the key, it can set or reset the variable, and the same functions are assigned to it when you release the key. If you assign the set function to the key, the value entered is assigned to the data type.

In other words:

If the digit 1 is entered as the value:

- A flag bit receives logical 1
- A flag byte receives the value 01h
- A flag word receives the value 0001h
- A double word receives the value 00000001h

For values greater than 1, you must specify at least a byte address for the variable.

6.15.3 Softkeys

Softkeys are function keys that carry out a different function, depending on the screen in which the appear. The current function of a softkey is described in the current screen. In this context, you can use images, background images, selection images, static texts, and selection texts.

If you use a selection text to label softkeys, you can use the function key for several functions within a screen.

The action to be performed is determined by the:

- Screen number
- Number of the selection text
- Variable value transferred with the softkey.

Depending on the operating device, the number of keys you can use as softkeys varies.

Example:

We want a softkey (F1) in screen 10 to be able to switch a pump on and off.

1. Create a text list (pump) with two entries.

Value	Text
0	Switch Pump OFF
1	Switch Pump ON

Fig. 6-108: Text list for example softkey.

2. Define the variables.

Symbolic Name	Address (Example)
Softkey Labeling	M100.0
Softkey Status	M100.1
Image of the screen	MW110

Fig. 6-109: Variables for example softkey.

3. Create the screen (number 10).

Set up a controller variable (M 100.0) next to or above a function key.

Link the controller variable with the representation type Selection Text for cyclical output with the text list (pump).

Link the function key F1 of the screen with the variable Softkey Status (M 100.1), (set/reset).

4. Create the controller program to perform the following: Output A32.0 is to be used to control the pump.

Evaluate the screen number (MW 110); (must have the value 10).

Evaluate the edge for M 100.1.

Create a ELTACO function for pump output A32.0.

Use it to set flag M 100.0 to 0 when the pump is on.

Use it to set flag M 100.0 to 1 when the pump is off

6.15.4 Reaction Time of Function and Soft Keys

Whenever function keys need to influence PLC variables, they are given highest priority when transferred via the protocol. The reaction times during the transfer procedure are protocol-specific and range from 60 to 120 ms. This is the period of time which elapses after a key has been pressed until an output is set or reset in the PLC. The reaction time varies depending on the protocol itself, the load on the protocol (cyclical data, etc.) and the cycle time of the PLC.

Note that reaction times can be influenced by the polling times of the variables, messages and images of the LEDs.

6.15.5 Using Control Keys as Function Keys

Control keys can alternatively be used as function keys to trigger certain actions in the PLC. They can be defined to carry out the same functions as function keys, i.e. they are capable of assigning any values to a variable. The transfer procedure is independent of the screen parameter assignment. Thus, if a control key is to carry out a specific function in a screen, it should not be programmed as a ‘screen selector key’ at the same time. The screen-specific evaluation is identical to that of the function key.

6.15.6 Status LEDs of Function Keys

For each function key status LED, a 2-bit piece of information is available in the cyclical polling area. One bit activates or deactivates the corresponding status LED, the other bit displays the flashing attribute of the status LED. The status LEDs can only be influenced by the controller.

The following exceptions apply:

- You have programmed a function key as a direct selector key for a message screen
- You have entered a value greater than 0 (zero) as the message priority.

In these cases, the status LED of this function key can not be influenced by the controller! In these situations, the status LED can only be controlled using the message functions.

If the operating device you are using has less status LEDs than can be controlled here, the superfluous bits have no function.

To minimize the transfer times, select the length of the polling area so that only the bytes required for status LEDs are transferred.

See chapter “Message Priority for Direct Display“ on page 6-144.

6.16 Working with the Cyclic Poll Area

The cyclical polling area is a freely definable memory area in the controller.

The controller writes to this memory area.

The operating device polls this memory area cyclically. In other words, it reads the content in regular intervals.

The polling area is created in a byte-oriented or word-oriented manner.

The controller must be able to access this memory area bit-by-bit, and the memory area must be continuous.

The operating device accesses this memory area byte-by-byte or word-by-word.

The polling area is broken down into three zones:

1. Write Coordination byte (1 byte)
2. Serial message channel (2 bytes – high byte and low byte)
3. Control bytes for the status LEDs of the function keys (number depends on the operating device type)

You must enter the starting address for the polling area in the system parameters for the polling area.

Here, enter the length of the polling area and the polling time as well.

The length of the polling area is based on the number of status LEDs on the operating device that is being used.

The polling time is based on the total system load. Note the cycle times for other variables!

The structures of byte- and word-oriented polling areas are a little different. Therefore, a selection cannot be changed once made.

6.16.1 Byte-Oriented Polling Area

The byte-oriented polling area is located on a byte address. The controller must be able to access this area in bit-mode!

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte Address +0	Write Coordination Byte							
	Free	Free	BS	DDF	LM	PL	RQ	ED
Byte Address +1	Serial Message Channel Low Byte							
Byte Address +2	Serial Message Channel High Byte							
Byte Address +3	LED1 On/Off	LED1 Flash	LED2 On/Off	LED2 Flash	LED3 On/Off	LED3 Flash	LED4 On/Off	LED4 Flash
Byte Address +4	LED5 On/Off	LED5 Flash	LED6 On/Off	LED6 Flash	LED7 On/Off	LED7 Flash	LED8 On/Off	LED8 Flash
Byte Address +5	LED9 On/Off	LED9 Flash	LED10 On/Off	LED10 Flash	LED11 On/Off	LED11 Flash	LED12 On/Off	LED12 Flash
Byte Address +6	LED13 On/Off	LED13 Flash	LED14 On/Off	LED14 Flash	LED15 On/Off	LED15 Flash	LED16 On/Off	LED16 Flash
Byte Address +7	LED17 On/Off	LED17 Flash	LED18 On/Off	LED18 Flash	LED19 On/Off	LED19 Flash	LED20 On/Off	LED20 Flash
Byte Address +8	LED21 On/Off	LED21 Flash	LED22 On/Off	LED22 Flash	LED23 On/Off	LED23 Flash	LED24 On/Off	LED24 Flash
Byte Address +9	LED25 On/Off	LED25 Flash	LED26 On/Off	LED26 Flash	LED27 On/Off	LED27 Flash	LED28 On/Off	LED28 Flash
Byte Address +10	LED29 On/Off	LED29 Flash	LED30 On/Off	LED30 Flash	LED31 On/Off	LED31 Flash	LED32 On/Off	LED32 Flash
Byte Address +11	LED33 On/Off	LED33 Flash	LED34 On/Off	LED34 Flash	LED35 On/Off	LED35 Flash	LED36 On/Off	LED36 Flash
Byte Address +12	LED37 On/Off	LED37 Flash	LED38 On/Off	LED38 Flash	LED39 On/Off	LED39 Flash	LED40 On/Off	LED40 Flash
Byte Address +13	LED41 On/Off	LED41 Flash	LED42 On/Off	LED42 Flash	LED43 On/Off	LED43 Flash	LED44 On/Off	LED44 Flash
Byte Address +14	LED45 On/Off	LED45 Flash	LED46 On/Off	LED46 Flash	LED47 On/Off	LED47 Flash	LED48 On/Off	LED48 Flash

Fig. 6-110: Byte-oriented polling area

See chapter “Write Coordination Byte“ on page 6-247.
See chapter “Serial Message Channel“ on page 6-242.

6.16.2 Word-Oriented Polling Area

The word-oriented polling area is located on a word address. The controller must be able to access this area in bit-mode!

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Word Address +0	Write Coordination Byte															
	Free	Free	BS	DDF	LM	PL	RQ	ED	Free	Free	Free	Free	Free	Free	Free	Free
Word Address +1	Serial Message Channel High Byte								Serial Message Channel Low Byte							
Word Address +2	LED1 On/Off	LED1 Flash	LED2 On/Off	LED2 Flash	LED3 On/Off	LED3 Flash	LED4 On/Off	LED4 Flash	LED5 On/Off	LED5 Flash	LED6 On/Off	LED6 Flash	LED7 On/Off	LED7 Flash	LED8 On/Off	LED8 Flash
Word Address +3	LED9 On/Off	LED9 Flash	LED10 On/Off	LED10 Flash	LED11 On/Off	LED11 Flash	LED12 On/Off	LED12 Flash	LED13 On/Off	LED13 Flash	LED14 On/Off	LED14 Flash	LED15 On/Off	LED15 Flash	LED16 On/Off	LED16 Flash
Word Address +4	LED17 On/Off	LED17 Flash	LED18 On/Off	LED18 Flash	LED19 On/Off	LED19 Flash	LED20 On/Off	LED20 Flash	LED21 On/Off	LED21 Flash	LED22 On/Off	LED22 Flash	LED23 On/Off	LED23 Flash	LED24 On/Off	LED24 Flash
Word Address +5	LED25 On/Off	LED25 Flash	LED26 On/Off	LED26 Flash	LED27 On/Off	LED27 Flash	LED28 On/Off	LED28 Flash	LED29 On/Off	LED29 Flash	LED30 On/Off	LED30 Flash	LED31 On/Off	LED31 Flash	LED32 On/Off	LED32 Flash
Word Address +6	LED33 On/Off	LED33 Flash	LED34 On/Off	LED34 Flash	LED35 On/Off	LED35 Flash	LED36 On/Off	LED36 Flash	LED37 On/Off	LED37 Flash	LED38 On/Off	LED38 Flash	LED39 On/Off	LED39 Flash	LED40 On/Off	LED40 Flash
Word Address +7	LED41 On/Off	LED41 Flash	LED42 On/Off	LED42 Flash	LED43 On/Off	LED43 Flash	LED44 On/Off	LED44 Flash	LED45 On/Off	LED45 Flash	LED46 On/Off	LED46 Flash	LED47 On/Off	LED47 Flash	LED48 On/Off	LED48 Flash

Fig. 6-111: Word-oriented polling area

See chapter “Write Coordination Byte“ on page 6-247.
See chapter “Serial Message Channel“ on page 6-242.

6.16.3 Serial Message Channel

The serial message channel is a part of the cyclical polling area and is used to transfer 16-bit information. The numbers of serial messages, selection of message screens, external selection of screens and transfer of control codes are made possible via this data channel.

The following handshake is used for the information transfer:

The PLC stores a value (> 0) in this data word. This value is then transferred to the operating device which will write the value 0 into this data word again. This indicates to the PLC that it can now transfer the next value. The value is interpreted by the operating terminal and its function is executed.

Values can be:

- Message numbers
- Screen numbers (screen number + 8000H)
- Control Codes

6.16.4 Image of Status LEDs

The LED image enables the controller to control the status LEDs of the function keys of the connected operating device. The functions ON, OFF, or FLASHING can be set for each status LED. As soon as the controller sets a bit, the assigned LED on the operating device is influenced accordingly.

In this context, it is important that the length of the polling area and the polling time were also set correctly. If these additional parameters were not set correctly, problems may occur during the LED control.

For a function key that leads directly to the message screen, the status LED is influenced by the message system. In this way, the message system indicates that a new message has been received and has not yet been acknowledged. To influence the status LED of this function key from the controller, you must set the message priority to 0 (zero).

Bit 1	Bit 2	Status of the LED
0	0	OFF
0	1	OFF, FLASHING is preset
1	0	ON
1	1	FLASHING

Fig. 6-112: Truth table for a status LED

6.16.5 Polling Time

The polling time specifies the intervals the operating device will use to read the variable for the cyclical polling area. The polling of this variable also covers the Write Coordination byte, the serial message channel, and the image of the status LEDs.

Settings in or around a half a second have proven useful in most protocols. If the cycle time set is too low, the interface protocol can no longer follow requests, and reaction performance deteriorates.

There is no universal recipe, however.

The options available primarily depend on the individual project. However, at the very least, times greater than 100 ms should be preselected. For further information, please contact our support hotline.

6.16.6 Size of the Polling Area

Depending on the data type and operating device, the polling area has a length of up to 23 bytes. The entry allows adjustment to suit the area actually used, if you can avoid using the image of the status LED or part of this. The basic setting for all operating devices is a length of 12 bytes.

6.16.7 Read Coordination Byte

The Read Coordination byte is used for handshake and data coordination with the controller.

If necessary, the controller reads the Read Coordination byte and evaluates the individual bits.

Bit	Abbreviation	Function
0	EA	Editing Request
1	EZ	Editing Status
2	RA	Refresh Request
3	LM	Liveness Flag
4	DDA	Data Set Download Active
5		Not used
6		
7		

Fig. 6-113: Structure of the Read coordination byte

The Read Coordination byte only works together with the Write Coordination byte.

6.16.7.1 Editing Request

The operating device uses the Editing Request bit to indicate to the controller that the value of a variable will be changed.

For this purpose, the operating device writes a logical 1 to the **Editing Request** bit in the Read Coordination byte.

To allocate an editing release to the operating device, the controller writes a logical 1 to the Editing Release bit in the Write Coordination byte.

6.16.7.2 Editing Status

The operating device uses the Editing Status bit to indicate to the controller that the value of a variable could be changed.

Once the operating device has received the Editing Release from the controller, the device sets the **Editing Status** bit in the Read Coordination byte to logical 1.

The operator can now change the variable value. To send the changed variable value to the controller, the operator must select the **Enter** key to complete the entry**.**

The operator can then change other variable values.

Then, the operator must press the **Data Release** key. This resets the **Editing Status** bit to logical 0.

The **Refresh Request** and **Refresh Acknowledgment** bits are used to write the new variable value to the controller.

Once the controller has read the new variable value, it uses the **Refresh Acknowledgment** bit to indicate of the Write Coordination byte that the **Editing Status** bit can once again be reset to logical 0.

6.16.7.3 Refresh Request

If you changed a variable value in the operating device, and selected the Data Release key, the **Refresh Request**bit in the Read coordination byte must be set to logical 1.

This triggers the read process in the controller, and then confirms it with the **Refresh Acknowledgment** bit in the Write Coordination byte.

6.16.7.4 Liveness Flag (Read Coordination Byte)

In some communication protocols, you can not control the operability of the interface in the controller. The **Liveness flag** has been developed to address this shortfall. This is a simple function, which has proven very effective in practice.

Whenever the controller needs to know whether the connection is still active, it writes a logical 1, and subsequently a logical 0, to bit 3 of the

Write Coordination byte.

The operating device constantly monitors the Liveness flag in the Write Coordination byte and compares it with the status of the Liveness flag in the Read Coordination byte. As soon as the two bytes are no longer the same, the operating device copies bit 3 from the Write Coordination byte to the Read Coordination byte.

Within a timeout time, the controller must now also check whether both statuses are identical.

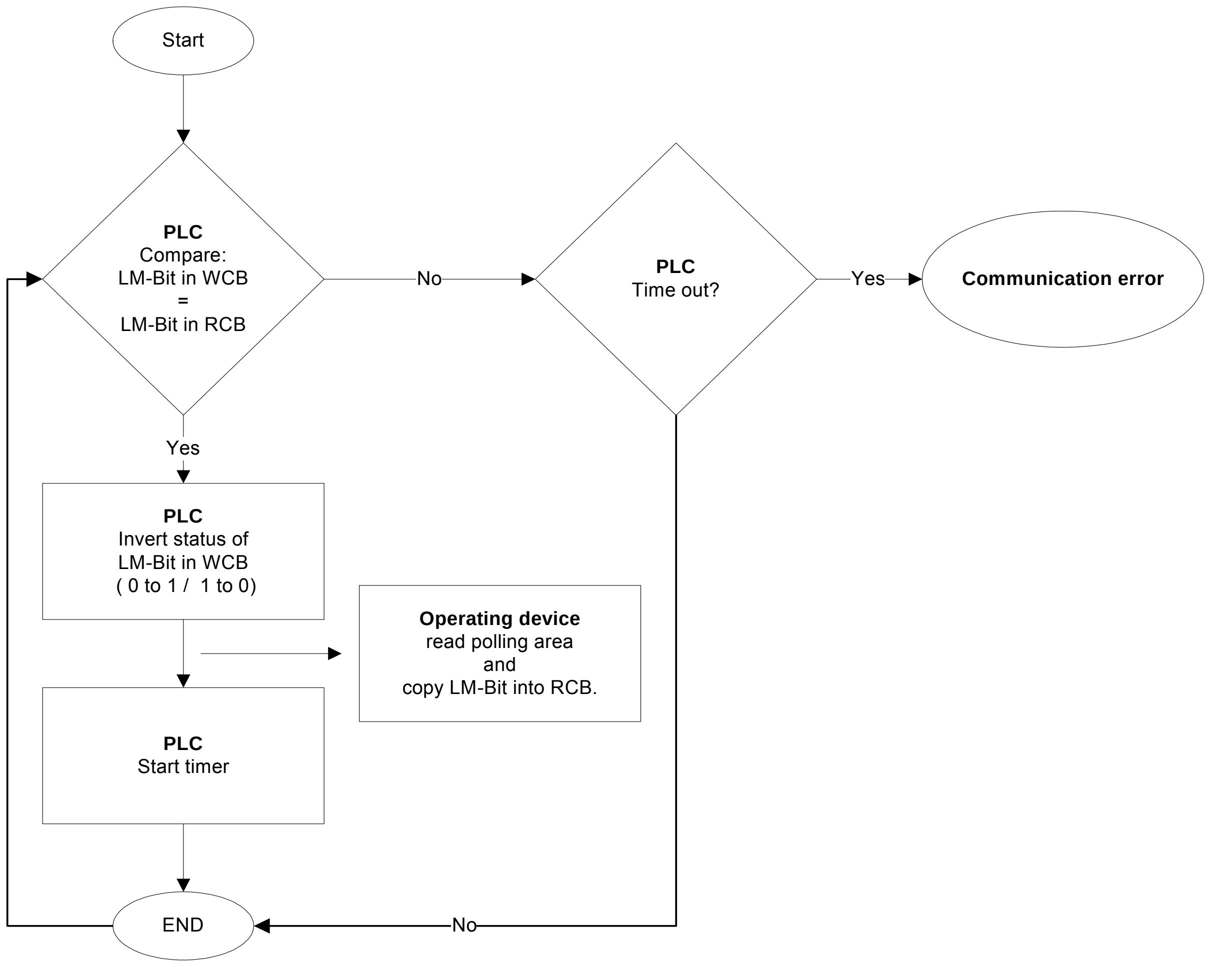


Fig. 6-114: Monitoring the liveness flag

Define the following settings for the liveness flag monitoring.

System parameters, Poll area:

- Create a variable for the Write Coordination byte.
- Enter a polling time that meets the requirements of real life situations.

System parameters, General parameters:

- Specify a polling time that meets the requirements of real life situations.

- Create a variable for the Read Coordination byte.

When you determine the timeout time in the controller, remember to take the transfer times and polling times into account.

6.16.7.5 Data Set Download Active

As soon as the operating device transfers a data set to the controller, it writes a logical 1 to the Data Set Download Active bit. After all data were sent, the operating device writes a logical 0 to the Data Set Download Active bit. The controller can now work with the new data set values.

6.16.8 Write Coordination Byte

The term Write Coordination byte indicates that the controller writes this byte.

The Write Coordination byte is only read by the operating device. This byte is used together with the Read Coordination byte for the handshake and data coordination with the controller.

Here, the controller indicates its current status to the operating device. The individual bits are independent of each other.

Bit	Abbrevia-tion	Function
0	ED	External Data Release
1	RQ	Refresh Acknowledgment
2	PL	Delete Password
3	LM	Liveness Flag
4	DDF	Data Set Download Release
5	BS	Screen Saver
6		Free
7		

Fig. 6-115: Structure of the write coordination byte

The Write Coordination byte only works together with the Read Coordination byte.

6.16.8.1 External Data Release

The controller can use the External Data Release bit to influence data release in the operating device. If the operator would like to change a value in the operating device, he must first request data release. For this purpose, the operating device writes a logical 1 to the Editing Request bit in the Read Coordination byte. During this time, the Data Release status LED flashes.

Once the controller establishes that the Editing Request bit in the Read Coordination byte is set to logical 1, it can release the editing process in the operating device by setting the External Data Release bit in the Write Coordination byte to logical 1. The Data Release status LED is then lit.

6.16.8.2 Refresh Acknowledgement

Once the controller has read the Refresh Request bit in the Read Coordination byte as logical 1, it can read in the changed variable value. When finished, the controller can write a logical 1 to the Refresh Acknowledgment bit, and confirm execution to the operating device.

6.16.8.3 Delete Password

When the operator exits a screen for which he requires a password for access, password protection needs to be activated again for this screen. This can be forced by the controller by entering a logical 1 in the Delete Password bit.

6.16.8.4 Liveness Flag (Write Coordination Byte)

In some communication protocols, you can not control the operability of the interface in the controller. The **Liveness flag** has been developed to address this shortfall. This is a simple function, which has proven very effective in practice.

Whenever the controller needs to know whether the connection is still active, it writes a logical 1, and subsequently a logical 0, to bit 3 of the Write Coordination byte.

The operating device constantly monitors the Liveness flag in the Write Coordination byte and compares it with the status of the Liveness flag in the Read Coordination byte. As soon as the two bytes are no longer the same, the operating device copies bit 3 from the Write Coordination byte to the Read Coordination byte.

Within a timeout time, the controller must now also check whether both statuses are identical.

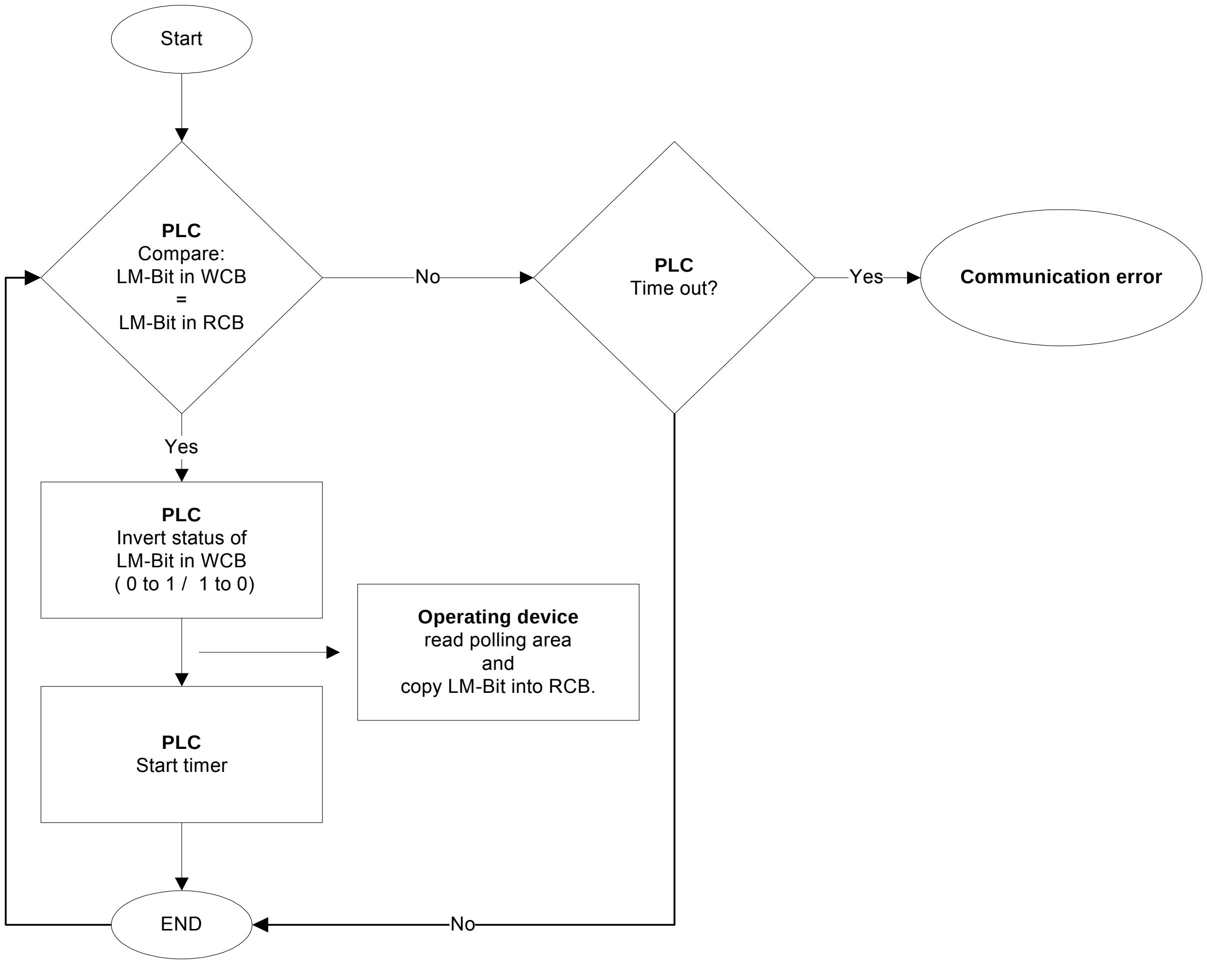

Start
PLC Compare: LM-Bit in WCB = LM-Bit in RCB
No
PLC Time out?
Yes
Communication error
Yes
PLC Invert status of LM-Bit in WCB (0 to 1 / 1 to 0)
Operating device read polling area and copy LM-Bit into RCB.
PLC Start timer
END
No

Fig. 6-116: Monitoring the liveness flag

Define the following settings for the liveness flag monitoring.

System parameters, Poll area:

- Create a variable for the Write Coordination byte.
- Enter a polling time that meets the requirements of real life situations.

System parameters, General parameters:

- Specify a polling time that meets the requirements of real life situations.
- Create a variable for the Read Coordination byte.

When you determine the timeout time in the controller, remember to take the transfer times and polling times into account.

6.16.8.5 Data Set Download Release

The controller determines the start time of a data set transfer from the operating device to the controller by writing a logical 1 in the Data Set Download Release bit.

6.16.8.6 Screensaver

If the controller sets the BS bit of the write coordination byte (WCB) the display of the operating device is turned off immediately.

6.17 How do I Configure the Contrast/Brightness Setting for the Operating Device?

For all operating devices, the contrast/brightness is set by the software.

For this purpose, you need to set up one of the following the system variables in any screen:

- **LcdContrast** to set the contrast.
- **LcdBackLight** to set the brightness.

1. Open the **Language** branch of the project tree.
2. Double-click the name of any screen.

The screen editor opens the screen.

3. In the toolbar, click the **Variable** button.
4. Drag the mouse to open a frame.

The **Screen element variable** dialog appears.

5. Select the system variable **LcdContrast** or **LcdBackLight** in the variable folder **System variables/basic functions**.
6. Select the **Decimal number** representation type.
7. Click the **Edit type** button.

The **Decimal number** dialog appears.

8. Select the **Input** radio button from the **Field type** area.
9. If necessary, modify the settings made in the **Format** area.

Use the following procedure to enter the contrast value numerically:

10. Select the **Standard** radio button from the **Editor** area.
11. Select the **with Enter** radio button from the **Data acceptance** area.

Use the following procedure to enter the contrast value incrementally:

12. Select the **Increment** radio button from the **Editor** area.
13. Select the **Upon any modification** (for each change) radio button from the **Data acceptance** area.

Use the following procedure to enter the contrast value in mixed mode:

14. Select the **Mixmode** radio button from the **Editor** area.
15. Select the **with +, - or Enter** radio button from the **Data acceptance** area.
16. Enter the permitted minimum input value into the **Lower limit** field.
17. Enter the permitted maximum input value into the **Upper limit** field.
18. Click **OK** to confirm your entries.

You are returned to the **Screen element Variable** dialog.

19. Click **OK** to confirm your entries.

You are returned to the screen editor.

Please refer to the user manual of the respective operating device for the upper and lower limit values. For information, please refer to the chapter Contrast Setting or Brightness Setting, respectively.

When selecting the input mode, take the capabilities offered by the respective operating device into consideration. If the operating device is not equipped with a numeric keypad, only the Incremental mode may be available for use. On touch-screen devices, a keypad is automatically displayed to allow you to enter the values either numerically or incrementally.

6.18 Renaming Objects

Enter any unique name for the object.

Confirm with **OK**.

6.19 Image of the Screen Number

You can have the number of the current screen of the operating device written to a controller variable or script variable. The controller variable must be a 16-bit variable. Only the first 16 bits are transferred at script-variables (32 bits). The following 16 bits are not changed and may not be evaluated.

For each screen change, the operating device writes the current screen number in this variable.

This means that you can access the user interface from the controller.

6.20 Image of the Keyboard

In a chain of bytes, there is one bit that displays the status of each key of an operating device. If the bit for a key is set to logical 1, this means that the key is pressed. Once the key is released, the bit is set to logical 0 again.

To enable the keyboard image to be read, request code 7FFCh must first be written into the cyclical polling area. The operating device then writes the current keyboard image into the agreed variable in the controller.

Each operating device can have a specific number of keys and therefore has its own keyboard image.

6.20.1 Keyboard Image for the VCP 02.2

The keyboard image consists of eight bytes that must be stored contiguously in the controller.

Address	**Byte Number**
+ 0	1
+ 1	2
+ 2	3
+ 3	4
+ 4	5
+ 5	6
+ 6	7
+ 7	8

Fig. 6-117: Arrangement of the keyboard image in the controller

Byte Number	**Bit**	**Key**	**Byte Number**	**Bit**	**Key**
1	0	F1	2	0	F2
	1	Cursor Left		1	Cursor Down
	2	Not assigned		2	Not assigned
	3	Not assigned		3	Not assigned
	4	Not assigned		4	Not assigned
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned
3	0	F3	4	0	F4
1	Plus	1	Minus		
2	Not assigned	2	Not assigned		
3	Not assigned	3	Not assigned		
4	Not assigned	4	Not assigned		
5	Not assigned	5	Not assigned		

Fig. 6-118: VCP 02.2 keyboard image

Byte Number	**Bit**	**Key**	**Byte Number**	**Bit**	**Key**
6	Not assigned	6	Not assigned		
7	Not assigned	7	Not assigned		
5	0	Data Release	6	0	Help
	1	Enter	1	Not assigned	
	2	Not assigned	2	Not assigned	
	3	Not assigned	3	Not assigned	
	4	Not assigned	4	Not assigned	
	5	Not assigned	5	Not assigned	
	6	Not assigned	6	Not assigned	
	7	Not assigned	7	Not assigned	
7	0	Not assigned	8	0	Not assigned
	1	Not assigned		1	Not assigned
	2	Not assigned		2	Not assigned
	3	Not assigned		3	Not assigned
	4	Not assigned		4	Not assigned
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned

Fig. 6-118: VCP 02.2 keyboard image

6.20.2 Keyboard Image for the VCP 05.2

The keyboard image consists of eight bytes that must be stored contiguously in the controller.

Address	Byte Number
+ 0	1
+ 1	2
+ 2	3
+ 3	4
+ 4	5
+ 5	6
+ 6	7
+ 7	8

Fig. 6-119: Arrangement of the keyboard image in the controller

Byte Number	Bit	Key	Byte Number	Bit	Key
1	0	Data Release	2	0	Cursor Left
	1	Enter		1	Cursor Right
	2	Delete		2	Cursor Up
	3	Help		3	Cursor Down
	4	Cursor Home		4	Decimal point
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned
3	0	0	4	0	5
	1	1		1	6
	2	2		2	7
	3	3		3	8
	4	4		4	9
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned

Fig. 6-120: VCP 05.2 keyboard image

Byte Number	Bit	Key	Byte Number	Bit	Key
5	0	Plus	6	0	F4
	1	Minus	1	F5	
	2	F1	2	F6	
	3	F2	3	Not assigned	
	4	F3	4	Not assigned	
	5	Not assigned	5	Not assigned	
	6	Not assigned	6	Not assigned	
	7	Not assigned	7	Not assigned	
7	0	Not assigned	8	0	Not assigned
	1	Not assigned		1	Not assigned
	2	Not assigned		2	Not assigned
	3	Not assigned		3	Not assigned
	4	Not assigned		4	Not assigned
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Page down
	7	Not assigned		7	Print

Fig. 6-120: VCP 05.2 keyboard image

6.20.3 Keyboard Image for the VCP 08.2

The keyboard image consists of eight bytes that must be stored contiguously in the controller.

Address	Byte Number
+ 0	1
+ 1	2
+ 2	3
+ 3	4
+ 4	5
+ 5	6
+ 6	7
+ 7	8

Fig. 6-121: Arrangement of the keyboard image in the controller

Byte Number	Bit	Key	Byte Number	Bit	Key
1	0	Cursor Left	2	0	Data Release
	1	Enter		1	Cursor Right
	2	Delete		2	Cursor Up
	3	Help		3	Cursor Down
	4	Cursor Home		4	Decimal point
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned
3	0	0	4	0	5
	1	1		1	6
	2	2		2	7
	3	3		3	8
	4	4		4	9
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned

Fig. 6-122: VCP 08.2 keyboard image

Byte Number	Bit	Key	Byte Number	Bit	Key
5	0	Plus	6	0	F4
	1	Minus	1	F5	
	2	F1	2	F6	
	3	F2	3	F7	
	4	F3	4	F8	
	5	Not assigned	5	Not as-signed	
	6	Not assigned	6	Not as-signed	
	7	Not assigned	7	Not as-signed	
7	0	F9	8	0	F14
	1	F10		1	F15
	2	F11		2	Not assigned
	3	F12		3	Not assigned
	4	F13		4	Not assigned
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Page down
	7	Not assigned		7	Print

Fig. 6-122: VCP 08.2 keyboard image

6.20.4 Keyboard Image for the VCP 20.2

The keyboard image consists of eight bytes that must be stored contiguously in the controller.

Address	Byte Number
+ 0	1
+ 1	2
+ 2	3
+ 3	4
+ 4	5
+ 5	6
+ 6	7
+ 7	8

Fig. 6-123: Arrangement of the keyboard image in the controller

Byte Number	Bit	Key	Byte Number	Bit	Key
1	0	F1	2	0	F2
	1	F9		1	F4
	2	F11		2	F12
	3	Cursor Left		3	Cursor Down
	4	7		4	8
	5	4		5	5
	6	1		6	2
	7	0		7	Decimal point
3	0	F3	4	0	F7
	1	F5		1	F6
	2	Cursor Up		2	Cursor Home
	3	Cursor Right		3	Help
	4	9		4	Data Release
	5	6		5	Plus
	6	3		6	Minus
	7	Delete		7	Enter

Fig. 6-124: VCP 20.2 keyboard image

Byte Number	Bit	Key	Byte Number	Bit	Key
5	0	F8	6	0	Not assigned
	1	F10	1	Not as-signed	
	2	Not assigned	2	Not as-signed	
	3	Not assigned	3	Not as-signed	
	4	Not assigned	4	Not as-signed	
	5	Not assigned	5	Not as-signed	
	6	Not assigned	6	Not as-signed	
	7	Not assigned	7	Not as-signed	
7	0	Not assigned	8	0	Not assigned
	1	Not assigned		1	Not assigned
	2	Not assigned		2	Not assigned
	3	Not assigned		3	Not assigned
	4	Not assigned		4	Not assigned
	5	Not assigned		5	Not assigned
	6	Not assigned		6	Not assigned
	7	Not assigned		7	Not assigned

Fig. 6-124: VCP 20.2 keyboard image

6.21 Starting the Small Operator Terminal

Once you have connected the small operator terminal to the supply voltage, the device boot procedure starts. The boot procedure also includes additional load procedures, which are used to load the operating system, the terminal file (project) and the visualization runtime.

The Windows CE operating system is installed on the operating device. Running on the operating system is the visualization runtime software TSvisRT.

6.21.1 Loading Procedure on Windows CE Operating Procedure

The initialization starts the Launch.exe program.

The program allows you to use the keys **Cursor Down** and **Enter** or the buttons to make changes to the configuration.

The Launch.exe program has 3 operating modes:

- Normal (no key / button is pressed)
- Setup Main (Key **Enter** / button **Press For Setup Main Menu** was pressed)
- Administration (**Cursor Down** key followed by **Enter** / **Admin** button was pressed)

6.21.1.1 Normal Operating Mode

The program AppStarter.exe starts from the internal Flash memory.

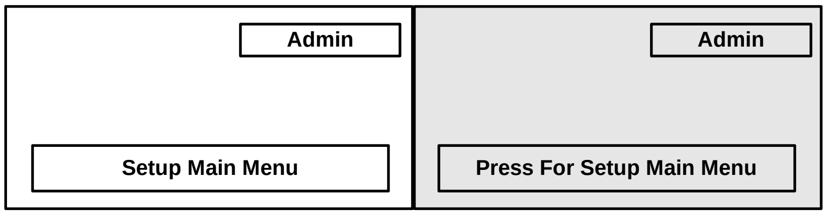


Fig. 6-125: Display after startup (operating devices with keys / operating devices with touch screen)

The following message is issued if the AppStarter.exe file does not exist.

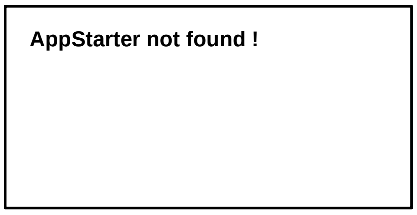

AppStarter not found !

Fig. 6-126: Error message after startup

6.21.1.2 Setup Main Operating Mode

If you press the key **Enter** or the **Press For Setup Main Menu** button during the startup phase, the Setup Main mode starts.

The normal entries apply to operating devices with keys only. The gray entries apply to operating devices equipped with a touch screen.

Tutorial

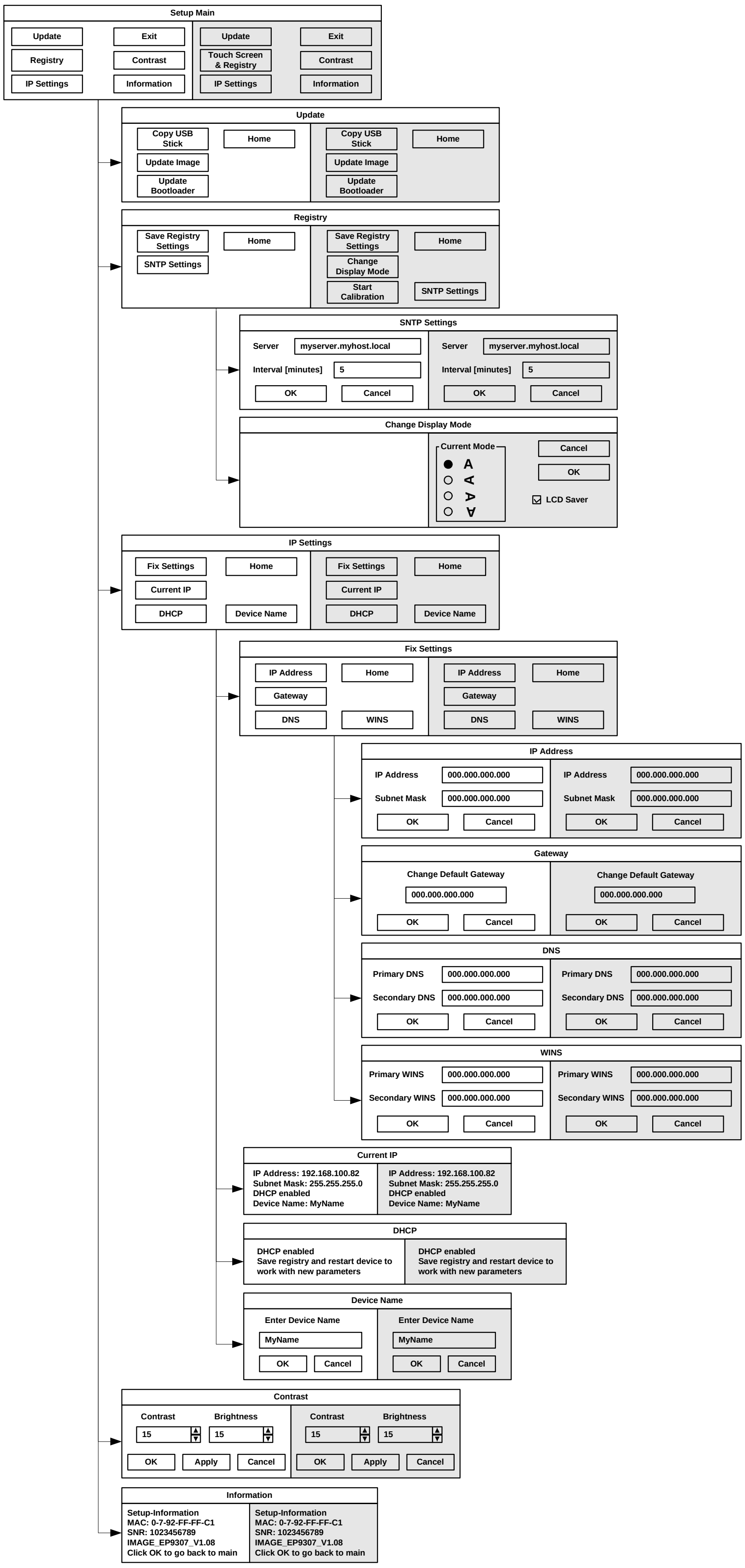


Fig. 6-127: Setup Main operating mode

Some settings are password-protected. The password is "+-+-".

Update, Copy USB Stick:

This function copies the data from the USB stick to the internal flash file system.

Several projects can be managed in subdirectories below the directory TSvisRT. If more than one project is in different subdirectories, a choice dialog is displayed. Only directories which contain a project file (xxxx.cb) are listed.

The entire TSvisRT directory or the corresponding subdirectory and the AppStarter.exe are copied into the target directory of the flash file system.

Update, Update Image:

If the Image subdirectory on the memory stick contains a file xxxx.nb0, this file is used to perform the image update. In this case, the Flash Registry is always deactivated so that the image is processed with its new default registry.
The user is informed in each case once the update has been successfully completed.

Update, Update Bootloader:

If the Bootloader subdirectory on the memory stick contains a file xxxx.nb0, this file is used to perform the bootloader update.
The user is informed in each case once the update has been successfully completed.

Registry, Save Registry Settings:

The entire registry is saved.

Registry, Change Display Mode:

Set-up of display adjustment.

Registry, Start Calibration:

The touch screen calibration process is started. After calibration, the values must be saved using "Save Registry".

Registry, SNTP Settings:

You can define a time server in the intranet or internet. The interval of synchronization is set in minutes.
This entry is password-protected.

IP Settings, Fix Settings, IP Address:

The system deselects DHCP and enters the information is read from the registry.

This entry is password-protected.

All addresses must be given in the format "xxx.xxx.xxx.xxx". Numbers smaller than 100 you have to fill up with zeros. (Example: 192.168.42.1 -> 192.168.042.001)

IP Settings, Fix Settings, Gateway:

The system deselects DHCP and enters the information is read from the registry.

This entry is password-protected.

All addresses must be given in the format "xxx.xxx.xxx.xxx". Numbers smaller than 100 you have to fill up with zeros. (Example: 192.168.42.1 -> 192.168.042.001)

IP Settings, Fix Settings, DNS:

The system deselects DHCP and enters the information is read from the registry.

This entry is password-protected.

All addresses must be given in the format "xxx.xxx.xxx.xxx". Numbers smaller than 100 you have to fill up with zeros. (Example: 192.168.42.1 -> 192.168.042.001)

IP Settings, Fix Settings, WINS:

The system deselects DHCP and enters the information is read from the registry.

This entry is password-protected.

All addresses must be given in the format "xxx.xxx.xxx.xxx". Numbers smaller than 100 you have to fill up with zeros. (Example: 192.168.42.1 -> 192.168.042.001)

IP Settings, Current IP:

Displays the current IP address, subnet mask, device name and the DHCP status.

IP Settings, DHCP:

The system enables DHCP. After enabling DHCP this setting must be saved using "Save Registry".

This entry is password-protected.

IP Settings, Device Name:

You can define a device name with up to 14 characters. Via a FTP connection you can access the device with the device name instead of the IP address.

This entry is password-protected.

Contrast:

The operating mode setup main is displayed with default values for contrast and brightness to ensure reading also at faulty values. If you change a value, you have to confirm this in a dialog. If you push **Cancel** or 5 seconds pass, the values are not accepted.

Information:

The following information is output: MAC address, serial number and image version.

6.21.1.3 Administration Operating Mode

If you press the **Cursor Down** key followed by the **Enter** key / **Admin** button during the startup phase, the Administration mode of operation starts.

You can use the Admin.ini file to manage the device. This file must exist in the root directory of the USB stick.

This file is used as a dongle to prevent users from changing the device during normal operation.

Possible contents for the Admin.ini file:

Observe upper and lower case for all entries!

Explorer=Off	Deactivates the Explorer in the registry. The change becomes effective on the next device reboot.
Explorer=On	Activates the Explorer in the registry. The change becomes effective on the next device reboot.
Registry=Default	Destroys the current registry and activates the default registry of the image. The change becomes effective on the next device reboot.
Start=MyProgramm.exe	Starts the application MyProgramm.exe
StartRepllog=On	Enables automatic startup of the Repllog.exe program in the registry. The change becomes effective on the next device reboot.
StartRepllog=Off	Disables automatic startup of the Repllog.exe program in the registry. The change becomes effective on the next device reboot.
DeviceName=MyName	Defines the device name of the operating device
Demomode=On	Enables demo mode for TSvisRT. The change becomes effective on the next device reboot.
Demomode=Off	Disables demo mode for TSvisRT. The change becomes effective on the next device reboot.
;Demomode=Off	Comment, no impact

6.21.2 Function of the AppStarter.exe Program

The AppStarter.exe program creates all the necessary registry settings and can also store the registry, if desired.

If the Explorer is activated, the system shuts it down.

The AppStarter.exe file then starts the TSvisLD_CE.exe file from the Flash File System (FFS).

6.21.3 Function of the TSvisLD.exe Program

The TSvisLD.exe loads the following components into the memory of the operating system in accordance with the instructions in the TSvisRT_CE.ini file:

- User application
- Protocol driver
- TSvisRT firmware

The program then unpacks the compressed application file (*.CB) and starts the TSvisRT Runtime component.

6.21.4 Memory Media Used

The following memory media are used in small operator terminals with the operating system Windows CE.

Fig. 6-128: Memory media used

Operating system memory TSvisRT Runtime Protocol driver Application	⇦	**Flash file system (FlashDrv)** AppStarter.exe Subdirectory TSvisRT\Project name (with TSvisRT Runtime, protocol driver and application)	⬅	USB stick (Hard disk)
		Registry settings	⬅	Admin.ini IPSetting.ini
	◀	Image storage in Flash	⬅	Subdirectory Image
	◀	Bootloader storage in Flash	⬅	Subdirectory Bootloader

Legend:

⇦	Copying carried out by operating system
◀	Copying carried out by the bootloader
⬅	Copying carried out by the Launch.exe

6.21.5 Important Files and Directories

File	**Storage location**	**Function**
AppStarter.EXE	Root directory on USB stick	Starts TSvisRT_LD.exe
Admin.INI	Root directory on USB stick	File with administration files
TSvisRT_CE.INI	Subdirectory**TSvisRT\Project name** on USB stick	Initialization file for TSvisRT_LD.exe
SPSTtxxxxxxx.yyy.DLL	Subdirectory**TSvisRT\Project name** on USB stick	Protocol driver
*.CB	Subdirectory**TSvisRT\Project name** on USB stick	Compressed application file
TSvisRT_CE.EXE	Subdirectory**TSvisRT\Project name** on USB stick	TSvisRT Runtime
TSvisLD_CE.EXE	Subdirectory**TSvisRT\Project name** on USB stick	TSvisRT loader
EBOOT.nb0	Subdirectory Bootloader	Windows CE Bootloader
NK.nb0	Subdirectory Bootloader	Operating system Windows CE

Fig. 6-129: Important files and directories

6.22 Communication With a Controller

Communication between a controller (host computer, for example) and an operating device may occur with any interface, except those for the logging printer and parallel outputs. The interface used always depends on the connected counterpart or on the network.

For more information on the interfaces themselves, see the manual for the corresponding operating device.

A standard cable, measuring about 3 m (9.843 ft), is available to ensure a secure connection for each connection option.

More detailed information is available on possible connections to different controllers and networks.

See chapter “Controller and Bus Connections“ on page 7-1.

7 Controller and Bus Connections

For more information on the individual connections possible for small operator terminals in the standard or bus models, see the following chapters. The same 25 pin D-SUB connector is used for small operator terminals of the standard model with a universal interface. For devices of the bus model, different connectors are used for the connection.

See chapter “3S serial“ on page 7-2.
See chapter “Bosch BUEP19E“ on page 7-15.
See chapter “BRC-Symbolic“ on page 7-25.
See chapter “DeviceNet“ on page 7-35.
See chapter “IndraLogic“ on page 7-61.
See chapter “PROFIBUS-DP raw“ on page 7-74.

7.1 3S serial

The protocol provides random read and write access to all global data objects of the controller.

The programming software adopts the data objects of the project_name.SYM file which are created when the IndraLogic project is compiled.

The connected operating device uses the symbolic name to access a data object.

7.1.1 Data Types

The length of a variable is determined by the length defined in the programming software IndraLogic.

7.1.1.1 Single Variables

You can access variables of the following type: BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL, and STRING. Floating point numbers are interpreted in IEEE format. The variable type REAL is required for this purpose.

7.1.1.2 String Variables

For string variables, the variable type STRING(N) is used, where N is the length of the string.

7.1.2 Programming

7.1.2.1 Protocol Parameters

Baud Rate

This parameter specifies the communication rate.

Configurable Values (Baud)	Default Value
4800	
9600	
19200	
38400	X

Fig. 7-1: Baud rate

Parity

This parameter specifies the parity used to control the communication.

Configurable Values	Default Value
None	X
Even	
Odd	

Fig. 7-2: Parity

Data Bits

This parameter specifies the number of data bits.

Configurable Values	Default Value
5	
6	
7	
8	X

Fig. 7-3: Data bits

Stop Bits

This parameter specifies the number of stop bits.

Configurable Values	Default Value
1	X
1.5	
2	

Fig. 7-4: Stop bits

Waiting Time for Response

Specify a waiting time for the Produced Data toggle bit monitoring.

Configurable Values	Default Value
0 ms, 50 ms to 65000 ms	500 ms

Fig. 7-5: Waiting time for response

Delay until Connection Set-Up

This parameter specifies the waiting time after which the operating device starts the communication.

Configurable Values	Default Value
5 s to 255 s	5 s

Fig. 7-6: Delay until connection set-up

Byte Order

This parameter specifies the destination hardware's CPU type.

Configurable Values	Default Value
Intel	
Motorola	X

Fig. 7-7: Byte order

Controllers

This parameter indicates the IndraLogic runtime system in the control.

Configurable Values	Default Value
Standard	X
PLCWinNT	

Fig. 7-8: Controllers

Path for Variable List *.sym

This parameter specifies the directory in which the variable list *.sym is stored.

To select a directory, click the Browse button.

The variable list *.sym is created by the programming software Indra-Logic when compilation takes place.

7.1.2.2 Polling Area

The poll area is used to manage the write coordination byte (WCB), the serial message channel and the LEDs in the function keys. This area is continuously polled by the operating device.

This protocol requires you to set up the poll area with three single variables.

Area	Permitted Data Types
CBW	BYTE
Message Channel	WORD
LEDs in the Function Keys	ARRAY[1..N] OF BYTE

Fig. 7-9: Dats types for the poll area

7.1.2.3 Status Messages

Status messages are the static assignment of flags (bits) in the controller to plain text messages in the operating device. For status message addressing, use the data types ARRAY[1..N] OF BYTE or

ARRAY[1..N] OF WORD.

Data Type	Length of the Message System in Bytes
ARRAY OF BYTE	N
ARRAY OF WORD	N x 2

Fig. 7-10: Length of the message system in bytes

7.1.2.4 Date and Time

The variables for synchronizing the time and date must use the data type ARRAY [1..N] OF BYTE.

Variable	Length
Date with a 2-digit year	3 Bytes
Date with a 4-digit year	4 Bytes
Time	3 Bytes
Weekday	1 Byte

Fig. 7-11: Byte lengths for the date and time

7.1.2.5 Variant Buffer

The variable for the variant buffer must use the data type BYTE or USINT.

7.1.2.6 Tables

The variable for representation of tables must use the data type ARRAY [1..N]. The ARRAY [1..N] has to be of one of the following base data types:

- BOOL,
- BYTE,
- WORD,
- DWORD,
- SINT,
- INT,
- DINT,
- USINT,
- UINT,
- UDINT,
- REAL or
- STRING.

3S serial

7.1.3 Physical Connection

Plug-in connectors on the operating device for connection to the cont-roller.

7.1.3.1 Pin Assignment

Pin	Designation	Function
6	TD	Transmitted Data
15	CTS	Clear to send
17	RTS	Request to send
18	RD	Received data
25	SGND	Signal Ground

Fig. 7-12: Pin assignment RS232

Pin	Designation	Function
8	T(A)	Transmitted Data (-)
9	T(B)	Transmitted Data (+)
11	SGND	Signal Ground
22	R(A)	Received Data (-)
23	R(B)	Received Data (+)

Fig. 7-13: Pin assignment RS485

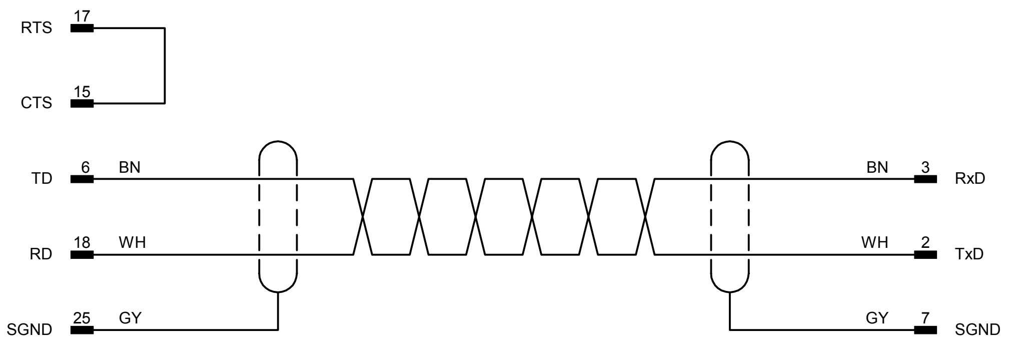
3S serial

7.1.3.2 Cable RS232 - Rexroth PPC-R

The following cabling diagram applies to operating devices with an universal interface **only**.

Operating device

Rexroth
PPC-R



D-SUB
male connector
25 pin

D-SUB
male connector
15 pin

Both ends of the shield are connected to the metallic housing.

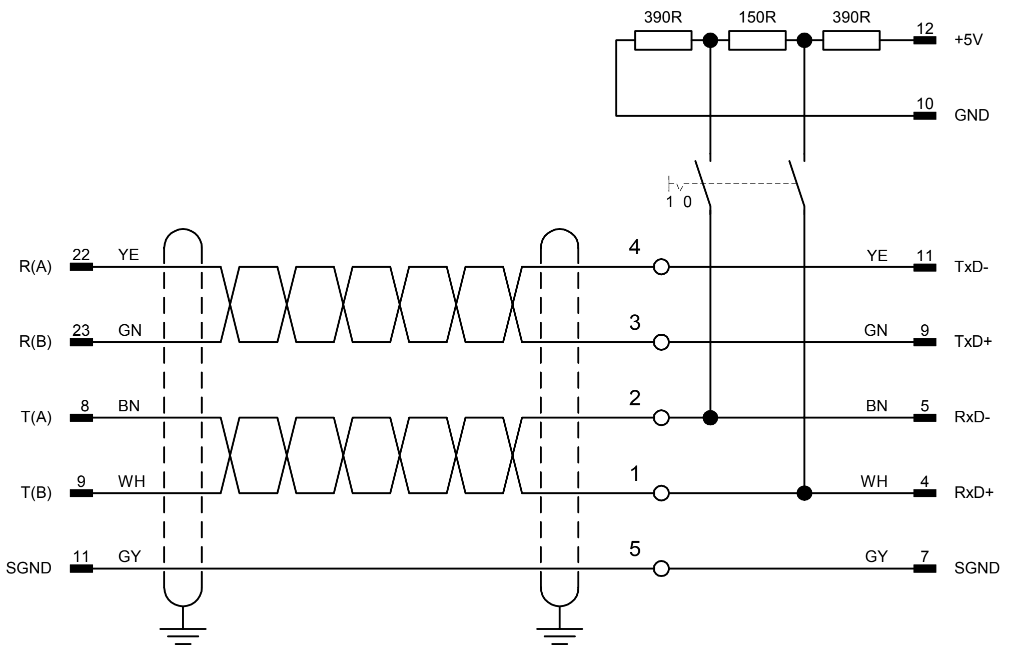
3S serial

7.1.3.3 Cable RS485 - Rexroth PPC-R

The following cabling diagram applies to operating devices with an universal interface **only**.

Operating device

Rexroth
PPC-R



D-SUB
male connector
25 pin

D-SUB
male connector
9 pin

Both ends of the shield are connected to the metallic housing.

7.1.4 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error

Code	XXXXX
Subcode	XXXXX
Retries	XXXXX

Code	Subcode	Error Type	Possible Cause
50	03	Framing error on serial interface	
	05	CRC error on serial interface	
	06	Parity error on serial interface	
60	10	Wrong telegram length	
	20	Wrong telegram Ident Number	
	30	Wrong block number	
	40	Wrong checksum	
	50	Negative acknowledgement	
	60	Waiting time exceeded: No response	Cable interruption, connection cut-off, wrong baud rate
70		Error from the controller	

Fig. 7-14: Error messages for 3S serial

3S serial

7.1.5 Applications

7.1.5.1 IndraLogic from version 1.0

The programming software takes the global variables from the symbol file project_name.SYM and inserts them into the variable list.

The symbolic names cannot be longer than 80 characters.

The entries in the variable list cannot be modified.

Declaring Global Variables

To declare global variables in IndraLogic:

1. Select **Auto Declare** from the **Edit** menu.

The **Declare Variable** dialog opens.

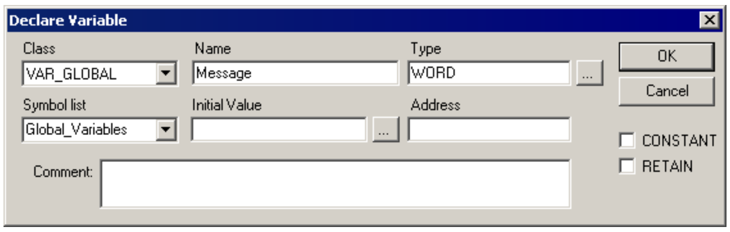
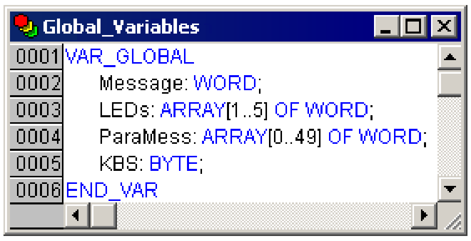


Fig. 7-15: Example of a variable declaration for global variables

2. Select the VAR_GOBAL class from the **Class** field.
3. Enter a name (Message) and a type (WORD).
4. Repeat step 3 for all additional global variables.
5. Click **OK** to confirm your input.

The **Global_Variables** window opens.



```
VAR_GLOBAL
    Message: WORD;
    LEDs: ARRAY[1..5] OF WORD;
    ParaMess: ARRAY[0..49] OF WORD;
    KBS: BYTE;
END_VAR
```

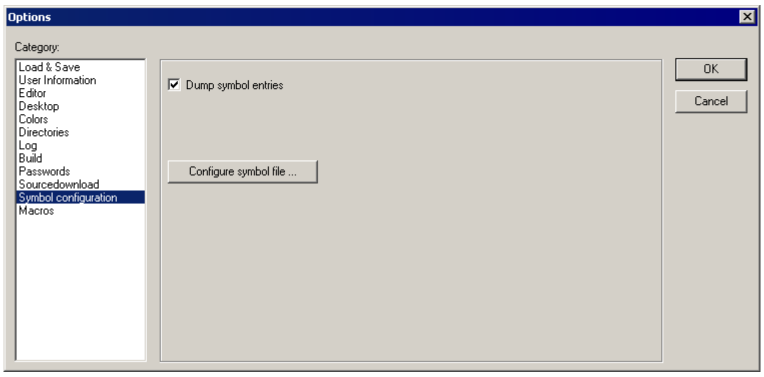
Fig. 7-16: Window Global variables

Activate Output into Symbol File

Specify the following settings in IndraLogic to write the global variables into a symbolic file.

1. Select **Options** from the **Project** menu.
2. Select **Symbol configuration**.

The **Options** dialog will look as follows:

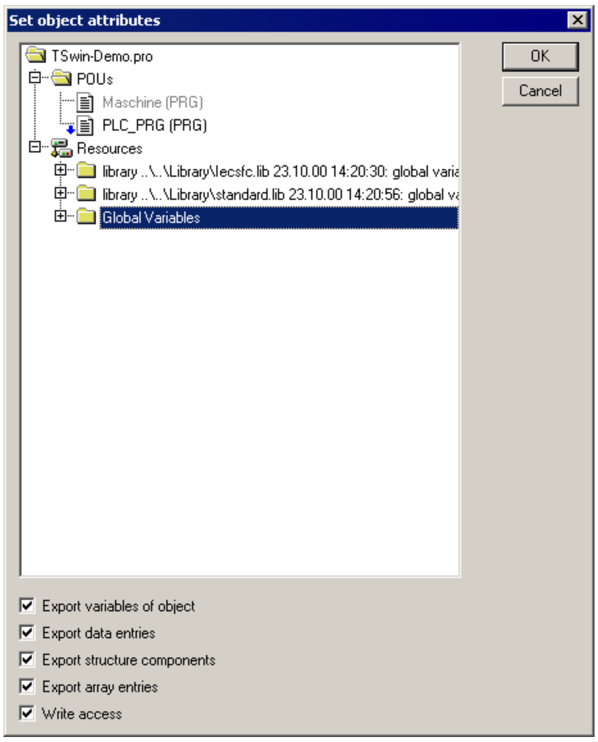


☑ Dump symbol entries

Fig. 7-17: Dialog Options - symbol configuration

3. Select the **Dump symbol entries** check box.
4. Click the **Configure symbol file** button.

The **Set object attributes** window opens.


Set object attributes
TSwin-Demo.pro
POUs
Maschine (PRG)
PLC_PRG (PRG)
Resources
library ..\..\Library\Iecsfc.lib 23.10.00 14:20:30: global varia
library ..\..\Library\standard.lib 23.10.00 14:20:56: global va
Global Variables
OK
Cancel

☑ Export variables of object
☑ Export data entries
☑ Export structure components
☑ Export array entries
☑ Write access

Fig. 7-18: Dialog Set object attributes

5. Select the **Global variables** entry.
6. Click **OK** to confirm your selection.

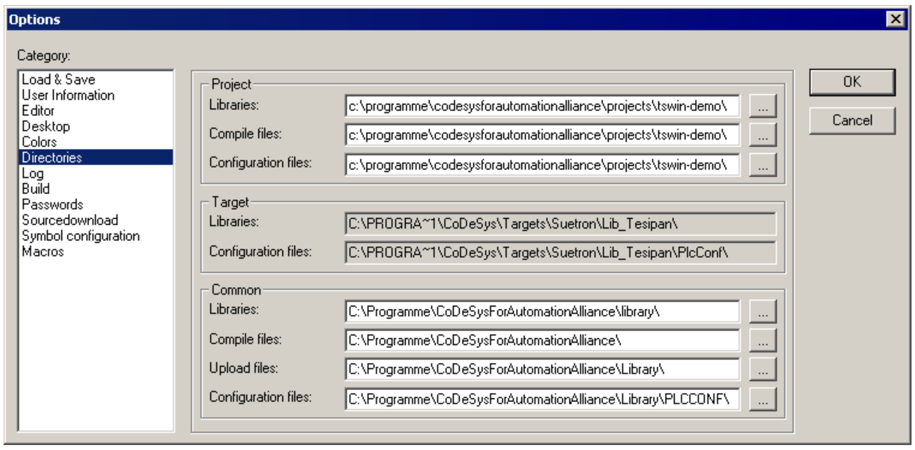
You are returned to the **Options** dialog.

Now you need to specify the position where the symbol file is to be stored.

3S serial

1. Select **Directories** from the **Options** dialog.

The **Options** dialog will look as follows:



Project
Libraries: c:\programme\codesysforautomationalliance\projects\tswin-demo\
Compile files: c:\programme\codesysforautomationalliance\projects\tswin-demo\
Configuration files: c:\programme\codesysforautomationalliance\projects\tswin-demo\

Target
Libraries: C:\PROGRA~1\CoDeSys\Targets\Suetron\Lib_Tesipan\
Configuration files: C:\PROGRA~1\CoDeSys\Targets\Suetron\Lib_Tesipan\PlcConf\

Common
Libraries: C:\Programme\CoDeSysForAutomationAlliance\library\
Compile files: C:\Programme\CoDeSysForAutomationAlliance\
Upload files: C:\Programme\CoDeSysForAutomationAlliance\Library\
Configuration files: C:\Programme\CoDeSysForAutomationAlliance\Library\PLCCONF\

Fig. 7-19: Dialog Options - directories

2. From the **Project** area, select a directory for the **compile files**.
3. Click **OK** to confirm your selection.

You are returned to the **Options** dialog.

The symbol file will not be created until a compilation process takes place and is stored in the same directory as the project!

Target System Settings

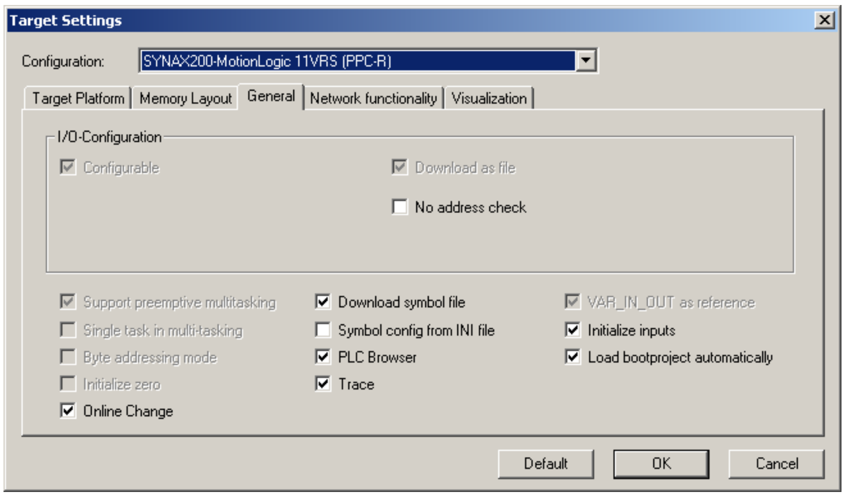
Select the following settings for the target system to ensure the symbol file is sent to the target system:

1. Open the **Resources** tab.
2. Double-click **Target settings**.

The **Target settings** dialog opens.

3. Open the **General** tab.
4. Select the **Download Symbol File** check box.

The **Target settings** dialog might look like the example below:


Target Settings

Configuration: SYNAX200-MotionLogic 11VRS (PPC-R)

Target Platform | Memory Layout | General | Network functionality | Visualization

I/O-Configuration
- ☑ Configurable
- ☑ Download as file
- ☐ No address check

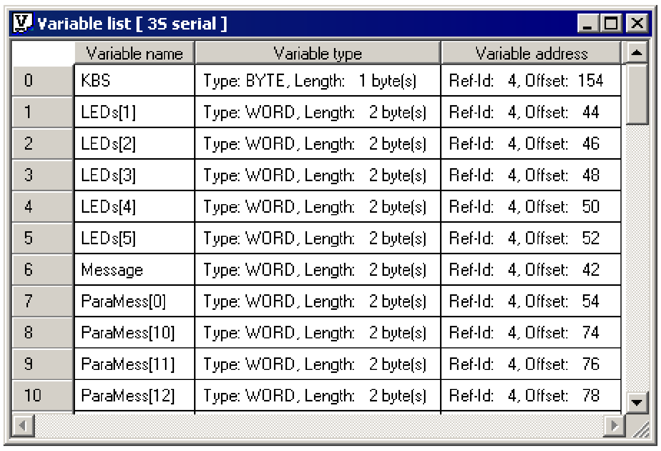
- ☑ Support preemptive multitasking
- ☐ Single task in multi-tasking
- ☐ Byte addressing mode
- ☐ Initialize zero
- ☑ Online Change
- ☑ Download symbol file
- ☐ Symbol config from INI file
- ☑ PLC Browser
- ☑ Trace
- ☑ VAR_IN_OUT as reference
- ☑ Initialize inputs
- ☑ Load bootproject automatically

Default | OK | Cancel

Fig. 7-20: Dialog Target settings

Variable List

The programming software automatically places the symbolic variable entries created in the example into the variable list if you specified the correct directory and name in the communications parameters.


Variable list [3S serial]

	Variable name	Variable type	Variable address
0	KBS	Type: BYTE, Length: 1 byte(s)	Ref-Id: 4, Offset: 154
1	LEDs[1]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 44
2	LEDs[2]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 46
3	LEDs[3]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 48
4	LEDs[4]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 50
5	LEDs[5]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 52
6	Message	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 42
7	ParaMess[0]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 54
8	ParaMess[10]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 74
9	ParaMess[11]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 76
10	ParaMess[12]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 78

Fig. 7-21: Variable list

This makes the variables globally available in the programming software.

7.2 Bosch BUEP19E

The Bosch BUEP19E protocol allows you:

- random read and write access to all PLC data
- bit-by-bit access to all byte, word and double-word oriented data types
- byte-by-byte access to all data words in a data block.

The size of the address area depends on the controller being used.

This protocol supports a connection to the following CPU modules:

- CL150
- CL200
- CL350
- CL400
- CL500
- CL550
- PCL

7.2.1 Data Types

Direct access is possible to the following data types.

The size of the individual data areas depends on the controller of the controller's CPU.

Type	Mnemonic	Access
Input Bit	BE	Bit Access
Input Byte	BYE	Byte Access
Input Word	WE	Word Access
Input Double-Word	DWE	Word Access
Output Bit	BA	Bit Access
Output Byte	BYA	Byte Access
Output Word	WA	Word Access
Output Double-Word	DWA	Word Access
Flag Bit	BM	Bit Access
Flag Byte	BYM	Byte Access
Flag Word	WM	Word Access
Flag Double-Word	DWM	Word Access

Fig. 7-22: Data types for Bosch BUEP19E

Type	Mnemonic	Access
Timer Word	WT	Word Access 0 to 127
Counter Word	WZ	Word Access 0 to 127
Data Buffer Byte	BYDP	Byte Access 0 to 511
Data Buffer Word	WDP	Word Access 0 to 511
Data Block Byte	DBxBYD	Byte Access 0 to 511
Data Block Word	DBxWD	Word Access 0 to 511
Data Block Double-Word	DBxDWD	Word Access
Data Field Byte	BLxBYDF	Byte Access 0 to 24575
Data Field Word	BLxWDF	Word Access 0 to 24575
Data Field Double-Word	BLxDWDF	Word Access 0 to 24575

Fig. 7-22: Data types for Bosch BUEP19E

Counter:

When a counter address is accessed, the counter value is interpreted in binary form. The maximum counter value is 8191.

Timer:

Timer values are made up of a time value and a time base.

The operating device reads the 2-byte variable and converts it internally into an imaginary, unsigned 4-byte variable, that represents the time value in reference for the base 0.01 seconds.

Before the operating device writes a timer value to the controller, it converts the unsigned 4-byte variable back into a 2-byte variable with a time value for the smallest possible time base.

Data field:

If you defined the data field as a linear area, the data field number must be set to the value 255.

7.2.2 Programming

7.2.2.1 Protocol parameters

With the protocol parameters, you can adapt the communication of the controller used.

Baud Rate

This parameter specifies the communication rate.

Configurable Values (Baud)	Default Value
300	
600	
1200	
2400	
4800	
9600	
19200	X
38400	
57600	
76800	
115200	

Fig. 7-23: Baud rate

Parity

This parameter specifies the parity used to control the communication.

Configurable Values	Default Value
None	
Even	X
Odd	

Fig. 7-24: Parity

Handshake

This parameter specifies the method used to control the communication.

Configurable values	Default Value
No Handshake	X
Hardware	
Software	

Fig. 7-25: Handshake

Data Bits

This parameter specifies the number of data bits.

Configurable Values	Default Value
5	
6	
7	
8	X

Fig. 7-26: Data bits

Stop Bits

This parameter specifies the number of stop bits.

Configurable Values	Default Value
1	X
1.5	
2	

Fig. 7-27: Stop bits

Use Coordination Flag

This parameter specifies whether you are using a coordination flag for the communication.

Configurable Values	Default Value
OFF	X
ON	

Fig. 7-28: Use coordination flag

Coordination Flag

Configurable Values	Default Value
0 to 255	0

Fig. 7-29: Coordination flag

Process Coordination Flag

This parameter specifies the number of the process coordination flag.

Configurable Values	Default Value
0 (System stop state)	X
1 (System RUN state)	
2 (I/O status)	
3 (I/O status or STOP	
4 (PE)	
5 (PE or STOP)	
6 (OB1)	
7 (OB1 or STOP)	
15 (no process coor-dination)	

Fig. 7-30: Process Coordination Flag

Destination Module

This parameter specifies the CPU module you are using.

Configurable Values	Default Value
CL500	X
CL350/CL400	
CL150/CL200/ CL550/PCL	

Fig. 7-31: Destination Module

Block Check

This parameter specifies the block check to be performed for the communication.

Configurable Values	Default Value
CRC16	
LRC8	X

Fig. 7-32: Block Check

7.2.2.2 Input Syntax

The following image illustrates the structure of the input syntax for variables in the programming software.

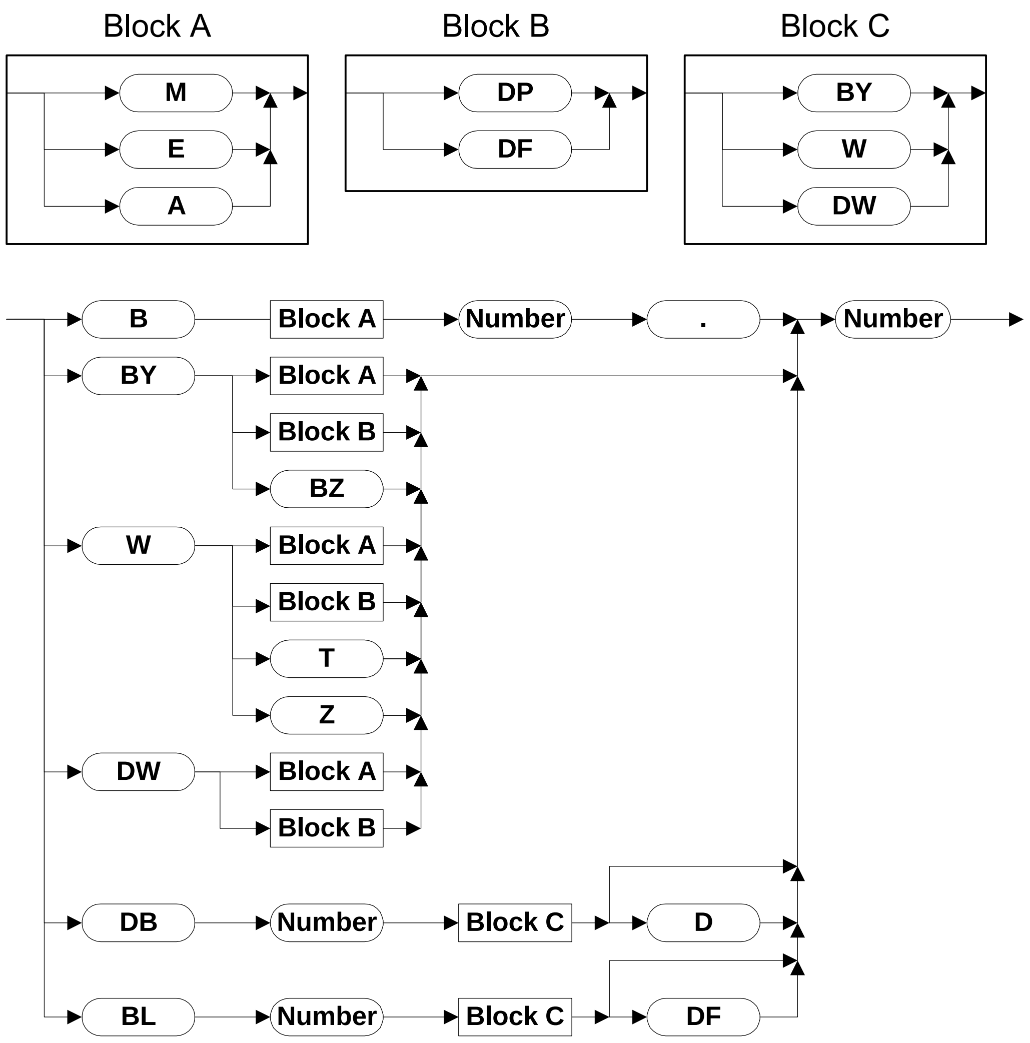


Fig. 7-33: Syntax diagram

Bosch BUEP19E

7.2.3 Physical Connection

Steckverbindungen am Bediengerät für den Anschluss an die PG-Schnittstelle der Bosch Steuerung.

7.2.3.1 Pin Assignment

Pin	Designation	Function
6	TD	Transmitted Data
15	CTS	Clear to send
17	RTS	Request to send
18	RD	Received data
25	SGND	Signal Ground

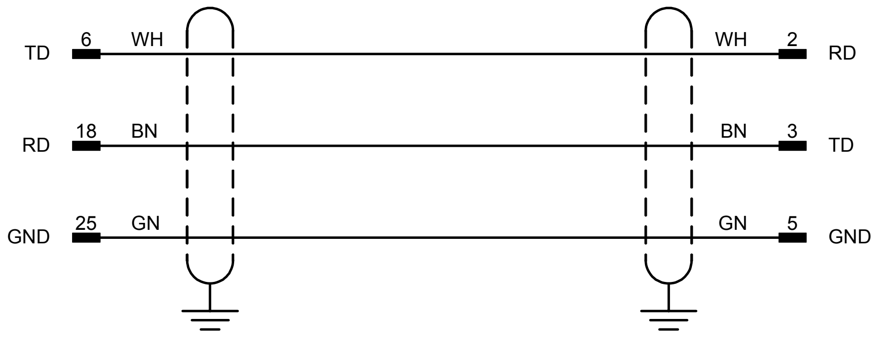
Fig. 7-34: Pin assignment RS232

7.2.3.2 Cable SER1 RS232 - Bosch CL150/CL151

The following cabling diagram applies to operating devices with an universal interface **only**.

Operating Device

Bosch
CL150/CL151



D-SUB
Male Connector
25 Pin

D-SUB
Male Connector
9 Pin

Both ends of the shield are connected to the metallic housing.

Bosch BUEP19E

7.2.4 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error

Code	XXXXX
Subcode	XXXXX
Retries	XXXXX

Code	Subcode	Description	Possible Cause
1	1	Slave not ready	Wrong baud rate or cable defective
	3	Error in protocol frame	
	5	CRC error	
	6	Wrong parity	
	10	No cyclic data defined	
	16	Receive buffer overrun	
Bosch-Specific Error Messages			
1	50	No connection setup	
	51	Wrong acknowledgment during connection setup	
	52	Wrong acknowledgment after sending information block	Wrong block check set, PG uses LRC8. The first peripheral participant determines the block check used !
	53	No response telegram	
	54	Timeout - No response telegram	
	55	Block time exceeded	
	56	No acknowledgment	
	57	EOT - Aborted by controller	
2	58	Incorrect number of data received	Check if the screen, in which the error occurred, contains a variable with an odd number of bytes which accesses a word address or a double-word address.

Fig. 7-35: Error Messages for Bosch BUEP19E

Bosch BUEP19E

Code	**Subcode**	**Description**	**Possible Cause**
Error from Programmable Controller			
3	1	Addressed module does not exist	
	16	Module can not be addressed	
	35	The address field has been protected by the user	
	36	Access to this address field is not permitted	
	37	Writing to timer is not permitted	
	38	Block number too large	
	39	Block does not exist	
	40	Block too small	
	147	Flag area (CL200 only) exceeded	Flag area defined is outside of BYM0 to BYM191
4	32	Addressed data type (command code) unknown in PST (peripheral station)	
	33	Protocol code unknown in PST	
	35	Specified coordination flag unknown in PST	
	37	Parameter code in telegram and specified parameters do not match	
	38	Block length and actual number of data do not match	
	40	Telegram type unknown	
	41	Command type unknown	
	58	Starting address and operand type do not match (word at odd address)	Defective R500 module possible
	59	Starting address outside of specified address range	
	60	Invalid parameter for specified command	
	61	Invalid operand type	
	64	PST has not received an identification telegram	
	99	Specified data length greater than addressed data area	
	210	Coordination flag is disabled	
Error from Operating Device			
40		System variable error	Undefined system variable

Fig. 7-35: Error Messages for Bosch BUEP19E

7.3 BRC-Symbolic

The protocol provides random read and write access to all global data objects of the controller.

The programming software adopts the data objects of the symbol file (file_name.SYM) which are created when the controller project is compiled.

The connected operating device uses the symbolic name to access a data object.

The device data base (GSD) file RX02081A.GSD can be used to set the operating devices' parameters in the PLC software for the PROFIBUS. After a standard installation of the programming software, this file is available in the following path: **C:\program files\rexroth\VI-Composer\FBs\PB\TYP_GSD** or in our Internet download area.

7.3.1 Data Objects

The length of a variable is determined by the length defined in the programming software IndraLogic.

7.3.1.1 Single Variables

You can access variables of the following type: BOOL, SINT, INT, DINT, BYTE, USINT, WORD, UINT, DWORD, UDINT, REAL, STRING, LINT, ULINT, LREAL and BITORBYTE.

7.3.1.2 String Variables

For string variables, the variable type STRING(N) is used, where N is the length of the string.

String variables can not be longer than 64 characters.

7.3.1.3 Tables

If ARRAY variables are used in table fields, the data type ARRAY [1..N] must be used. ARRAY [1..N] must be of one of the following base types:

- BOOL
- BYTE
- WORD
- DWORD
- SINT

- INT
- DINT
- USINT
- UINT
- UDINT
- LINT
- ULINT
- LREAL
- REAL or
- STRING.

7.3.2 Programming

7.3.2.1 Connection Settings

Transport Layer

Select the physical connection option.

Configurable Values	**Default Value**
SIS serial	X
PROFIBUS-DP	
Ethernet	

Fig. 7-36: Transport layer

Connection Name

You can define symbolic names for up to 16 connections. These connection names are used as SIS protocol connection settings.

Variable List

This parameter specifies the directory in which the variable list *.sym is stored.

To select a directory, click the Browse button.

The variable list *.sym is created by the programming software IndraLogic when compilation takes place.

From this file, the programming software reads all entries whose symbolic name is shorter than 80 characters.

Controller Address (optional)

Enter a numerical 15 digit value for the controller address.

Axis Address (optional)

Enter a numerical 3 digit value for the axis address.

7.3.2.2 Protocol Parameters for SIS

Connection Name

The names which you entered into the connection settings are used as connection names.

Address

This parameter specifies the SIS address of the communication partici-pant.

Configurable Values	**Default Value**
1 to 126	1

Fig. 7-37: Address

Controllers

This parameter indicates the IndraLogic runtime system in the control.

Configurable Values	**Default Value**
SIS-Master	X
SIS-Multi-Master	

Fig. 7-38: Controller

Duplex Operation

Select this parameter when using an RS232/RS422/RS485 interface.

The parameter Half Duplex applies only in conjunction with the RS485 interface.

Configurable Values	**Default Value**
Full Duplex (RS232, RS422)	X
Half Duplex (RS485/2-Draht)	

Fig. 7-39: Duplex operation

Device Address

This parameter specifies the SIS address of the operating device.

Configurable Values	**Default Value**
1 to 126	1

Fig. 7-40: Device address

Baud Rate

This parameter specifies the communication rate.

Configurable Values (Baud)	**Default Value**
4800	
9600	
19200	
38400	X
46875	

Fig. 7-41: Baud Rate

Parity

This parameter specifies the parity used to control the communication.

Configurable Values	Default Value
None	
Even	X
Odd	

Fig. 7-42: Parity

Handshake

This parameter specifies the method used to control the communication.

Configurable values	Default Value
No Handshake	X
Hardware	
Software	

Fig. 7-43: Handshake

Data Bits

This parameter specifies the number of data bits.

Configurable Values	Default Value
8	X

Fig. 7-44: Data bits

Stop Bits

This parameter specifies the number of stop bits.

Configurable Values	Default Value
1	X

Fig. 7-45: Stop bits

Delay Until Connection Set-up

This parameter specifies the waiting time after which the operating device starts the communication.

Configurable Values	Default Value
5 s to 255 s	5 s

Fig. 7-46: Delay until connection set-up

Response Time Slave

This parameter specifies how long the operating device waits for a response from the controller.

Configurable Values	Default Value
100 ms to 25500 ms	1000 ms

Fig. 7-47: Response time - slave

Bus Idle Time

This parameter specifies the waiting time between sending and recei-ving.Applies only if the interface is operated in the half duplex mode.

Configurable Values	Default Value
0 ms to 65535 ms	5 ms

Fig. 7-48: Bus idle time

Repetitions

This parameter specifies how often the communication is repeated after an error occurred.

Configurable Values	Default Value
0 to 5	3

Fig. 7-49: Repetitions

7.3.2.3 Protocol Parameters for PROFIBUS-DP

Device Address

This parameter specifies the SIS address of the operating device.

Configurable Values	Default Value
1 to 126	2

Fig. 7-50: Device address

Delay Until Connection Set-up

This parameter specifies the waiting time after which the operating device starts the communication.

Configurable Values	Default Value
5 s to 255 s	5 s

Fig. 7-51: Delay until connection set-up

Response Time Slave

This parameter specifies how long the operating device waits for a response from the controller.

Configurable Values	Default Value
10 ms to 65535 ms	1000 ms

Fig. 7-52: Response time - slave

7.3.2.4 Protocol Parameters for Ethernet

Connection Table

In the connections table you define connections for up to 16 controllers.

In column **connection name** the connection names of the main dialog are adopted.

Enter a valid IP address to each connection name into the column **IP - address of control**.

7.3.2.5 Polling Area

The poll area is used to manage the write coordination byte, the serial message channel and the LEDs in the function keys. This area is continuously polled by the operating device.

This protocol requires you to set up the poll area with three single variables. The variables must be part of the same connection!

The data types BYTE, USINT, WORD or UINT must be used to address the poll area.

7.3.2.6 Status Messages

Status messages are the static assignment of flags (bits) in the controller to plain text messages in the operating device. For status message addressing, use the data types BYTE, USINT, WORD, UINT, DWORD, UDINT, or ARRAY[1..N]. The following applies when using ARRAY: The type size multiplied by N provides the size of the message system in bytes.

7.3.2.7 Date & Time

Use the data type ARRAY[1..N]_OF_UINT8 to transfer the time and date; where N must correspond to the number of bytes to be transferred. If a 2-digit year format is used, you need to transfer 7 bytes; for a 4-digit year format, 8 bytes must be transferred.

7.3.3 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error

Code	XXXXX
Subcode	XXXXX
Retries	XXXXX

Code	Subcode	Description	Possible Cause
40		Illegal system variable	The system variable is not supported by this operating device.
50	03	Framing error on serial interface	
	06	Parity error on serial interface	
	10	Poll area error	No poll area defined
	12	Poll area error	Poll area defined more than once
	16	Memory overrun	
	50	Memory allocation not possible	No memory allocated
51		SPC3 hardware defect	
	1	Wrong initialization	Inputs/outputs not correctly initialized
	2	Wrong initialization	
	4	Wrong initialization	Memory not correctly initia-lized
	16	No hardware / no SPC3	Interface hardware missing or defective
52		SPC3 not configured properly	
	1	Physical connection error	Cable defective or not con-nected
	2	Logical connection error	Wrong participant number
	3	Wrong input length	
	4	Wrong output length	
	5	Invalid configuration	
54		SPC3 disconnects from bus	
55		Internal SPC3 error	

Fig. 7-53: Error messages, BRC-Symbolic

BRC-Symbolic

Code	Subcode	Description	Possible Cause
56		Waiting time in SPC3 exceeded; no more telegrams received from master	
60	40	Wrong checksum	
	60	Waiting time exceeded: No response	Cable interruption, connection cutoff, wrong baud rate
	70	Transmission buffer too small	
80		Field bus error PROFIBUS	
	10	Ready bit in status byte of master not set	
80		Field bus error Ethernet	
	10	Response telegram error	Wrong start token in data module
	11	Error while establishing the ethernet connection to a server	
	12	Error sending a telegram	
	13	Timeout, no response from control	
90		Response with length 0 received...	
	21	...while reading the error number	
91 to 110	1	Error in response telegram	
	2	Error in response telegram	
120			
	1	Telegram contains errors	
	2	Variable does not exist	
	3	Illegal process	
	4	User not logged in	
	5	Value can not be converted as required by variable type	
	6	Handle was expected	
	7	No memory	

Fig. 7-53: Error messages, BRC-Symbolic

7.4 DeviceNet

The operating device is incorporated into the DeviceNet network as a DeviceNet slave. The communication between a controller (master or scanner) and the operating device (slave) is based on the Predefined Master/Slave Connection Set.

Explicit Message Connections and Poll I/O Connections are used as Connection Instances.

7.4.1 Explicit Message

Explicit Messages are used to exchange data between the operating device and the controller. This requires you to create a function block in the controller which assembles the payload into Explicit Messages.

7.4.1.1 Storing Data

All data displayed by the operating device are stored in the operating device’s data memory.

The size of the data memory is 2500 words.

The data memory is word-oriented. The addresses are always word addresses, both from the operating device's and the controller's pers-pective.

7.4.1.2 Exchanging Data

You need to create a program (function block) in the controller which is used to establish a data exchange between the operating device and the controller by means of Explicit Messages. For this, make sure that the data on both devices are consistent.

You can carry out the data exchange cyclically or carry it out event-controlled through the controller using the I/O Poll Telegram.

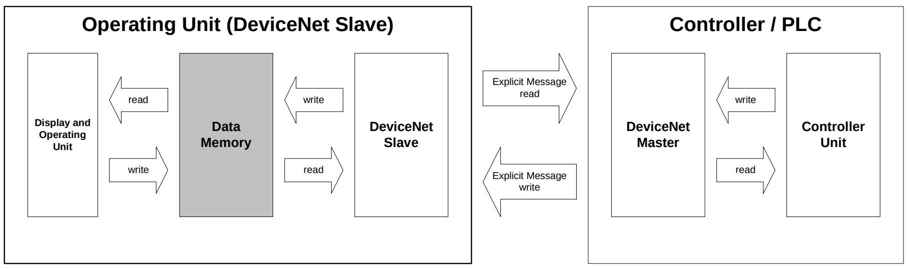


Fig. 7-54: Data exchange, DeviceNet

7.4.1.3 Data Memory

The data memory in the operating device is referred to as the Memory Object.

The Memory Object Address has the following values.

Address	**Designation**	**Comment**
0x8A	Class ID	
0x01	Instance ID	
0x01	Attribute	Not required

Fig. 7-55: Memory object addresses

The service

- for a read access is 0x33 and
- for a write access is 0x35.

7.4.1.4 Read Service

The following table illustrates the structure of the Explicit Message for the Read service. Each field of the telegram is one byte long.

Byte	**Request Telegram**	**Response Telegram**
1	MAC ID	MAC ID
2	Service ID 0x33	Service ID 0xB3
3	Class ID 0x8A	1st Data Word Low Byte
4	Instance ID 0x01	1st Data Word High Byte
5	Word Address Low Byte	2nd Data Word Low Byte
6	Word Address High Byte	2nd Data Word High Byte
7	Number of Bytes Low Byte	
8	Number of Bytes High Byte	

Fig. 7-56: Structure of the Explicit Message for the Read service

The word address corresponds to the offset within the data memory in the operating device.

The byte order for the

- word address,
- number of bytes and
- data word

can be specified in the communication parameters.

☞ See chapter „Byte Order“ on page 7-52.

7.4.1.5 Write Service

The following table illustrates the structure of the Explicit Message for the Write service. Each field of the telegram is one byte long.

Byte	Request Telegram	Response Telegram
1	MAC ID	MAC ID
2	Service ID 0x35	Service ID 0xB5
3	Class ID 0x8A	
4	Instance ID 0x01	
5	Word Address Low Byte	
6	Word Address High Byte	
7	1st Data Word Low Byte	
8	1st Data Word High Byte	
9	2nd Data Word Low Byte	
10	2nd Data Word High Byte	

Fig. 7-57: Structure of the Explicit Message for the Write service

The word address corresponds to the offset within the data memory in the operating device.

The byte order for the

- word address,
- number of bytes and
- data word

can be specified in the communication parameters.

☞ See chapter „Byte Order“ on page 7-52.

7.4.1.6 Fragmentation

By means of fragmentation, up to 384 bytes can be transferred in one explicit message.

7.4.2 Poll I/O Connection

On account of the EDS data, the Poll I/O Connection is installed automatically between the DeviceNet master and the operating device. With this connection, 2 bytes are transferred cyclically from the controller’s OUT area to the operating device and 5 bytes are transferred from the operating device to the IN area of the controller.

7.4.2.1 Receive Data of the Operating Device (Consumed Data)

The Consumed Connection Size is 2 bytes.

Byte	Designation
1	Initialization
2	Control Byte

Fig. 7-58: Structure of the Consumed Data

7.4.2.2 Transmit Data of the Operating Device (Produced Data)

The Produced Connection size is 5 bytes.

Byte	Designation
1	Initialization
2	Control Byte
3	Word Address - Low Byte
4	Word Address - High Byte
5	Number of Altered Bytes

Fig. 7-59: Structure of the Produced Data

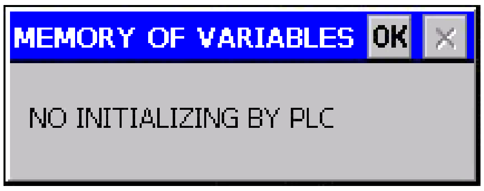
Byte 1 - Initialization

During the boot process, the operating device writes the value 0x00 into its Produced Data. Thus, the controller needs to initialize the entire data memory in the operating device once.

Initialization means that all variable values in the data memory of the operating device are set the same as in the controller. When this process is completed, the controller writes the value 0xC3 into the Consumed Data.

The operating device acknowledges by writing the value 0xC3 into the Produced Data.

After the boot process, the operating device waits for the initialization to complete, before it accesses the data memory. If the initialization is not performed within the time period set for the Delay until Connection Setup, the following message appears.


MEMORY OF VARIABLES

NO INITIALIZING BY PLC

Fig. 7-60: Message NO INITIALIZATION BY PLC

To remove the message, press **OK**, the **Help** key or a key that calls up another screen.

Byte 2 - Control Byte

During initialization, the control byte is set to the value 0x00.

Bit 0:

Bit 0 is the toggle bit.

After you change a variable on the operating device, bit 0 in the Produced Data toggles.

If the controller detects that bit 0 in the Produced Data and bit 0 in the Consumed Data do not match, the controller has to read the changed data of the variable by „Explicit Message“.

Once the read operation is complete, the controller sets the value of bit 0 in the Consumed Data to the same value as that of bit 0 in the Produced Data.

If the controller detects that bit 0 in the Produced Data and bit 0 in the Consumed Data differ, bit 0 in the Consumed Data must be set to the same value as that of bit 0 in the Produced Data!

If no bit synchronization takes place, the operating device displays the error message Timeout Error: Code 60, Subcode1“.

Bit 1 to bit 7

Bits 1 to 7 are reserved.

Byte 3 and Byte 4 - Word Address

Bytes 3 and 4 in the Produced Data contain the word address starting from which a variable has changed. The variable can be several bytes long. The controller uses this word address to selectively read the changed variable from the data memory of the operating device.

Byte 5 - Number of Bytes

Byte 5 in the Produced Data contains the number of bytes as the size information for the changed variable.

7.4.2.3 Module/Network Status

The operating devices are not equipped with diagnostic LEDs for DeviceNet status indication.

To indicate the module/network status, use the system variable **Com-BaudrateA** instead.

Siehe Kapitel „ComBaudrateA“ auf Seite 7-53.

7.4.3 Object Definitions

Class ID	Object Name
0x01	Identity Object
0x02	Message Router (not supported)
0x03	DeviceNet Object
0x04	Assembly Object (not supported)
0x05	Connection Object
0x8A	VCP Object

Fig. 7-61: Object definitions

7.4.3.1 Identity Object

Class Attribute of Identity Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x01	Revision	Get	2 (6-2.2)	0x01

Fig. 7-62: Class Attribute of Identity Object

Instance Attribute of the Identity Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x01	Vendor ID	Get	2 (6-2.2)	0x238
0x02	Device Type	Get	2	0x00
0x03	Product Code	Get	2	0x01
0x04	Revision	Get	2	0x0101 (1.001)
0x05	Status	Get	2	
0x06	Serial Number	Get	4	0x01
0x07	Product Name	Get	32	„DeviceNet for VCP-Serie“

Fig. 7-63: Instance Attribute of Identity Object

Instance Service of the Identity Object

Service ID	Service Name	Description
0x0E	Get_Attribute_Single	Returns the Contents of the Specific Attribute
0x05	Reset	Invoke the Reset Service in the VCP

Fig. 7-64: Instance Service of Identity Object

Message Router Object

This object is not supported.

7.4.3.2 DeviceNet Object

Class Attribute of the DeviceNet Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x01	Revision ID	Get	1 (5-5.3)	0x02

Fig. 7-65: Class Attribute of the DeviceNet Object

Instance Attribute of the DeviceNet Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x01	MAC ID	Get	1 (5-5.3)	
0x02	BaudRate	Get	1 (5-5.3)	
0x05	Allocation Information	Get	1 (5-5.3)	

Fig. 7-66: Instance Attribute of the DeviceNet Object

Instance Service of the DeviceNet Object

Service ID	Service Name	Description
0x0E	Get_Attribute_Single	Read attribute
0x4B	Allocation Master/Slave Connection Set	Request to use predefined master/slave connection set
0x4C	Release Group 2 Identifier Set	Release desired connection

Fig. 7-67: Instance Service of the DeviceNet Object

7.4.3.3 Assembly Object

Class Attribute of the Assembly Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x01	Revision	Get	1	0x02

Fig. 7-68: Class Attribute of the Assembly Object

Instance 1 Attribute - Static Input - of the Assembly Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x03	Set Data	Get	5	

Fig. 7-69: Instance 1 Attribute - Static Input - of the Assembly Object

Instance 2 Attribute - Static Output - of the Assembly Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Attribute Value
0x03	Set Data	Get	2	

Fig. 7-70: Instance 2 Attribute - Static Output - of the Assembly Object

7.4.3.4 Connection Object

Class Attribute of the Connection Object

Service ID	Service Name	Description
0x0E	Get_Attribute_Single	Read Attribute

Fig. 7-71: Class Service of the Connection Object

Instance Attribute of the Connection Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Value Expl.	Value IO-Poll
0x01	State	Get	1		
0x02	Instance Type	Get	1	0	1
0x03	TransportClass_Trigger	Get	1	0x38	0x82
0x04	Produced Connection ID	Get	2		
0x05	Consumer Connection ID	Get	2		
0x06	Initial Comm Characteristics	Get	1	0x21	0x01
0x07	Produced Connection Size	Get	2	0xFFFF	5
0x08	Consumed Connection Size	Get	2	0xFFFF	2
0x09	Expected Packed Rate	Get/Set	2	Default	Default
0x0C	Watchdog Timeout Action	Get/Set	1	Default	Default

Fig. 7-72: Instance Attribute of the Connection Object

Attribute ID	Attribute Name	Access Rule	Data Size (Byte)	Value Expl.	Value IO-Poll
0x0D	Produced Connection Path Length	Get	2	0	6
0x0E	Produced Connection Length	Get	6	0	20 04 24 01 30 03
0x0F	Consumed Connection Path Length	Get	2	0	6
0x10	Consumed Connection Path	Get	6	0	20 04 24 02 30 03

Fig. 7-72: Instance Attribute of the Connection Object

Instance Service of the Connection Object

Service ID	Service Name	Description
0x0E	Get_Attribute_Single	Read Attribute
0x10	Set_Attribute_Single	Write Attribute

Fig. 7-73: Instance Service of the Connection Object

7.4.3.5 VCP Object

Instance Service of the VCP Object

Service ID	Service Name	Description
0x33	Block StringN Read	Read Data by Each Data Unit
0x35	Block StringN Write	Write Data by Each Data Unit

Fig. 7-74: Instance Service of the VCP Object

Service ID = 0x33 (Block StringN Read)

Byte	Designation
1	MAC ID
2	Service ID (0x33)
3	Class ID (0x8A)
4	Instance ID (0x01)
5	Word Address Low Byte *

Fig. 7-75: Request without Attribute parameter

Byte	**Designation**
6	Word Address High Byte *
7	Word Number of Bytes Low Byte *
8	Word Number of Bytes High Byte *

Fig. 7-75: Request without Attribute parameter

Byte	**Designation**
1	MAC ID
2	Service ID (0xB3)
3	1st Data Word Low Byte **
4	1st Data Word High Byte **
5	2nd Data Word Low Byte **
6	2nd Data Word High Byte **

Fig. 7-76: Response without Attribute parameter

Byte	**Designation**
1	MAC ID
2	Service ID (0x33)
3	Class ID (0x8A)
4	Instance ID (0x01)
5	Attribute (0x01)
6	Word Address Low Byte *
7	Word Address High Byte *
8	Word Number of Bytes Low Byte *
9	Word Number of Bytes High Byte *

Fig. 7-77: Request with Attribute parameter

Byte	**Designation**
1	MAC ID
2	Service ID (0xB3)
3	1st Data Word Low Byte **
4	1st Data Word High Byte **
5	2nd Data Word Low Byte **
6	2nd Data Word High Byte **

Fig. 7-78: Response with Attribute parameter

Service ID = 0x35 (Block StringN Write)

Byte	Designation
1	MAC ID
2	Service ID (0x35)
3	Class ID (0x8A)
4	Instance ID (0x01)
5	Word Address Low Byte *
6	Word Address High Byte *
7	1st Data Word Low Byte **
8	1st Data Word High Byte **
9	2nd Data Word Low Byte **
10	2nd Data Word High Byte **

Fig. 7-79: Request without Attribute parameter

Byte	Designation
1	MAC ID
2	Service ID (0xB5)

Fig. 7-80: Response without Attribute parameter

Byte	Designation
1	MAC ID
2	Service ID (0x35)
3	Class ID (0x8A)
4	Instance ID (0x01)
5	Attribute (0x01)
6	Word Address Low Byte *
7	Word Address High Byte *
8	1st Data Word Low Byte **
9	1st Data Word High Byte **
10	2nd Data Word Low Byte **
11	2nd Data Word High Byte **

Fig. 7-81: Request with Attribute parameter

Byte	Designation
1	MAC ID
2	Service ID (0xB5)

Fig. 7-82: Response with Attribute parameter

* Depends on the protocol parameter Byte Order for Address/Length.

** Depends on the protocol parameter Byte Order for Data.

See chapter „Byte Order“ on page 7-52.

7.4.4 Format of the Explicit Message

Class ID (1 Byte)	**Service ID (1 Byte)**	**Instance ID (1 Byte)**	**Service Data**		**Service Name**
			Attribute ID (1 Byte)	**Data (n Byte)**	
0x01 (Identity Object)	0x0E (Get)	0x01	0x01	Vendor ID	Get Attribute Single Vendor ID
			0x02	Product Type	Get Attribute Single Product Type
			0x03	Product Code	Get Attribute Single Product Code
			0x04	Vendor Revision	Get Attribute Single Vendor Revision
			0x05	ID Status	Get Attribute Single ID Status
			0x06	Serial Number	Get Attribute Single Serial Number
			0x07	Product Name	Get Attribute Single Product Name
	0x05	0x01	N/A or 0x01		RESET
0x03 (Device-Net Object)	0x0E (Get)	0x01	0x01	MAC ID	Get Attribute Single MAC ID
			0x02	Baud Rate	Get Attribute Single Baud Rate
			0x05	Allocation Information	Get Attribute Single Allocation Information
0x04 (Assembly Object)	0x0E (Get)	0x64	IN DATA		IN DATA
	0x0E (Get)	0x65	OUT DATA		OUT DATA

Fig. 7-83: Format of the Explicit Message

Class ID (1 Byte)	Service ID (1 Byte)	Instance ID (1 Byte)	Service Data		Service Name
			Attribute ID (1 Byte)	Data (n Byte)	
0x05 (Connec-tion Ob-ject)	0x0E (Get)	0x01 (Expli-cit Message) 0x02 (Polled I/O)	0x01	State	Get Attribute Single State
	0x02	Instance Type	Get Attribute Single Instance Type		
	0x03	Transport Class Trig-ger	Get Attribute Single Transport Class Trig-ger		
	0x04	Produced Connection ID	Get Attribute Single Produced Connection ID		
	0x05	Consumed Connection ID	Get Attribute Single Consumed Connection ID		
	0x06	Initial Comm. Cha-racteristics	Get Attribute Single Initial Comm. Cha-racteristics		
	0x07	Produced Connection Size	Get Attribute Single Produced Connection Size		
	0x08	Consumed Connection Size	Get Attribute Single Consumed Connection Size		
	0x09	Expected Packet Rate	Get Attribute Single Packet Rate		
	0x0C	Watchdog Timeout Action	Get Attribute Single Watchdog Timeout Ac-tion		

Class ID (1 Byte)	**Service ID (1 Byte)**	**Instance ID (1 Byte)**	**Service Data**		**Service Name**
			Attribute ID (1 Byte)	**Data (n Byte)**	
0x8A (PT Object)	0x33	0x1	Service Data		Block String Read
	0x35	0x1	Service Data		Block String Write
	0x33	0x1	0x0	Data	Block String Read *
	0x35	0x1	0x0	Data	Block String Write *

Fig. 7-83: Format of the Explicit Message

* Alternative, depends on the communication parameter **Attribute**.

7.4.5 EDS File

The EDS file ensures that the Poll I/O Connection is automatically installed between the DeviceNet master and operating device.

7.4.6 Programming

7.4.6.1 Protocol Parameters

With the protocol parameters, you can adapt the communication of the controller used.

Baud Rate

This parameter specifies the communication rate.

Configurable Values (kBaud)	**Default Value**
125	X
250	
500	

Fig. 7-84: Baud rate

Node Number

Use the node number to set the MAC ID for the operating device.

Configurable Values	Default Value
0 to 63	0

Fig. 7-85: Node number

Delay until Connection Set-Up

Specify this value to set the period of time the operating device waits before it sends the first Duplicate MAC ID Check Request Message. Messages arriving before this time has elapsed are not evaluated.

Configurable Values	Default Value
5 s to 255 s	5 s

Fig. 7-86: Delay until connection set-up

Waiting Time for Response

Specify a waiting time for the Produced Data toggle bit monitoring.

Configurable Values	Default Value
0 ms, 50 ms to 65000 ms	500 ms

Fig. 7-87: Waiting time for response

Siehe Kapitel „Byte 2 - Control Byte“ auf Seite 7-39.

Attribute

The Attribute parameter is not required in the Explicit Message to access the Memory Object. Some controllers do, however, force and transfer the Attribute parameter to generate an Explicit Message.

In this case, select the check box: Explicit Message Contains the Parameter 'Attribute'.

Configurable Values	**Default Value**
OFF	
ON	X

Fig. 7-88: Explicit Message contains the Attribute parameter

Byte Order

The byte order for the DeviceNet protocol is Low-High.

See Appendix J in Volume 1 of the DeviceNet Specification.

The experience of some users has shown that some controllers do not use this byte order. In these cases, a byte-swap must be performed in the controller. For ease of programming, the user also has the option of swapping the byte order in the operating device.

Select two selection fields depending on the actual byte order:

Configurable Values	**Default Value**
Address/Length Low-High	X
Data Low-High	X
Address/Length High-Low	
Data High-Low	

Fig. 7-89: Byte order

7.4.6.2 Input Syntax

The following image illustrates the structure of the input syntax for variables in the programming software.

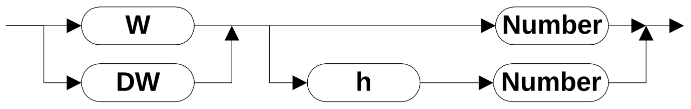


Fig. 7-90: Syntax diagram

7.4.6.3 Variables

The variable addresses specify an offset in the data memory of the operating device.

Variable Name	Address	Low Byte	High Byte
Word Access to Address 127	W 127		
Word Access to the Highest Address	W 2500		
Double-Word Access to Address 371	DW 371		
Double-Word Access to the Highest Address	DW 2499		
Bit Access to Bit 5 in Address 500	W 500	5	5
Bit Field Access to Bit 3 to Bit 12 in Address 1500	W 1500	3	12

Fig. 7-91: Addresses in the data memory of the operating device

7.4.6.4 System Variables

Since the operating devices do not have diagnostic LEDs, you need to use system variables to indicate specific DeviceNet statuses. You can create these system variables within any screen as output variables. To do so, select the representation type Selection Text and link the variable with a text list containing a text string for each status.

ComBaudrateA

This system variable can be used to indicate the statuses of the module/network LED.

Value	Text	Meaning
0	LED Off	
1	LED Green	Device is Assigned
2	LED Flashes Green	DUP_MAC_ID Test is Okay But a Connection is Not Established
3	LED Flashes Red	Connection Terminated After a Time Delay
4	LED Red	BUS-OFF Status
5	LED Flashes Red And Green	DUP_MAC_ID Error

Fig. 7-92: Statuses of the module/network LED

ComHandshakeA

This system variable allows you to indicate whether the data memory has been initialized by the master.

See chapter „Byte 1 - Initialization“ on page 7-38.

Value	Status
0	Not initialized
1	Initialized

Fig. 7-93: Initialization states of the data memory

7.4.7 Physical Connection

7.4.7.1 DeviceNet, 5 Pin Male Connector Strip

The connectors on the operating device and the cable connectors for the connection to the DeviceNet bus must comply with the DeviceNet connector profile.

Parameter	**Specification**
Male Connector Strip	
Number of Pins	5
Tension Nut	None
Rotation	None
Standards	Figure on Inquiry

Contact Assignment	nc = 1 (Not Connected)
	CAN_L = 2
	Drain = 3
	CAN_H = 4
	nc = 5 (Not Connected)

Female Connector Strip	
Number of Pins	5
Tension Nut	None
Rotation	None
Standards	Figure on Inquiry

Contact Assignment	nc = 1 (Not Connected)
	CAN_L = 2
	Drain = 3
	CAN_H = 4
	nc = 5 (Not Connected)

Physical Properties	
Contact Surface	30 Micro Inch Gold Minimum Over 50 Micro Inch Nickel Minimum or 5 Micro Inch Gold Minimum Over 20 Micro Inch Palladium-Nickel Minimum Over 50 Micro Inch Nickel, All Gold Must be 24 Karat
Lifetime of Contacts	100 Insertion-Extractions
Electrical Properties	
Operating Voltage	25 Volt Minimum
Current Carrying Capacity	8 Ampere Minimum

Fig. 7-94: 5 pin male connector strip profile for DeviceNet

Parameter	**Specification**
Contact Resistance	Nominal: Less than 1 mOhm Maximum: 5 mOhm Over Life
Ambient Conditions	
Water Resistance	None
Oil Resistance	Nne
Ambient Temperature Operation	–40 °C to +70 °C Full Power With Linear Derating to 0 Ampere at 80 °C
Ambient Temperature Storage	–40 °C to +85 °C

Fig. 7-94: 5 pin male connector strip profile for DeviceNet

Pin Assignment

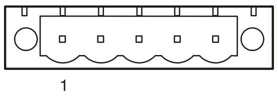


Fig. 7-95: 5 pin connector

Connector in the small operating terminal: 5 pin male connector strip.

Fig. 7-96: Pin assignment of DeviceNet interface

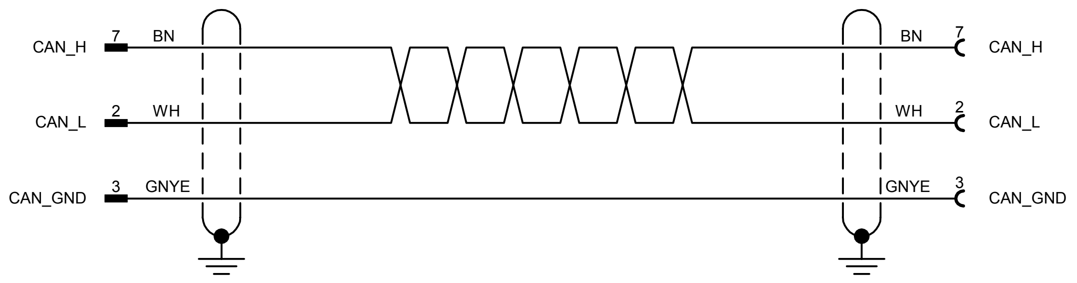
Pin	Designation	Function
1	nc	Not Connected
2	CAN_L	CAN_L Bus Line (Dominant LOW)
3	Drain	Shield
4	CAN_H	CAN_H Bus Line (Dominant HIGH)
5	nc	Not Connected

For the connection, use a 5 pin connector with gold-plated contacts and cover cap.

7.4.7.2 Cable DeviceNet, 9 Pin D-SUB

Operating device

Next DeviceNet participant



D-SUB
male connector
9 pin

D-SUB
female connector
9 pin

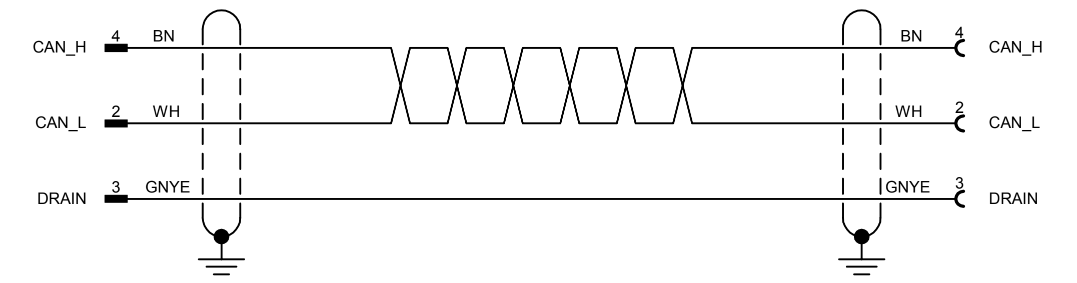
Both ends of the shield are connected to the metallic housing.

Contrary to the recommendations made in the CiA Draft Standard 102, the cable is only equipped with the wires needed to meet the current communication requirements.

7.4.7.3 Cable DeviceNet, 5 Pin Male Connector Strip

Operating device

Next DeviceNet participant



Male connector
5 pin

Female connector
5 pin

Both ends of the shield are connected to pin 3.

Contrary to the recommendations made in the CiA Draft Standard 102, the cable is only equipped with the wires needed to meet the current communication requirements.

7.4.8 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error

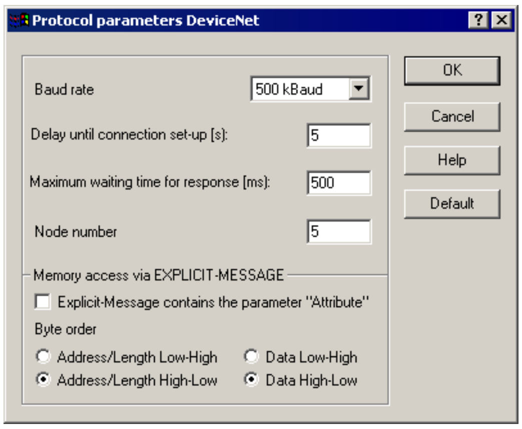
Code	XXXXX
Subcode	XXXXX
Retries	XXXXX

Code	Subcode	Error Type	Possible Cause
50		Hardware error	
	1	CAN controller error (stuff error)	
	2	CAN controller error (form error)	
	3	CAN controller error (acknowledge error)	Device is not connected to the bus.
	4	CAN controller error (bit 1 Error)	Short-circuit between the CAN_L and CAN_H line.
	5	CAN controller error (bit 0 error)	Short-circuit between the CAN_L and CAN_H line.
	6	Error from CAN controller (CRC error)	
60		Data exchange error	
	1	Toggle bit in the control byte is not processed by the controller or the DeviceNet master.	If the controller detects that the toggle bit in the Produced Data and the toggle bit in the Consumed Data differ, the toggle bit in the Consumed Data must be set to the same value as that of the toggle bit in the Produced Data.
	2	Changed data in the operating device is not read by controller	Although transferred by Poll I/O the controller does not read from the assigned address

Fig. 7-97: DeviceNet error messages

7.4.9 Applications

7.4.9.1 Rexroth PPC


Protocol parameters DeviceNet

Baud rate: 500 kBaud

Delay until connection set-up [s]: 5

Maximum waiting time for response [ms]: 500

Node number: 5

Memory access via EXPLICIT-MESSAGE

☐ Explicit-Message contains the parameter "Attribute"

Byte order

○ Address/Length Low-High ○ Data Low-High

● Address/Length High-Low ● Data High-Low

Fig. 7-98: Protocol parameters for Rexroth PPC

7.5 IndraLogic

The protocol provides random read and write access to all global data objects of the controller.

The programming software adopts the data objects of the project_name.SYM file which are created when the IndraLogic project is compiled.

The connected operating device uses the symbolic name to access a data object.

7.5.1 Data Types

The length of a variable is determined by the length defined in the programming software IndraLogic.

7.5.1.1 Single Variables

You can access variables of the following type: BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL, and STRING. Floating point numbers are interpreted in IEEE format. The variable type REAL is required for this purpose.

7.5.1.2 String Variables

For string variables, the variable type STRING(N) is used, where N is the length of the string.

7.5.2 Programming

7.5.2.1 Protocol Parameters

Baud Rate

This parameter specifies the communication rate.

Configurable Values (Baud)	**Default Value**
4800	
9600	
19200	
38400	X

Fig. 7-99: Baud rate

Data Bits

This parameter specifies the number of data bits.

Configurable Values	**Default Value**
5	
6	
7	
8	X

Fig. 7-100: Data bits

Stop Bits

This parameter specifies the number of stop bits.

Configurable Values	**Default Value**
1	X
1.5	
2	

Fig. 7-101: Stop bits

Parity

This parameter specifies the parity used to control the communication.

Configurable Values	Default Value
None	X
Even	
Odd	

Fig. 7-102: Parity

Maximum Waiting Time For Response

This parameter specifies how long the operating device waits for a response from the PLC.

Configurable Values	Default Value
100 ms to 25500 ms	1000 ms

Fig. 7-103: Waiting time for response

Delay until Connection Set-Up

This parameter specifies the waiting time after which the operating device starts the communication.

Configurable Values	Default Value
5 s to 255 s	5 s

Fig. 7-104: Delay until connection set-up

Byte Order

This parameter specifies the destination hardware's CPU type.

Configurable Values	Default Value
Intel	
Motorola	X

Fig. 7-105: Byte order

Controllers

This parameter specifies the runtime system of the controller.

Configurable Values	Default Value
Standard	X
PLCWinNT	

Fig. 7-106: Controllers

Path for Variable List *.sym

This parameter specifies the directory in which the variable list *.sym is stored.

To select a directory, click the Browse button.

The variable list *.sym is created by the programming software Indra-Logic when compilation takes place.

7.5.2.2 Polling Area

The poll area is used to manage the write coordination byte (WCB), the serial message channel and the LEDs in the function keys. This area is continuously polled by the operating device.

This protocol requires you to set up the poll area with three single vari-ables.

Area	Valid Data Types
KBS (write coordination byte)	BYTE, USINT, WORD, UINT
Message Channel	WORD, UINT
Function Key LEDs	BYTE, USINT, WORD, UINT, DWORD, UDINT, ARRAY[1..N]

Fig. 7-107: Data types for the poll area

7.5.2.3 Status Messages

Status messages are the static assignment of flags (bits) in the control-ler to plain text messages in the operating device. For status message addressing, use the data types BYTE, USINT, WORD, UINT, DWORD, UDINT, or ARRAY[1..N]. The following applies when using ARRAY: The type size multiplied by N provides the size of the message system in bytes.

7.5.2.4 Date and Time

The variables for synchronizing the time and date must use the data types USINT or ARRAY [1..N] OF BYTE.

Variable	Length
Date with a 2-digit year	3 Bytes
Date with a 4-digit year	4 Bytes
Time	3 Bytes
Weekday	1 Byte

Fig. 7-108: Byte lengths for the date and time

7.5.2.5 Variant Buffer

The variable for the variant buffer must use the data type BYTE or USINT.

7.5.2.6 Tables

The variable for representation of tables must use the data type ARRAY [1..N]. The ARRAY [1..N] has to be of one of the following base data types:

- BOOL,
- BYTE,
- WORD,
- DWORD,
- SINT,
- INT,
- DINT,
- USINT,
- UINT,
- UDINT,
- REAL or
- STRING.

7.5.3 Physical Connection

Plug-in connectors on the operating device for connection to the controller.

7.5.3.1 Pin Assignment

Pin	**Designation**	**Function**
6	TD	Transmitted Data
15	CTS	Clear to send
17	RTS	Request to send
18	RD	Received data
25	SGND	Signal Ground

Fig. 7-109: Pin assignment RS232

Pin	**Designation**	**Function**
8	T(A)	Transmitted Data (-)
9	T(B)	Transmitted Data (+)
11	SGND	Signal Ground
22	R(A)	Received Data (-)
23	R(B)	Received Data (+)

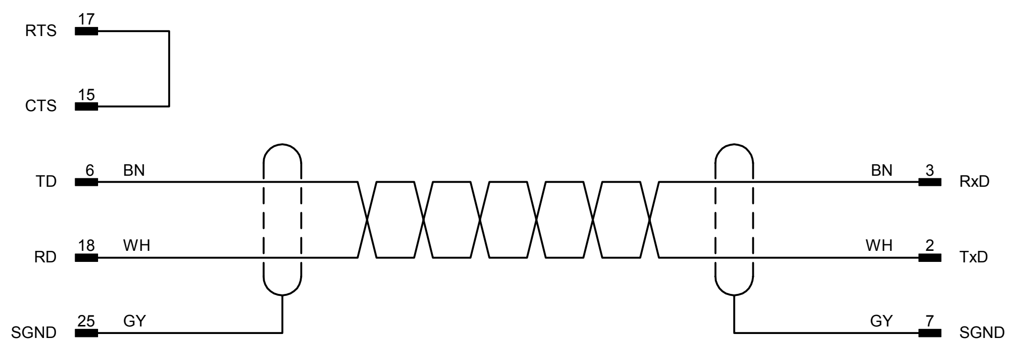
Fig. 7-110: Pin assignment RS485

7.5.3.2 Cable RS232 - Rexroth PPC-R

The following cabling diagram applies to operating devices with an universal interface **only**.

Operating device

Rexroth
PPC-R



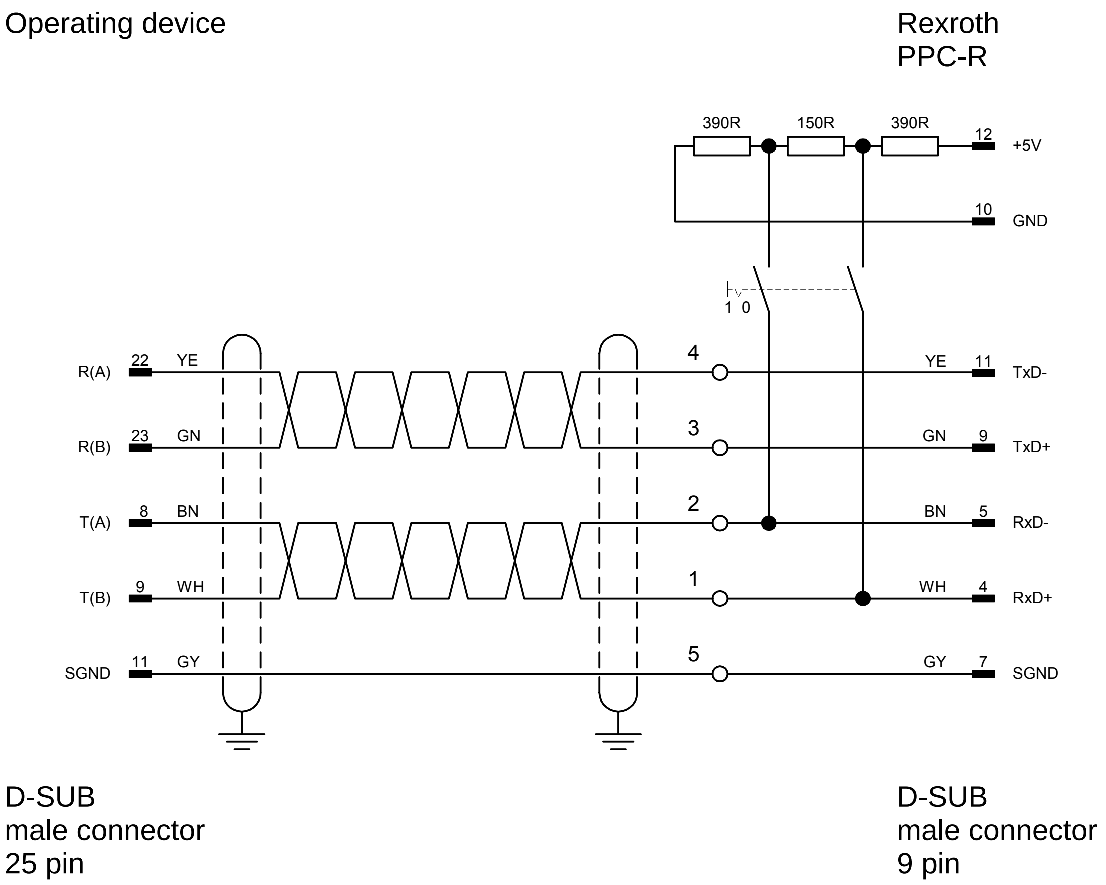
D-SUB
male connector
25 pin

D-SUB
male connector
15 pin

Both ends of the shield are connected to the metallic housing.

7.5.3.3 Cable RS485 - Rexroth PPC-R

The following cabling diagram applies to operating devices with an universal interface **only**.



Both ends of the shield are connected to the metallic housing.

7.5.4 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error

Code	XXXXX
Subcode	XXXXX
Retries	XXXXX

Code	Subcode	Description	Possible Cause
50	03	Framing error on serial interface	
	05	CRC error on serial interface	
	06	Parity error on serial interface	
	10	Polling area error	No polling area defined
60	10	Wrong telegram length	
	20	Wrong telegram Ident Number	
	30	Wrong block number	
	40	Wrong checksum	
	50	Negative acknowledgment	
	60	Waiting time exceeded: No response	Cable interruption, connection cut-off, wrong baud rate
70		Error from the controller	
	50	No service	Wrong service code
	51	No variable list	Variable list in controller is missing
80	20	Variable types not the same	Recompile the project using the current symbol file and reload it into the operating device.
	30	Invalid symbol	
	40	Waiting time exceeded	No valid symbol list in controller.Specify a higher value for Delay until Connection Set-Up

Fig. 7-111: Error messages, IndraLogic

7.5.5 Applications

7.5.5.1 IndraLogic from version 1.0

The programming software takes the global variables from the symbol file project_name.SYM and inserts them into the variable list.

The symbolic names cannot be longer than 80 characters.

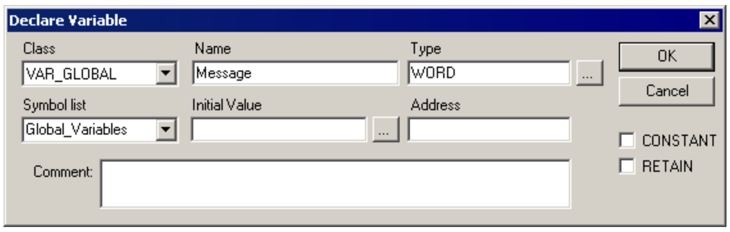
The entries in the variable list cannot be modified.

Declaring Global Variables

To declare global variables in IndraLogic:

1. Select **Auto Declare** from the **Edit** menu.

The **Declare Variable** dialog opens.

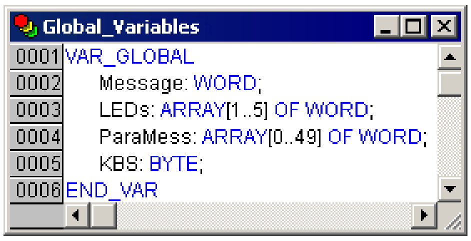


Class: VAR_GLOBAL
Name: Message
Type: WORD
Symbol list: Global_Variables
Initial Value:
Address:
Comment:

Fig. 7-112: Example of a variable declaration for global variables

2. Select the VAR_GOBAL class from the **Class** field.
3. Enter a name (Message) and a type (WORD).
4. Repeat step 3 for all additional global variables.
5. Click **OK** to confirm your input.

The **Global_Variables** window opens.



```
VAR_GLOBAL
    Message: WORD;
    LEDs: ARRAY[1..5] OF WORD;
    ParaMess: ARRAY[0..49] OF WORD;
    KBS: BYTE;
END_VAR
```

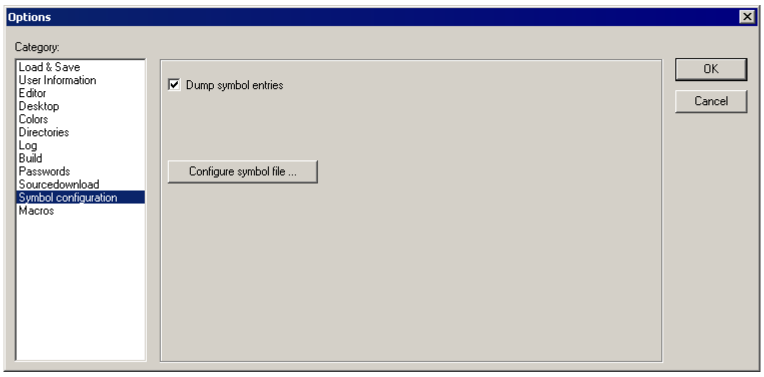
Fig. 7-113: Window Global variables

Activate Output into Symbol File

Specify the following settings in IndraLogic to write the global variables into a symbolic file.

1. Select **Options** from the **Project** menu.
2. Select **Symbol configuration**.

The **Options** dialog will look as follows:


Options

Category:

Load & Save
User Information
Editor
Desktop
Colors
Directories
Log
Build
Passwords
Sourcedownload
Symbol configuration
Macros

☑ Dump symbol entries

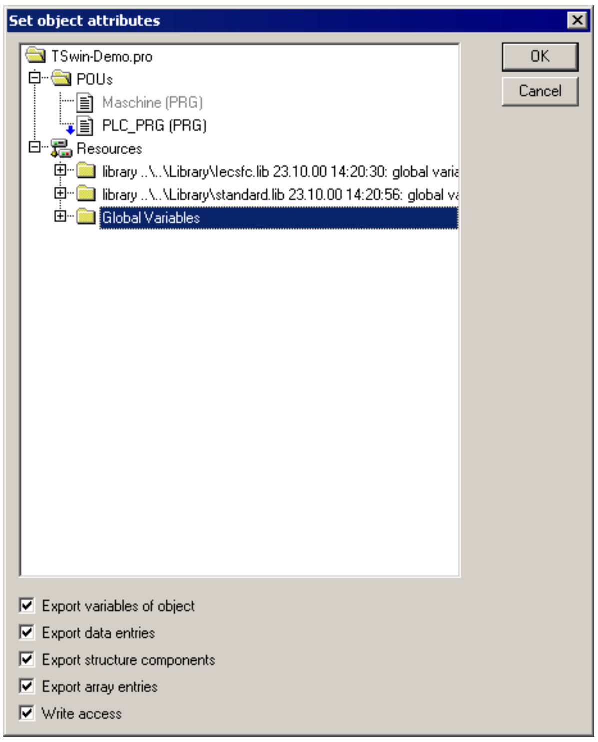
Configure symbol file ...

OK
Cancel

Fig. 7-114: Dialog Options - symbol configuration

3. Select the **Dump symbol entries** check box.
4. Click the **Configure symbol file** button.

The **Set object attributes** window opens.


Set object attributes

TSwin-Demo.pro
POUs
Maschine (PRG)
PLC_PRG (PRG)
Resources
library ..\..\Library\Iecsfc.lib 23.10.00 14:20:30: global varia
library ..\..\Library\standard.lib 23.10.00 14:20:56: global va
Global Variables

OK
Cancel

☑ Export variables of object
☑ Export data entries
☑ Export structure components
☑ Export array entries
☑ Write access

Fig. 7-115: Dialog Set object attributes

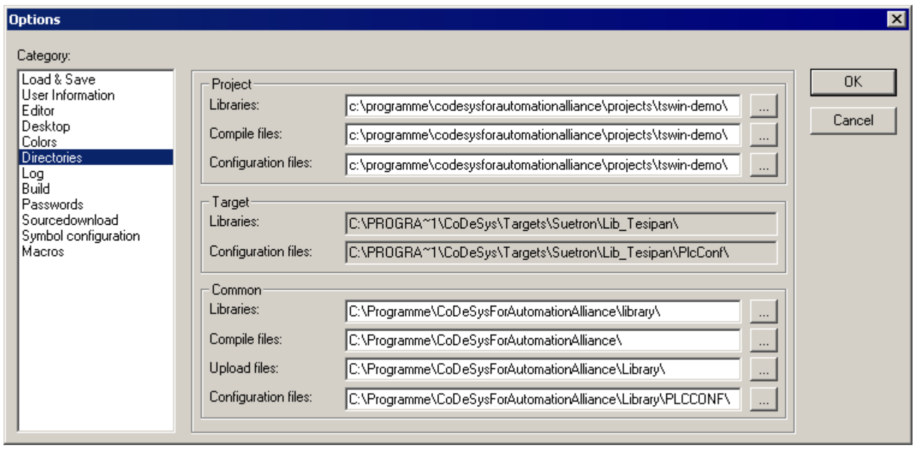
5. Select the **Global variables** entry.
6. Click **OK** to confirm your selection.

You are returned to the **Options** dialog.

Now you need to specify the position where the symbol file is to be stored.

1. Select **Directories** from the **Options** dialog.

The **Options** dialog will look as follows:



Options

Category:
- Load & Save
- User Information
- Editor
- Desktop
- Colors
- Directories
- Log
- Build
- Passwords
- Sourcedownload
- Symbol configuration
- Macros

Project
Libraries:	c:\programme\codesysforautomationalliance\projects\tswin-demo\
Compile files:	c:\programme\codesysforautomationalliance\projects\tswin-demo\
Configuration files:	c:\programme\codesysforautomationalliance\projects\tswin-demo\

Target
Libraries:	C:\PROGRA~1\CoDeSys\Targets\Suetron\Lib_Tesipan\
Configuration files:	C:\PROGRA~1\CoDeSys\Targets\Suetron\Lib_Tesipan\PlcConf\

Common
Libraries:	C:\Programme\CoDeSysForAutomationAlliance\library\
Compile files:	C:\Programme\CoDeSysForAutomationAlliance\
Upload files:	C:\Programme\CoDeSysForAutomationAlliance\Library\
Configuration files:	C:\Programme\CoDeSysForAutomationAlliance\Library\PLCCONF\

Fig. 7-116: Dialog Options - directories

2. From the **Project** area, select a directory for the **compile files**.
3. Click **OK** to confirm your selection.

You are returned to the **Options** dialog.

The symbol file will not be created until a compilation process takes place and is stored in the same directory as the project!

Target System Settings

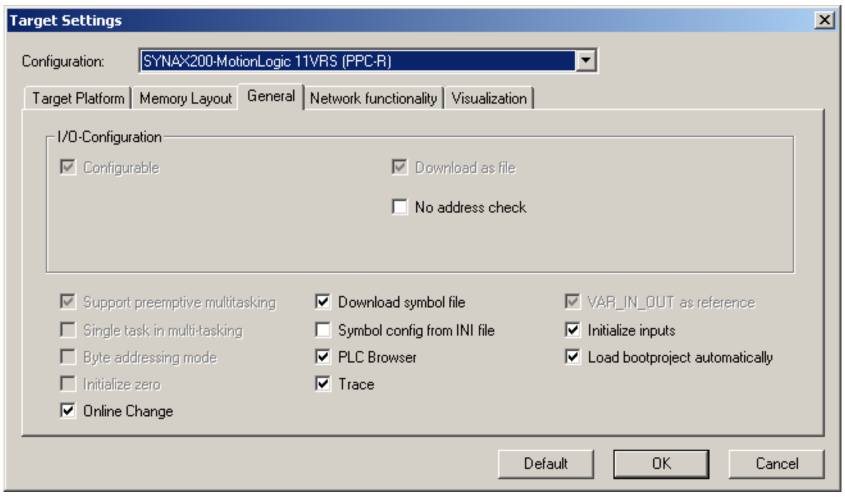
Select the following settings for the target system to ensure the symbol file is sent to the target system:

1. Open the **Resources** tab.
2. Double-click **Target settings**.

The **Target settings** dialog opens.

3. Open the **General** tab.
4. Select the **Download Symbol File** check box.

The **Target settings** dialog might look like the example below:


Target Settings

Configuration: SYNAX200-MotionLogic 11VRS (PPC-R)

Target Platform | Memory Layout | General | Network functionality | Visualization

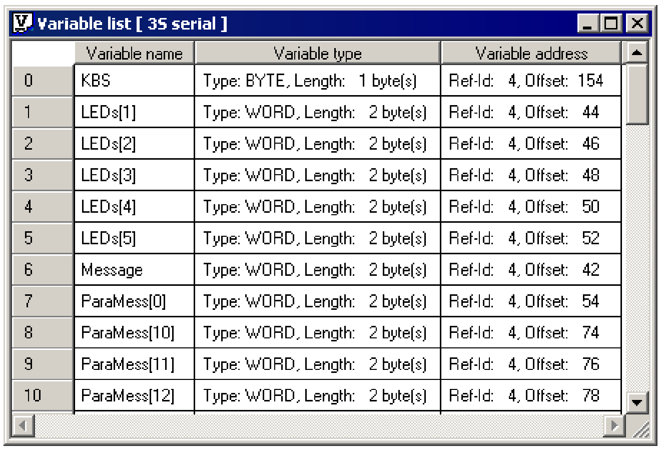
I/O-Configuration
- [✓] Configurable
- [✓] Download as file
- [ ] No address check

- [✓] Support preemptive multitasking
- [ ] Single task in multi-tasking
- [ ] Byte addressing mode
- [ ] Initialize zero
- [✓] Online Change
- [✓] Download symbol file
- [ ] Symbol config from INI file
- [✓] PLC Browser
- [✓] Trace
- [✓] VAR_IN_OUT as reference
- [✓] Initialize inputs
- [✓] Load bootproject automatically

Fig. 7-117: Dialog Target settings

Variable List

The programming software automatically places the symbolic variable entries created in the example into the variable list if you specified the correct directory and name in the communications parameters.


Variable list [3S serial]

	Variable name	Variable type	Variable address
0	KBS	Type: BYTE, Length: 1 byte(s)	Ref-Id: 4, Offset: 154
1	LEDs[1]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 44
2	LEDs[2]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 46
3	LEDs[3]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 48
4	LEDs[4]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 50
5	LEDs[5]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 52
6	Message	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 42
7	ParaMess[0]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 54
8	ParaMess[10]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 74
9	ParaMess[11]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 76
10	ParaMess[12]	Type: WORD, Length: 2 byte(s)	Ref-Id: 4, Offset: 78

Fig. 7-118: Variable list

This makes the variables globally available in the programming software.

7.6 PROFIBUS-DP raw

Profibus DP provides a manufacturer- and controller-independent data transmission protocol. The Profibus DP is a speed-optimized Profibus variant that is specially tailored to communication between programmable controllers and decentralized peripheral devices.

Profibus DP is implemented in the operating device and meets the requirements of parts 1 and 3 of the German standard DIN 19245. It also corresponds to the European field bus standard EN 50170.

As the operating device fulfills standardization requirements, it can be successfully integrated as a slave into the Profibus DP.

All operating devices can be linked using an integrated Profibus DP additional module. You can also link several operating devices to one master controller.

The entire PROFIBUS DP protocol is handled by the protocol chip SPC3. Transfer speeds of up to 12 MBaud are possible.

The operating device is used in the bus as a decentralized module that occupies up to 256 inputs and outputs. The size can be programmed from between 8- and 32-byte IN data, and between 8- and 32-byte OUT data. Data transfer is carried out via the peripheral area.

The input/output image is exchanged cyclically between the master and operating device via the bus. In this context, the operating device uses the cyclical input/output image for data exchange between the master and the slave. The data content to be interpreted is defined for both partners in a data profile.

All services required for running the operating device originate in the operating device. The operating device has client functions.

The controller reacts to the requests of the operating device. It has server functions.

The master module must interpret the incoming data according to the profile and also respond according to the profile. This is carried out using a function block in the controller that is able to interpret the requests in the IN data, and write a response to the OUT data.

7.6.1 Specification for PROFIBUS-DP

The specification of the operating device in PROFIBUS DP is defined using the device data base (GSD) file RX01081A.GSD.

7.6.1.1 Diagnosis

The operating device implements the station-related diagnosis.

5-byte user diagnosis data is transferred

- 1st byte = error number (1 = communication error in the operating device)
- 2nd and 3rd byte communication error code
- 4th and 5th byte communication error subcode

The communication-error code and subcode are the values that are also displayed on the operating device.

7.6.2 Data Profile

To allow direct data access to the different data areas in a controller, a data profile must be agreed between the master and the slave.

The first four bytes of the telegram length set are used as follows:

– Telegram sequential number and length
– Definition of the access
– Definition of the data area

7.6.2.1 Request telegram

Byte	Content	
1	Number of user data, sequential number of telegram	
2	Access	
3	Offset (Depends on Byte Order Setting)	Low (High)
4	High (Low)	
5	User Data 1st Byte	
6	User Data 2nd Byte	
n	User Data nth Byte	

Fig. 7-119: Request telegram

Byte 1: Number of User Data

Bit	Content
0	Number of User Data in Bytes. Specifies the number of bytes for the user data transfer.
1	
2	
3	
4	
5	Sequential Number of the Telegram. Identification number for each communication process. (0 = initialization cycle, 1 - 7 = normal sequential number)
6	
7	

Fig. 7-120: Number of User Data

Byte 2: Access

Bit	Content
0	Byte Number For Word Access 0 = First Byte, 1 = Second Byte
1	Reserved
2	
3	
4	
5	Access: 00 = Bit 01 = Byte 02 = Word 03 = Double word
6	
7	Data Direction: 0 = Read 1 = Write

Fig. 7-121: Access

Byte 3 and 4: Offset

These bytes contain the address for accessing a data area.

Byte 5 ff: User Data

The user data are located from byte 5 onwards to the end of the telegram.

7.6.2.2 Response Telegram

Byte	Content	
1	Number of User Data	
2	Access	
3	Return Code	Error
4		0x00
5	User Data 1st Byte	
6	User Data 2nd Byte	
n	User Data nth Byte	

Fig. 7-122: Response telegram

7.6.2.3 User Data

The user data are located from byte 5 onwards to end of the telegram.

7.6.2.4 Reading and Writing Bytes

Depending on the telegram length and access, up to 28 bytes of user data can be transferred during reading and writing operations.

When bytes are being read and written, the user data appears in the telegram as of byte five.

7.6.2.5 Reading Bits

When the system reads bits, it reads a byte, word or double word, based on the address width of the data area to be read.

The operating device masks out the requested bits, and displays the data in line with the display settings.

7.6.2.6 Writing Bits

Only an individual bit is set or deleted.

The controller receives a bit mask and link information from the operating device via the request telegram. The bit is set or deleted in the target address using the bit mask and the link information.

The byte order of the bit mask for word addresses is oriented to the protocol parameters specified for the byte order.

Byte	Content
5	Bit Mask
6	Logical Operation 0 = AND 1 = OR

Fig. 7-123: Writing to a byte address

Byte	Content
5	Bit Mask LOW
6	Bit Mask HIGH
7	Logical Operation 0 = AND 1 = OR

Fig. 7-124: Writing to a word address

7.6.3 Programming

7.6.3.1 Protocol Parameters

With the protocol parameters, you can adapt the communication of the controller used.

Maximum Waiting Time For Response

This parameter specifies how long the operating device waits for a response from the controller.

Configurable Values	Default Value
1 ms to 65535 ms	1000 ms

Fig. 7-125: Maximum waiting time for response

Delay until Connection Set-Up

This parameter specifies the waiting time after which the operating device starts the communication.

Configurable Values	Default Value
1000 ms to 65535 ms	5000 ms

Fig. 7-126: Delay until connection set-up

Station Number

Specifies the station number of the operating device within the PROFIBUS-DP structure. The station numbers 0 to 2 are reserved.

Configurable Values	Default Value
3 to 124	3

Fig. 7-127: Station number

Telegram Length

The telegram length is set to the PROFIBUS configuration. Specigy the same value in the PROFIBUS programming software.

Configurable Values	Default Value
8 Byte to 32 Byte	20 Byte

Fig. 7-128: Telegram length

Floating Point Number Format

This parameter specifies whether floating point numbers are exchanged in the Siemens-specific format or IEEE format.

Configurable Values	Default Value
OFF	X
ON	

Fig. 7-129: Floating point numbers in the Siemens format

Timer and Counter Format

This parameter specifies wether timer and counter are exchanged in the Siemens-specific format or IEEE format.

Configurable Values	Default Value
OFF	X
ON	

Fig. 7-130: Timer and counter in the Siemens format

Byte Order

Specify the byte order for word and double-word addresses. (Siemens = High-Low, Bosch = Low-High)

Configurable Values	Default Value
Low-High	X
High-Low	

Fig. 7-131: Byte order

Address Width

Specify the address width you want the operating device to use when accessing controller addresses.

Configurable Values	Default Value
1 = Byte Address	
2 = Word Address	X
4 = Double-Word Address	

Fig. 7-132: Adress width

7.6.3.2 Polling Area

Limits applying to the poll area:

- The variable must be word-oriented.
- The area must be contiguous.
- The controller must be able to access this area in bit-mode.
- The operating device must be able to access this area in word-mode.

7.6.3.3 Input Syntax

The following image illustrates the structure of the input syntax for variables in the programming software.

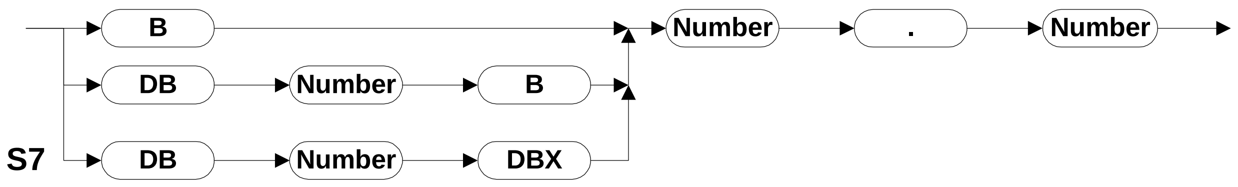


Fig. 7-133: Syntax diagram for bit access, PROFIBUS-DP.

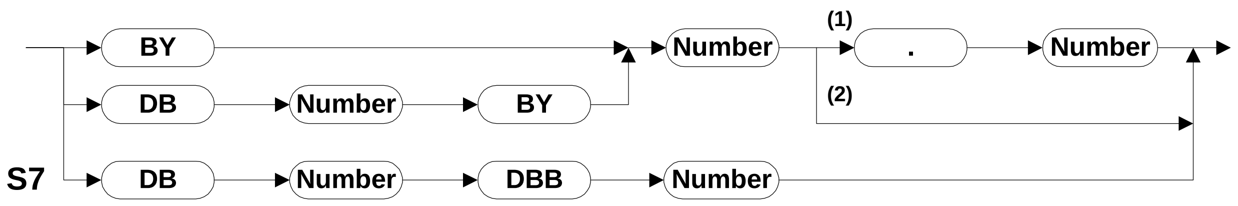


Fig. 7-134: Syntax diagram for byte access, PROFIBUS-DP.

1 The number in front of the point is a word or double-word address. The number after the point specifies the byte number within the word/double word.

2 The number in front of the point is a byte address.

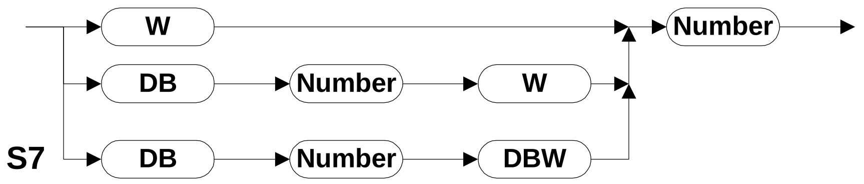


Fig. 7-135: Syntax diagram for word access, PROFIBUS-DP.

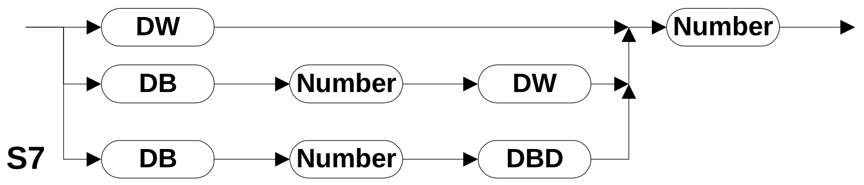


Fig. 7-136: Syntax diagram for double-word access, PROFIBUS-DP

7.6.4 Physical Connection

7.6.4.1 Pin Assignment

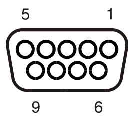


Fig. 7-137: 9 pin D-SUB female connector strip

Connector in the small operator terminal: 9 pin D-SUB female connector strip.

Pin	Designation	Function
1	nc	Not Connected
2	nc	Not Connected
3	RxD/TxD-P	Received Data / Transmitted Data Plus
4	CNTR-P	Repeater Control Signal Plus
5	DGND	Data Transmission Potential
6	VP	Supply Voltage of Terminators Plus
7	nc	Not Connected
8	RxD/TxD-N	Received Data / Transmitted Data Minus
9	CNTR-N	Repeater Control Signal Minus

Fig. 7-138: Pin assignment PROFIBUS DP

7.6.4.2 Cable for PROFIBUS-DP

In the wiring depicted below, the potential difference between the data reference potentials DGND of all connections are NOT to exceed +/- 7 V.

Ensure that no compensating current flow through the bus cable shield. Install a separate equipotential bonding conductor.

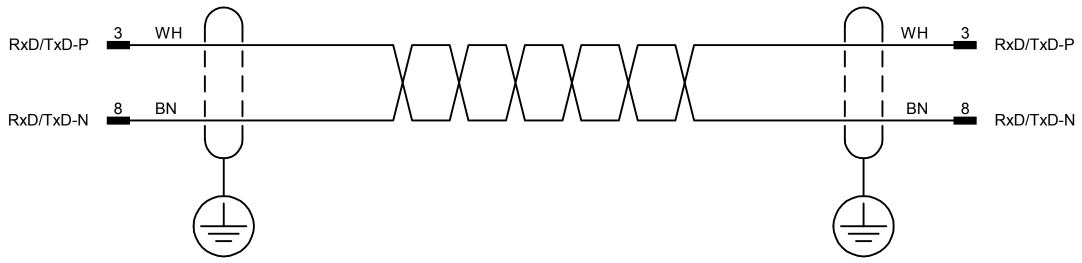


Fig. 7-139: Connecting cable PROFIBUS-DP

PROFIBUS-DP raw

There are two cable specifications for PROFIBUS-FMS and PROFIBUS-DP:

Parameter	Cable type A	Cable type B
Wave Impedance	135 to 165 Ohm (for f = 3 to 20 MHz)	100 to 135 Ohm (for f > 100 MHz
Cable capacity	< 30 pF/m	< 60 pF/m
Wire cross-section	> 0.34 mm^2	> 0.22 mm^2
Loop Impedance	< 110 Ohm/km	---
Signal attentuation	max. 9 dB	max. 9 dB
Cable type	twisted-pair 1 x 2 / 2 x 2 / 1 x 4 wires	twisted-pair1x 2 / 2 x 2 / 1 x 4 wires
Shielding	Copper braided shielding or braided shielding + foil shielding	Copper braided shielding or braided shielding + foil shielding

Fig. 7-140: Cable specification for PROFIBUS

7.6.4.3 Transfer Speed and Line Length

With the PROFIBUS, data can be transferred using different transfer speeds. However, the higher the transfer speed, the shorter the maximum permitted line length. The values listed in the following table apply to the cable type A which is more closely specified in DIN E 19245 part 3.

Baud Rate (Bit/s)	Line Length (m)
187 500	1000
500 000	400
1 500 000	200
3 000 000	100
6 000 000	100
12 000 000	100

Fig. 7-141: Transfer speed versus line length for PROFIBUS

7.6.5 Error Messages

Error messages are displayed on the operating device along with a code and subcode. Error messages are composed as follows:

Communication Error	
Code	XXXXX
Subcode	XXXXX
Retries	XXXXX

Code	Subcode	Error Type	Possible Cause
1			
	1	Slave is currently not ready	
	2	Packets out of sequence	
	3	Protocol framing error	
	4	Timeout	
	5	CRC error	
	6	Parity error	
	7	Send process aborted	
	8	Receive process aborted	
	9	Buffer too small for cyclic data	
	10	No cyclic data defined	
	12	Cyclic data already defined	
	15	The selected protocol is not supported	
	16	Receive buffer overrun	
	40	Undefined system variable	
50		Error initializing the SPC3	
	1	Buffer too large	
	2	No initialization of SPC3	
	4	No memory for telegram buffer	
60		No configuration from master	
61		Wrong input length	
62		Wrong output length	
63		Error in configuration data, reparameterization required	

Fig. 7-142: Fehlermeldungen PROFIBUS-DP

PROFIBUS-DP raw

Code	Subcode	Error Type	Possible Cause
64		Protocol chip requires configuration update, re-parameterization required.	
65		No communication via protocol chip, reparame-terization required.	
66		Protocol chip reset, reparameterization required.	
67		Watchdog time error, reparameterization re-quired.	
70		Operating device is not polled	
	0	Distinguishing feature for manufacturer	
	1	Distinguishing feature for manufacturer	
71	xxx	No response to order. xxx = variable number	
100		Base no. for error from PLC function block. PLC error is added to 100. The subcode indicates the offset value for the access, during which the er-ror occurred.	
	i.e. 102	Access to DB via FB111 / FB112 DB does not exist	

Fig. 7-142: Fehlermeldungen PROFIBUS-DP

7.6.6 Applications

The controller program, which is usually a function block, must handle the requests of the operating device in line with the data profile.

The details depend on the controller. The following sections explain the controller-specific applications that have been developed to date.

The device data base (GSD) file RX02081A.GSD can be used to set the parameters of the operating devices in the PLC software. This file is available in a subdirectory of the programming software and in our Internet download area.

7.6.6.1 Rexroth Controllers

The PLC program communicates with the PROFIBUS DP via the input/output peripheral area.

Each participant, including each operating device in the PROFIBUS DP, is assigned an IN and OUT data channel.

The channel is assigned using the parameterization of the Bosch controller's PROFIBUS DP master module. The module MP-DP12 is used in Bosch controllers.

Configuration in IndraLogic

Library IL_VCP_DP.lib

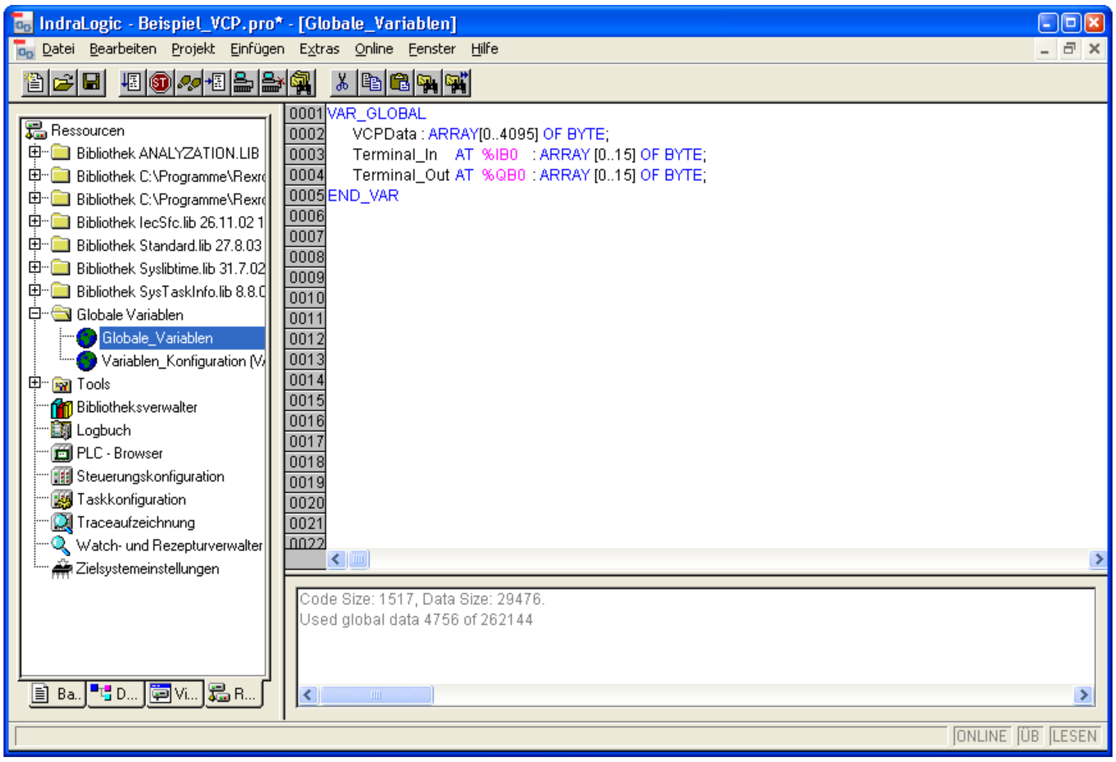
For a Profibus communication with the small operator terminal, the library IL_VCP_DP must be integrated into the IndraLogic programming interface. This makes the following function blocks available:

- VCP_PBS16_A4096 (for 16 byte data length)
- VCP_PBS32_A4096 (for 32 byte data length)
- VCP_PB32_A65536 (for 32 byte data length)

After integrating a communication block, the user must globally declare the following variables in the PLC program for each instance:

```
<Var.-name> AT %IB<Address of VCP in contr. conf.> ARRAY[0..<(Data length
VCP)-1>]OF BYTE <Var.-name> AT %QB<Address of VCP in contr. conf.> AR-
RAY[0..<(Data length VCP)-1>]OF BYTE<Var.-name> ARRAY[0..4096] OF BYTE or
<Var.-name> ARRAY[0..65536] OF BYTE when using the module
VCP_PB32_A65536.
```

PROFIBUS-DP raw


IndraLogic - Beispiel_VCP.pro* - [Globale_Variablen]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Ressourcen
Bibliothek ANALYZATION.LIB
Bibliothek C:\Programme\Rexr
Bibliothek C:\Programme\Rexr
Bibliothek IecSfc.lib 26.11.02 1
Bibliothek Standard.lib 27.8.03
Bibliothek Syslibtime.lib 31.7.02
Bibliothek SysTaskInfo.lib 8.8.0
Globale Variablen
Globale_Variablen
Variablen_Konfiguration (V
Tools
Bibliotheksverwalter
Logbuch
PLC - Browser
Steuerungskonfiguration
Taskkonfiguration
Traceaufzeichnung
Watch- und Rezepturverwalter
Zielsystemeinstellungen
Ba.. D... Vi... R...
ONLINE ÜB LESEN

```
VAR_GLOBAL
    VCPData : ARRAY[0..4095] OF BYTE;
    Terminal_In   AT  %IB0   : ARRAY [0..15] OF BYTE;
    Terminal_Out AT  %QB0  : ARRAY [0..15] OF BYTE;
END_VAR
```

```
Code Size: 1517, Data Size: 29476.
Used global data 4756 of 262144
```

Fig. 7-143: Variable declaration

Explanation of function blocks

Function blocks activate the Profibus-DP protocol for VCPxx small operator terminals. In addition, the I/O image for the physical addresses between PLC and operating terminal are handled.The data width used for transferring data is 16 or 32 bytes, depending on the function block being used. The address range provided through an ARRAY has a size of 4096 or 65536 bytes (total for inputs and outputs).

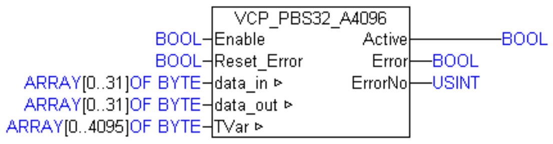

VCP_PBS32_A4096
BOOL Enable
BOOL Reset_Error
ARRAY[0..31]OF BYTE data_in
ARRAY[0..31]OF BYTE data_out
ARRAY[0..4095]OF BYTE TVar
Active BOOL
Error BOOL
ErrorNo USINT

Fig. 7-144: Example for function block VCP_PBS32_A4096

	Name	Type	Comment
VAR_IN	Enable	BOOL	TRUE: FB is processed FALSE: FB is not processed
	Reset_Error	BOOL	TRUE: 'Error' reset (to FALSE) and 'ErrorNo' is reset to 0

Fig. 7-145: Interface of function blocks

PROFIBUS-DP raw

	Name	Type	Comment
VAR_IN_OUT	Data_in	ARRAY OF BYTE [0..15] or ARRAY OF BYTE [0..31]	Data for connection of physical inputs for small operator terminal
	Data_out	ARRAY OF BYTE [0..15] or ARRAY OF BYTE [0..31]	Data for connection of physical outputs for small operator terminal
	TVar	ARRAY OF BYTE [0..4096] or ARRAY OF BYTE [0..65535]	Array used to read from and write to the operating terminal.
VAR_OUT	Active	BOOL	TRUE while 'Enable' is also TRUE
	Error	BOOL	TRUE while an error is pending. Can be reset using 'Reset-Error'.
	ErrorNo	USINT	Error type: 4: Calculation Error

Fig. 7-145: Interface of function blocks

VI-Composer

When configuration is carried out using the Rexroth VI-Composer, the addresses of the variable list refer to the corresponding byte in the 'TVar' array used in the PLC program to exchange data.

Error handling

As soon as an error occurs, communication is interrupted and COMMUNICATION ERROR, ERROR CODE 110 is displayed on the screen of the small operator terminal.

The error type (ErrorNo) indicates that this is an address computation error (CalculationError).

Configuration in WINSPS

Evaluation of the control bytes in the PLC Program:

The PLC program must cyclically poll the peripheral area that is assigned to the operating device. Using the sequential number, it must check whether a new request has been received from the operating device. In addition, bytes 1 and 2 must be copied, unchanged, from the request telegram to the response telegram, and 0x00 must be written to byte 3.

You require the following modules for this task. They are contained in a subfolder of the programming software’s installation folder.

Controller	Function Block
CL200	..\FBs\Profibus\BOSCH\WINSPS\CL200\BT_PB2.pxl
CL300	..\FBs\Profibus\BOSCH\WIN-SPS\CL345\BT_PB345.pxl
CL400	..\FBs\Profibus\BOSCH\WIN-SPS\CL345\BT_PB345.pxl
CL500	..\FBs\Profibus\BOSCH\WIN-SPS\CL345\BT_PB345.pxl
SoftSPS	..\FBs\Profibus\BOSCH\WINSPS\PLC\BT_PBPLC.pxl

Fig. 7-146: Function blocks for the programming software WINSPS

Error Handling in the PLC Program:

Errors can be entered in the return code, byte 4 of the response telegram. If no error occurs, byte 4 must be deleted. Possible errors are:

- DB does not exist.

Function Blocks Supplied:

You must configure the operating device as the slave using 'n Byte kons. Daten E/A' (n byte cons. data I/O) in the DP master module.

If you are using interrupts, you must save the scratch flags and the four registers used in the interrupt OB.

Inserting the Library Files in WINSPS

1. Start WinSPS.
2. Copy the corresponding pxl file to the ZSO directory of the PLC project.

You can only copy the file to the appropriate project directory as modules of the incorrect controller type are not recognized!

3. Assign the library in the toolbar.

Example:

```
FC10, R BT_PB345
```

4. Open the editor in the PLC software.

Example:

OB1

Select the PROFIBUS block from the 'Edit/Parameter list' menu.

Parameterizing the call-up function:

Example:

For two operating devices:

```
...
;DEF für Gerät 1
DEF                 10,-EZ_Basisn
DEF                 10,-AZ_Basisn
DEF                 DB50,-DBNR
DEF                 0,-WDNR
DEF                 20,-TLNG
;DEF für Gerät 2
DEF                 10,-EZ_Basisn2
DEF                 10,-AZ_Basisn2
DEF                 DB50,-DBNR2
DEF                 0,-WDNR2
DEF                 20,-TLNG2
;Aufruf Gerät 1
BA                  -BT_345,5FC10
P0               W  -EZ_Basisn
P1               W  -AZ_Basisn
P2                  -DBNR
P3               W  -WDNR
P4               W  -TLNG
;Aufruf Gerät 1
BA                  -BT_345,5FC10
P0               W  -EZ_Basisn2
P1               W  -AZ_Basisn2
P2                  -DBNR2
P3               W  -WDNR2
P4               W  -TLNG2
...
```

Function Block BT_PB345:

The function block BT_DP345 is used to decode the transfer protocol of the operating devices. It ensures consistent data transfer.

When you program the controller, note that a total of 64 bytes as of the address DB[P2] W[P3] are reserved for processing the protocol. Other program components cannot use this area!

The function block BT_PB345 uses the following parameters:

Parameter	Function
P0	EZ Base Address
P1	AZ Base Address
P2	Data block for storing the EZ/AZ data
P3	Base Address in Data Block [P2]
P4	Telegram Length corresponds to the number of EZ/AZ data of the slave configuration (8, 12, 16, 20, 28, or 32 bytes)

Fig. 7-147: Parameters for function block BT_PB345

Configuration in PROFI

Controller	Function Block
CL200	..\FBs\Profibus\BOSCH\PROFI\CL200\BT_MAIN.PBO ..\FBs\Profibus\BOSCH\PROFI\CL200\BT_READ.PBO ..\FBs\Profi-bus\BOSCH\PROFI\CL200\BT_WRITE.PBO ..\FBs\Profibus\BOSCH\PROFI\CL200\OB1.PBO
CL300	..\FBs\Profibus\BOSCH\PROFI\CL350400/ BT_MAIN.PCO ..\FBs\Profi-bus\BOSCH\PROFI\CL350400\BT_READ.PCO ..\FBs\Profi-bus\BOSCH\PROFI\CL350400\BT_WRITE.PCO ..\FBs\Profibus\BOSCH\PROFI\CL350400\OB1.PCO
CL400	..\FBs\Profibus\BOSCH\PROFI\CL350400/ BT_MAIN.PCO ..\FBs\Profi-bus\BOSCH\PROFI\CL350400\BT_READ.PCO ..\FBs\Profi-bus\BOSCH\PROFI\CL350400\BT_WRITE.PCO ..\FBs\Profibus\BOSCH\PROFI\CL350400\OB1.PCO
CL500	..\FBs\Profibus\BOSCH\PROFI\CL500/BT_MAIN.P5O ..\FBs\Profi-bus\BOSCH\PROFI\CL350400\BT_READ.P5O ..\FBs\Profi-bus\BOSCH\PROFI\CL350400\BT_WRITE.P5O ..\FBs\Profibus\BOSCH\PROFI\CL350400\OB1.P5O

Fig. 7-148: Function blocks for the programming software PROFI

Function Block BT_MAIN

Structure of the block:

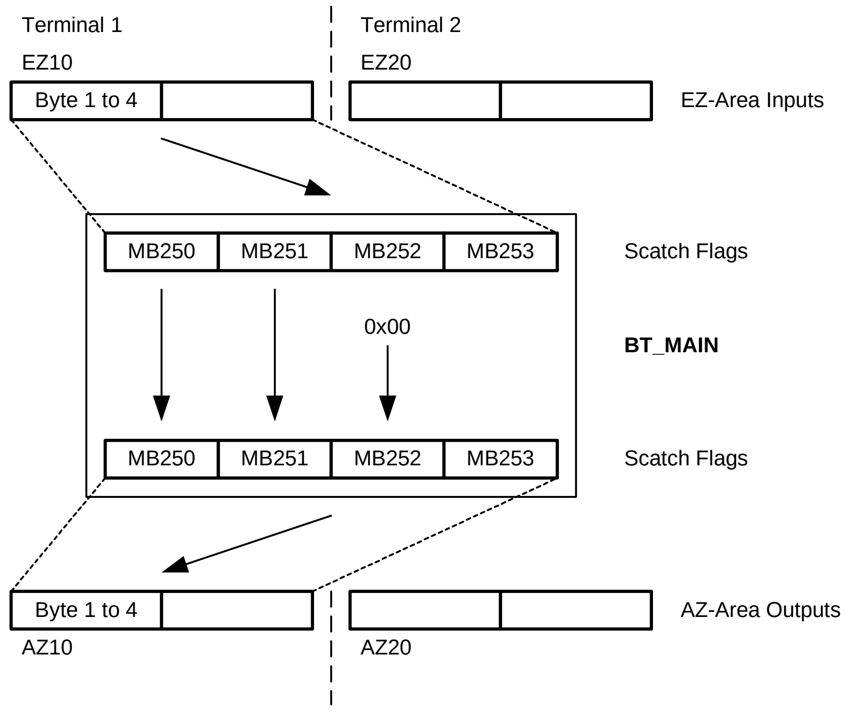


Fig. 7-149: Structure of the BT_MAIN

Function Block BT_MAIN Call-Up

Example:

Call-up in OB1

```
;OB1 Organisationsbaustein
;*****************************************
;Profibus-DP-Koomunikation mit Bediengerät
;Beispiel zur Einbindung im OB1
;*****************************************
;Befehle notwendig für Profibus
L  W         EZ2,A  ;Adresse muss mit Koppeladresse übereinstimmen
T  W         A,AZ2  ;nur für CL400
;Einmal pro Bediengerät aufrufen
BA -BT_MAIN,4       ;Aufruf für das erste Bediengerät
;                   +---+
P0 W           K10  ; < ! Adresse des Eingangsbereichs
P1 W           K10  ; < ! Adresse des Ausgangsbereichs
P2 W           DB0  ; < ! Nummer des Datenbausteins
P3 W            D0  ; < ! Datenwortnummer
;                   +---+
PE
```

Function Block BT_READ

The function block BT_READ interprets the subsequent bytes in the telegram as follows:

– Byte 2, bit 0 is interpreted as a byte code for a byte access to a word address.

Value	Meaning
0	Odd Address - Low Byte
1	Even Address - High Byte

Fig. 7-150: Byte code in byte 2

– Byte 3 contains the data block number.
– Byte 4 contains the data word number within the DB.

The program module doubles the data word number for the even-numbered byte number in the DB.

Function Block BT_WRITE

The function block BT_WRITE interprets the subsequent bytes in the telegram as follows:

– Byte 2, bit 0 is interpreted as a byte code for a byte access to a word address.

Value	Meaning
0	Odd Address - Low Byte
1	Even Address - High Byte

Fig. 7-151: Byte code in byte 2

– Byte 3 contains the data block number within the DB.
– Byte 4 contains the data word number within the DB (0 to 255).
– Byte 5 and
– Byte 6 contain the bit mask for the logical operation.
– Byte 7 contains the logical instruction (AND / OR).

Parameterization of the BM-DP12 Module

Set the parameters for the module using the Bosch DP software.

The device data base (GSD) file RX01081A.GSD which is supplied is directly read in by the DP software. This means that the data required to set the parameters of the operating devices are automatically available.

Select the operating device with the required data width.

The function block copies bytes 1 and 2, unchanged, from the request telegram to the response telegram, and writes 0x00 to byte 3.

The function block uses MW248 to MW254 as scratch flags.

For each operating device, the program block also requires any data word of a data block. The data word is transferred as a parameter during the call. The telegram sequential number is saved in this data word.

The function block cyclically checks the content of byte 1 – bits 5 to 7.

If the value 0 is contained here, the telegram number memory is reset.

If in byte 1, bits 5 to 7 are not equal to the content of the telegram number memory, a new request telegram has been received from the operating device, and this must be evaluated and a response sent.

The function block is called cyclically in OB1 with the corresponding parameters for each operating device.

Protocol Parameters for BM DP12

Set the following parameters for the protocol:

Parameter	Value
Maximum Waiting Time for Response [ms]	1000
Delay Until Connection Set-Up [ms]	5000
Station Number	3
Telegram Length	16
Floating Point Number in the Siemens Format	Inactive
Byte Order is High-Low	Inactive
Address Width	2

Fig. 7-152: Protocol parameters for the Bosch CL series

Set the parameters using the Bosch DP software. The supplied device data base file RX01081A.GSD is directly imported by the DP software. Therefore, the data required to set the parameters of the operating devices are available in the DP software. You can specify 8, 12, or 16 bytes for the telegram length.

Protocol Parameters for the Bosch CL Series

Set the following parameters for the protocol:

Parameter	Value
Maximum Waiting Time for Response [ms]	1000
Delay Until Connection Set-Up [ms]	5000
Station Number	3
Telegram Length	20
Floating Point Number in the Siemens Format	Inactive
Byte Order is High-Low	Inactive
Address Width	2

Fig. 7-153: Protocol parameters for the Bosch CL series

Defining Variables

Specify the variable addresses in either the hexadecimal notation or using the following syntax formats:

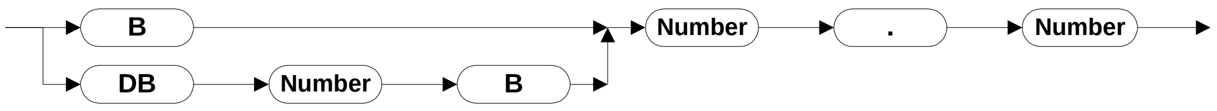


Fig. 7-154: Bit access for PROFIBUS using the Bosch CL series

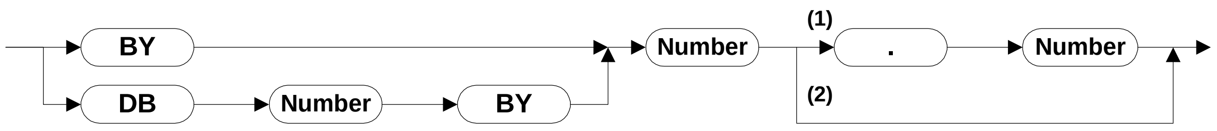


Fig. 7-155: Byte access for PROFIBUS using the Bosch CL series

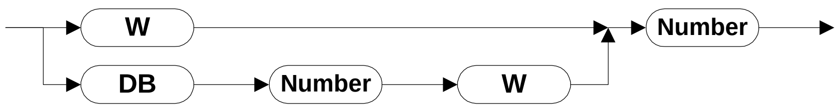


Fig. 7-156: Word access for PROFIBUS with the Bosch CL series

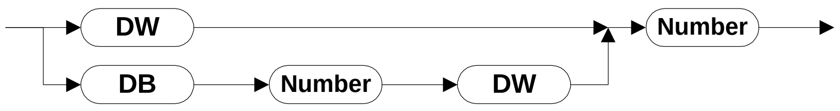


Fig. 7-157: Double-word access for PROFIBUS using the Bosch CL series

In the variable list of the programming software, you can also enter the addresses in hexadecimal notation:

Variable Name	Address (hex)	Low-Bit	High-Bit	PLC Access	PLC Address
Var1	DW H124B			Double Word	DB18 D150 to D153
Var2	W H124B			Word	DB18 D150 and D151
Var3	BY H124B			Byte	DB18 D151
Var4	BY H124B	1	1	Bit	DB18 D150 Bit 5
Var5	B H124B	13	13	Bit	DB18 D151 Bit 5

Fig. 7-158: Hexadecimal notation for addresses

PROFIBUS-DP raw

8 Shielding D-SUB Connectors

You must shield D-SUB connectors as follows:

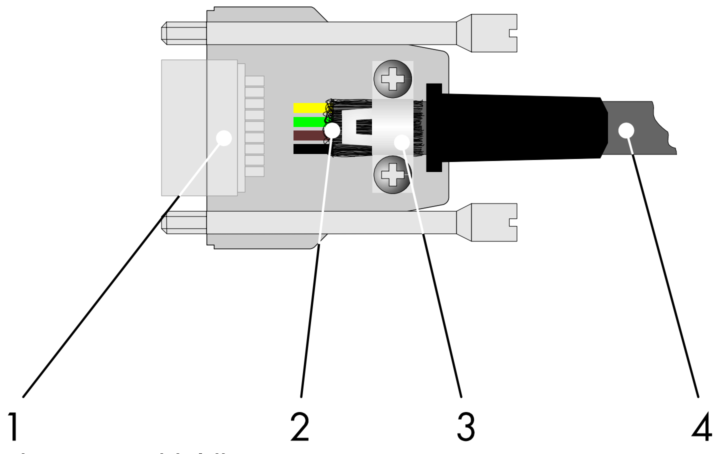


Fig. 8-1: Shielding D-SUB connectors

1 D-SUB connector
2 Shield
3 Cable clip
4 Cable

The shield must be folded back into a flat position over the cable sheath.

When fastening the cable with the cable clip, as much of the shielding as possible must be in contact with the housing and sufficient strain relieve must be ensured.

9 List of Figures

List of Figures

List of Figures

List of Figures

List of Figures

10 Index

11 Service & Support

11.1 Helpdesk

Unser Kundendienst-Helpdesk im Hauptwerk Lohr am Main steht Ihnen mit Rat und Tat zur Seite. Sie erreichen uns

Our service helpdesk at our headquarters in Lohr am Main, Germany can assist you in all kinds of inquiries. Contact us

- telefonisch - by phone: **+49 (0) 9352 40 50 60**
  über Service Call Entry Center Mo-Fr 07:00-18:00
  - via Service Call Entry Center Mo-Fr 7:00 am - 6:00 pm
- per Fax - by fax: **+49 (0) 9352 40 49 41**
- per e-Mail - by e-mail: **service.svc@boschrexroth.de**

11.2 Service-Hotline

Außerhalb der Helpdesk-Zeiten ist der Service direkt ansprechbar unter

After helpdesk hours, contact our service department directly at

+49 (0) 171 333 88 26

oder - or **+49 (0) 172 660 04 06**

11.3 Internet

Unter **www.boschrexroth.com** finden Sie ergänzende Hinweise zu Service, Reparatur und Training sowie die **aktuellen** Adressen *) unserer auf den folgenden Seiten aufgeführten Vertriebs- und Servicebüros.

☐	Verkaufsniederlassungen
▨	Niederlassungen mit Kundendienst

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit unserem für Sie nächstgelegenen Ansprechpartner auf.

*) Die Angaben in der vorliegenden Dokumentation können seit Drucklegung überholt sein.

At **www.boschrexroth.com** you may find additional notes about service, repairs and training in the Internet, as well as the **actual** addresses *) of our sales- and service facilities figuring on the following pages.

☐	sales agencies
▨	offices providing service

Please contact our sales / service office in your area first.

*) Data in the present documentation may have become obsolete since printing.

11.4 Vor der Kontaktaufnahme... - Before contacting us...

Wir können Ihnen schnell und effizient helfen wenn Sie folgende Informationen bereithalten:

1. detaillierte Beschreibung der Störung und der Umstände.
2. Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern.
3. Tel.-/Faxnummern und e-Mail-Adresse, unter denen Sie für Rückfragen zu erreichen sind.

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

1. Detailed description of the failure and circumstances.
2. Information on the type plate of the affected products, especially type codes and serial numbers.
3. Your phone/fax numbers and e-mail address, so we can contact you in case of questions.

11.5 Kundenbetreuungsstellen - Sales & Service Facilities

11.5.1 Deutschland - Germany

vom Ausland:	(0) nach Landeskennziffer weglassen!
from abroad:	don't dial (0) after country code!

Vertriebsgebiet Mitte Germany Centre Rexroth Indramat GmbH Bgm.-Dr.-Nebel-Str. 2 / Postf. 1357 97816 Lohr am Main / 97803 Lohr **Kompetenz-Zentrum Europa** Tel.: +49 (0)9352 40-0 Fax: +49 (0)9352 40-4885	**SERVICE AUTOMATION** **CALL ENTRY CENTER** **Helpdesk** **MO – FR** **von 07:00 - 18:00 Uhr** **from 7 am – 6 pm** **Tel. +49 (0) 9352 40 50 60** **Fax +49 (0) 9352 40 49 41** service.svc@boschrexroth.de	**SERVICE AUTOMATION** **HOTLINE 24 / 7 / 365** **außerhalb der Helpdesk-Zeit** **out of helpdesk hours** **Tel.: +49 (0)172 660 04 06** oder / or **Tel.: +49 (0)171 333 88 26**	**SERVICE AUTOMATION** **ERSATZTEILE / SPARES** verlängerte Ansprechzeit - extended office time - ♦ nur an Werktagen - only on working days - ♦ von 07:00 - 18:00 Uhr - from 7 am - 6 pm - **Tel. +49 (0) 9352 40 42 22**
Vertriebsgebiet Süd Germany South Bosch Rexroth AG Landshuter Allee 8-10 80637 München Tel.: +49 (0)89 127 14-0 Fax: +49 (0)89 127 14-490	Vertriebsgebiet West Germany West Bosch Rexroth AG Regionalzentrum West Borsigstrasse 15 40880 Ratingen Tel.: +49 (0)2102 409-0 Fax: +49 (0)2102 409-406 +49 (0)2102 409-430	Gebiet Südwest Germany South-West Bosch Rexroth AG Service-Regionalzentrum Süd-West Siemensstr. 1 70736 Fellbach Tel.: +49 (0)711 51046–0 Fax: +49 (0)711 51046–248	
Vertriebsgebiet Nord Germany North Bosch Rexroth AG Walsroder Str. 93 30853 Langenhagen Tel.: +49 (0) 511 72 66 57-0 Service: +49 (0) 511 72 66 57-256 Fax: +49 (0) 511 72 66 57-93 Service: +49 (0) 511 72 66 57-783	Vertriebsgebiet Mitte Germany Centre Bosch Rexroth AG Regionalzentrum Mitte Waldecker Straße 13 64546 Mörfelden-Walldorf Tel.: +49 (0) 61 05 702-3 Fax: +49 (0) 61 05 702-444	Vertriebsgebiet Ost Germany East Bosch Rexroth AG Beckerstraße 31 09120 Chemnitz Tel.: +49 (0)371 35 55-0 Fax: +49 (0)371 35 55-333	Vertriebsgebiet Ost Germany East Bosch Rexroth AG Regionalzentrum Ost Walter-Köhn-Str. 4d 04356 Leipzig Tel.: +49 (0)341 25 61-0 Fax: +49 (0)341 25 61-111

11.5.2 Europa (West) - Europe (West)

vom Ausland: (0) nach Landeskennziffer weglassen, from abroad: don't dial (0) after country code,

Italien: 0 nach Landeskennziffer mitwählen Italy: dial 0 after country code

Austria - Österreich Bosch Rexroth GmbH Electric Drives & Controls Stachegasse 13 1120 Wien Tel.: +43 (0) 1 985 25 40 Fax: +43 (0) 1 985 25 40-93	Austria – Österreich Bosch Rexroth GmbH Electric Drives & Controls Industriepark 18 4061 Pasching Tel.: +43 (0)7221 605-0 Fax: +43 (0)7221 605-21	Belgium - Belgien Bosch Rexroth NV/SA Henri Genesestraat 1 1070 Bruxelles Tel: +32 (0) 2 451 26 08 Fax: +32 (0) 2 451 27 90 info@boschrexroth.be service@boschrexroth.be	Denmark - Dänemark BEC A/S Zinkvej 6 8900 Randers Tel.: +45 87 11 90 60 Fax: +45 87 11 90 61
Great Britain – Großbritannien Bosch Rexroth Ltd. Electric Drives & Controls Broadway Lane, South Cerney Cirencester, Glos GL7 5UH Tel.: +44 (0)1285 863000 Fax: +44 (0)1285 863030 sales@boschrexroth.co.uk service@boschrexroth.co.uk	Finland - Finnland Bosch Rexroth Oy Electric Drives & Controls Ansatie 6 017 40 Vantaa Tel.: +358 (0)9 84 91-11 Fax: +358 (0)9 84 91-13 60	France - Frankreich Bosch Rexroth SAS Electric Drives & Controls Avenue de la Trentaine (BP. 74) 77503 Chelles Cedex Tel.: +33 (0)164 72-63 22 Fax: +33 (0)164 72-63 20 **Hotline: +33 (0)608 33 43 28**	France - Frankreich Bosch Rexroth SAS Electric Drives & Controls ZI de Thibaud, 20 bd. Thibaud (BP. 1751) 31084 Toulouse Tel.: +33 (0)5 61 43 61 87 Fax: +33 (0)5 61 43 94 12
France – Frankreich Bosch Rexroth SAS Electric Drives & Controls 91, Bd. Irène Joliot-Curie 69634 Vénissieux – Cedex Tel.: +33 (0)4 78 78 53 65 Fax: +33 (0)4 78 78 53 62	Italy - Italien Bosch Rexroth S.p.A. Via G. Di Vittorio, 1 20063 Cernusco S/N.MI **Hotline: +39 02 92 365 563** Tel.: +39 02 92 365 1 Service: +39 02 92 365 300 Fax: +39 02 92 365 500 Service: +39 02 92 365 516	Italy - Italien Bosch Rexroth S.p.A. Via Paolo Veronesi, 250 10148 Torino Tel.: +39 011 224 88 11 Fax: +39 011 224 88 30	Italy - Italien Bosch Rexroth S.p.A. Via Mascia, 1 80053 Castellamare di Stabia NA Tel.: +39 081 8 71 57 00 Fax: +39 081 8 71 68 85
Italy - Italien Bosch Rexroth S.p.A. Via del Progresso, 16 (Zona Ind.) 35020 Padova Tel.: +39 049 8 70 13 70 Fax: +39 049 8 70 13 77	Italy - Italien Bosch Rexroth S.p.A. Via Isonzo, 61 40033 Casalecchio di Reno (Bo) Tel.: +39 051 29 86 430 Fax: +39 051 29 86 490	Netherlands - Niederlande/Holland Bosch Rexroth Services B.V. Technical Services Kruisbroeksestraat 1 (P.O. Box 32) 5281 RV Boxtel Tel.: +31 (0) 411 65 19 51 Fax: +31 (0) 411 67 78 14 **Hotline: +31 (0) 411 65 19 51** services@boschrexroth.nl	Netherlands – Niederlande/Holland Bosch Rexroth B.V. Kruisbroeksestraat 1 (P.O. Box 32) 5281 RV Boxtel Tel.: +31 (0) 411 65 16 40 Fax: +31 (0) 411 65 14 83 www.boschrexroth.nl
Norway - Norwegen Bosch Rexroth AS Electric Drives & Controls Berghagan 1 or: Box 3007 1405 Ski-Langhus 1402 Ski Tel.: +47 64 86 41 00 Fax: +47 64 86 90 62 **Hotline: +47 64 86 94 82** jul.ruud@rexroth.no	Spain – Spanien Goimendi Automation Parque Empresarial Zuatzu C/ Francisco Grandmontagne no.2 20018 San Sebastian Tel.: +34 9 43 31 84 21 - service: +34 9 43 31 84 56 Fax: +34 9 43 31 84 27 - service: +34 9 43 31 84 60 sat.indramat@goimendi.es	Spain - Spanien Bosch Rexroth S.A. Electric Drives & Controls Centro Industrial Santiga Obradors s/n 08130 Santa Perpetua de Mogoda Barcelona Tel.: +34 9 37 47 94 00 Fax: +34 9 37 47 94 01	Spain - Spanien Bosch Rexroth S.A. Electric Drives & Controls c/ Almazara, 9 28760 Tres Cantos (Madrid) Tel.: +34 91 806 24 79 Fax: +34 91 806 24 72 fernando.bariego@boschrexroth.es
Sweden - Schweden Bosch Rexroth AB Electric Drives & Controls - Varuvägen 7 (Service: Konsumentvägen 4, Älfsjö) 125 81 Stockholm Tel.: +46 (0) 8 727 92 00 Fax: +46 (0) 8 647 32 77	Sweden - Schweden Bosch Rexroth AB Electric Drives & Controls Ekvändan 7 254 67 Helsingborg Tel.: +46 (0) 4 238 88 -50 Fax: +46 (0) 4 238 88 -74	Switzerland East - Schweiz Ost Bosch Rexroth Schweiz AG Electric Drives & Controls Hemrietstrasse 2 8863 Buttikon Tel. +41 (0) 55 46 46 111 Fax +41 (0) 55 46 46 222	Switzerland West - Schweiz West Bosch Rexroth Suisse SA Av. Général Guisan 26 1800 Vevey 1 Tel.: +41 (0)21 632 84 20 Fax: +41 (0)21 632 84 21

11.5.3 Europa (Ost) - Europe (East)

vom Ausland: (0) nach Landeskennziffer weglassen
from abroad: don't dial (0) after country code

Czech Republic - Tschechien Bosch -Rexroth, spol.s.r.o. Hviezdoslavova 5 627 00 Brno Tel.: +420 (0)5 48 126 358 Fax: +420 (0)5 48 126 112	Czech Republic - Tschechien DEL a.s. Strojírenská 38 591 01 Zdar nad Sázavou Tel.: +420 566 64 3144 Fax: +420 566 62 1657	Hungary - Ungarn Bosch Rexroth Kft. Angol utca 34 1149 Budapest Tel.: +36 (1) 422 3200 Fax: +36 (1) 422 3201	Poland – Polen Bosch Rexroth Sp.zo.o. ul. Staszica 1 05-800 Pruszków Tel.: +48 (0) 22 738 18 00 – service: +48 (0) 22 738 18 46 Fax: +48 (0) 22 758 87 35 – service: +48 (0) 22 738 18 42
Poland – Polen Bosch Rexroth Sp.zo.o. Biuro Poznan ul. Dabrowskiego 81/85 60-529 Poznan Tel.: +48 061 847 64 62 /-63 Fax: +48 061 847 64 02	Romania - Rumänien East Electric S.R.L. Bdul Basarabia no.250, sector 3 73429 Bucuresti Tel./Fax:: +40 (0)21 255 35 07 +40 (0)21 255 77 13 Fax: +40 (0)21 725 61 21 eastel@rdsnet.ro	Romania - Rumänien Bosch Rexroth Sp.zo.o. Str. Drobety nr. 4-10, app. 14 70258 Bucuresti, Sector 2 Tel.: +40 (0)1 210 48 25 +40 (0)1 210 29 50 Fax: +40 (0)1 210 29 52	Russia - Russland Bosch Rexroth OOO Wjatskaja ul. 27/15 127015 Moskau Tel.: +7-095-785 74 78 +7-095 785 74 79 Fax: +7 095 785 74 77 laura.kanina@boschrexroth.ru
Russia Belarus - Weissrussland ELMIS 10, Internationalnaya 246640 Gomel, Belarus Tel.: +375/ 232 53 42 70 +375/ 232 53 21 69 Fax: +375/ 232 53 37 69 elmis_ltd@yahoo.com	Turkey - Türkei Bosch Rexroth Otomasyon San & Tic. A..S. Fevzi Cakmak Cad No. 3 34630 Sefaköy Istanbul Tel.: +90 212 413 34 00 Fax: +90 212 413 34 17 www.boschrexroth.com.tr	Turkey - Türkei Servo Kontrol Ltd. Sti. Perpa Ticaret Merkezi B Blok Kat: 11 No: 1609 80270 Okmeydani-Istanbul Tel: +90 212 320 30 80 Fax: +90 212 320 30 81 remzi.sali@servokontrol.com www.servokontrol.com	Slowenia - Slowenien DOMEL Otoki 21 64 228 Zelezniki Tel.: +386 5 5117 152 Fax: +386 5 5117 225 brane.ozebek@domel.si

11.5.4 Afrika, Asien, Australien (inkl. Pazifischer Raum) - Africa, Asia, Australia (incl. Pacific Rim)

Australia - Australien	Australia - Australien	China	China
AIMS - Australian Industrial Machinery Services Pty. Ltd. 28 Westside Drive Laverton North Vic 3026 Melbourne Tel.: +61 3 93 14 3321 Fax: +61 3 93 14 3329 **Hotlines: +61 3 93 14 3321** **+61 4 19 369 195** enquires@aimservices.com.au	Bosch Rexroth Pty. Ltd. No. 7, Endeavour Way Braeside Victoria, 31 95 Melbourne Tel.: +61 3 95 80 39 33 Fax: +61 3 95 80 17 33 mel@rexroth.com.au	Shanghai Bosch Rexroth Hydraulics & Automation Ltd. Waigaoqiao, Free Trade Zone No.122, Fu Te Dong Yi Road Shanghai 200131 - P.R.China Tel.: +86 21 58 66 30 30 Fax: +86 21 58 66 55 23 richard.yang_sh@boschrexroth.com.cn gf.zhu_sh@boschrexroth.com.cn	Shanghai Bosch Rexroth Hydraulics & Automation Ltd. 4/f, Marine Tower No.1, Pudong Avenue Shanghai 200120 - P.R.China Tel: +86 21 68 86 15 88 Fax: +86 21 58 40 65 77
China	China	China	China
Bosch Rexroth China Ltd. 15/F China World Trade Center 1, Jianguomenwai Avenue Beijing 100004, P.R.China Tel.: +86 10 65 05 03 80 Fax: +86 10 65 05 03 79	Bosch Rexroth China Ltd. Guangzhou Repres. Office Room 1014-1016, Metro Plaza, Tian He District, 183 Tian He Bei Rd Guangzhou 510075, P.R.China Tel.: +86 20 8755-0030 +86 20 8755-0011 Fax: +86 20 8755-2387	Bosch Rexroth (China) Ltd. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023, P.R.China Tel.: +86 411 46 78 930 Fax: +86 411 46 78 932	Melchers GmbH BRC-SE, Tightening & Press-fit 13 Floor Est Ocean Centre No.588 Yanan Rd. East 65 Yanan Rd. West Shanghai 200001 Tel.: +86 21 6352 8848 Fax: +86 21 6351 3138
Hongkong	India - Indien	India - Indien	India - Indien
Bosch Rexroth (China) Ltd. 6th Floor, Yeung Yiu Chung No.6 Ind Bldg. 19 Cheung Shun Street Cheung Sha Wan, Kowloon, Hongkong Tel.: +852 22 62 51 00 Fax: +852 27 41 33 44 alexis.siu@boschrexroth.com.hk	Bosch Rexroth (India) Ltd. Electric Drives & Controls Plot. No.96, Phase III Peenya Industrial Area Bangalore – 560058 Tel.: +91 80 51 17 0-211...-218 Fax: +91 80 83 94 345 +91 80 83 97 374 mohanvelu.t@boschrexroth.co.in	Bosch Rexroth (India) Ltd. Electric Drives & Controls Advance House, II Floor Ark Industrial Compound Narol Naka, Makwana Road Andheri (East), Mumbai - 400 059 Tel.: +91 22 28 56 32 90 +91 22 28 56 33 18 Fax: +91 22 28 56 32 93 singh.op@boschrexroth.co.in	Bosch Rexroth (India) Ltd. S-10, Green Park Extension New Delhi – 110016 Tel.: +91 11 26 56 65 25 +91 11 26 56 65 27 Fax: +91 11 26 56 68 87 koul.rp@boschrexroth.co.in
Indonesia - Indonesien	Japan	Japan	Korea
PT. Bosch Rexroth Building # 202, Cilandak Commercial Estate Jl. Cilandak KKO, Jakarta 12560 Tel.: +62 21 7891169 (5 lines) Fax: +62 21 7891170 - 71 rudy.karimun@boschrexroth.co.id	Bosch Rexroth Automation Corp. Service Center Japan Yutakagaoka 1810, Meito-ku, NAGOYA 465-0035, Japan Tel.: +81 52 777 88 41 +81 52 777 88 53 +81 52 777 88 79 Fax: +81 52 777 89 01	Bosch Rexroth Automation Corp. Electric Drives & Controls 2F, I.R. Building Nakamachidai 4-26-44, Tsuzuki-ku YOKOHAMA 224-0041, Japan Tel.: +81 45 942 72 10 Fax: +81 45 942 03 41	Bosch Rexroth-Korea Ltd. Electric Drives and Controls Bongwoo Bldg. 7FL, 31-7, 1Ga Jangchoong-dong, Jung-gu Seoul, 100-391 Tel.: +82 234 061 813 Fax: +82 222 641 295
Korea	Malaysia	Singapore - Singapur	South Africa - Südafrika
Bosch Rexroth-Korea Ltd. 1515-14 Dadae-Dong, Saha-gu Electric Drives & Controls Pusan Metropolitan City, 604-050 Tel.: +82 51 26 00 741 Fax: +82 51 26 00 747 eunkyong.kim@boschrexroth.co.kr	Bosch Rexroth Sdn.Bhd. 11, Jalan U8/82, Seksyen U8 40150 Shah Alam Selangor, Malaysia Tel.: +60 3 78 44 80 00 Fax: +60 3 78 45 48 00 hhlim@boschrexroth.com.my rexroth1@tm.net.my	Bosch Rexroth Pte Ltd 15D Tuas Road Singapore 638520 Tel.: +65 68 61 87 33 Fax: +65 68 61 18 25 sanjay.nemade @boschrexroth.com.sg	TECTRA Automation (Pty) Ltd. 100 Newton Road, Meadowdale Edenvale 1609 Tel.: +27 11 971 94 00 Fax: +27 11 971 94 40 **Hotline: +27 82 903 29 23** georgv@tectra.co.za
Taiwan	Taiwan	Thailand	
Bosch Rexroth Co., Ltd. Taichung Industrial Area No.19, 38 Road Taichung, Taiwan 407, R.O.C. Tel : +886 - 4 -235 08 383 Fax: +886 - 4 -235 08 586 jim.lin@boschrexroth.com.tw david.lai@boschrexroth.com.tw	Bosch Rexroth Co., Ltd. Tainan Branch No. 17, Alley 24, Lane 737 Chung Cheng N.Rd. Yungkang Tainan Hsien, Taiwan, R.O.C. Tel : +886 - 6 –253 6565 Fax: +886 - 6 –253 4754 charlie.chen@boschrexroth.com.tw	NC Advance Technology Co. Ltd. 59/76 Moo 9 Ramintra road 34 Tharang, Bangkhen, Bangkok 10230 Tel.: +66 2 943 70 62 +66 2 943 71 21 Fax: +66 2 509 23 62 Hotline +66 1 984 61 52 sonkawin@hotmail.com	

11.5.5 Nordamerika - North America

USA Headquarters - Hauptniederlassung Bosch Rexroth Corporation Electric Drives & Controls 5150 Prairie Stone Parkway Hoffman Estates, IL 60192-3707 Tel.: +1 847 645-3600 Fax: +1 847 645-6201 servicebrc@boschrexroth-us.com repairbrc@boschrexroth-us.com	USA Central Region - Mitte Bosch Rexroth Corporation Electric Drives & Controls 1701 Harmon Road Auburn Hills, MI 48326 Tel.: +1 248 393-3330 Fax: +1 248 393-2906	USA Southeast Region - Südost Bosch Rexroth Corporation Electric Drives & Controls 2810 Premiere Parkway, Suite 500 Duluth, GA 30097 Tel.: +1 678 957-4050 Fax: +1 678 417-6637	**USA SERVICE-HOTLINE** - 7 days x 24hrs - **+1-800-REXROTH** **+1 800 739-7684**
USA Northeast Region – Nordost Bosch Rexroth Corporation Electric Drives & Controls 99 Rainbow Road East Granby, CT 06026 Tel.: +1 860 844-8377 Fax: +1 860 844-8595	USA West Region – West Bosch Rexroth Corporation Electric Drives & Controls 7901 Stoneridge Drive, Suite 220 Pleasanton, CA 94588 Tel.: +1 925 227-1084 Fax: +1 925 227-1081		
Canada East - Kanada Ost Bosch Rexroth Canada Corporation Burlington Division 3426 Mainway Drive Burlington, Ontario Canada L7M 1A8 Tel.: +1 905 335 5511 Fax: +1 905 335 4184 michael.moro@boschrexroth.ca	Canada West - Kanada West Bosch Rexroth Canada Corporation 5345 Goring St. Burnaby, British Columbia Canada V7J 1R1 Tel. +1 604 205 5777 Fax +1 604 205 6944 david.gunby@boschrexroth.ca	Mexico Bosch Rexroth Mexico S.A. de C.V. Calle Neptuno 72 Unidad Ind. Vallejo 07700 Mexico, D.F. Tel.: +52 55 57 54 17 11 Fax: +52 55 57 54 50 73 mario.francioli@boschrexroth.com.mx	Mexico Bosch Rexroth S.A. de C.V. Calle Argentina No 3913 Fracc. las Torres 64930 Monterrey, N.L. Tel.: +52 81 83 65 22 53 +52 81 83 65 89 11 +52 81 83 49 80 91 Fax: +52 81 83 65 52 80

11.5.6 Südamerika - South America

Argentina - Argentinien Bosch Rexroth S.A.I.C. "The Drive & Control Company" Rosario 2302 B1606DLD Carapachay Provincia de Buenos Aires Tel.: +54 11 4756 01 40 +54 11 4756 02 40 +54 11 4756 03 40 +54 11 4756 04 40 Fax: +54 11 4756 01 36 +54 11 4721 91 53 victor.jabif@boschrexroth.com.ar	Argentina - Argentinien NAKASE Servicio Tecnico CNC Calle 49, No. 5764/66 B1653AOX Villa Balester Provincia de Buenos Aires Tel.: +54 11 4768 36 43 Fax: +54 11 4768 24 13 **Hotline: +54 11 155 307 6781** nakase@usa.net nakase@nakase.com gerencia@nakase.com (Service)	Brazil - Brasilien Bosch Rexroth Ltda. Av. Tégula, 888 Ponte Alta, Atibaia SP CEP 12942-440 Tel.: +55 11 4414 56 92 +55 11 4414 56 84 Fax sales: +55 11 4414 57 07 Fax serv.: +55 11 4414 56 86 alexandre.wittwer@rexroth.com.br	Brazil - Brasilien Bosch Rexroth Ltda. R. Dr.Humberto Pinheiro Vieira, 100 Distrito Industrial [Caixa Postal 1273] 89220-390 Joinville - SC Tel./Fax: +55 47 473 58 33 Mobil: +55 47 9974 6645 prochnow@zaz.com.br
Columbia - Kolumbien Reflutec de Colombia Ltda. Calle 37 No. 22-31 Santafé de Bogotá, D.C. Colombia Tel.: +57 1 368 82 67 +57 1 368 02 59 Fax: +57 1 268 97 37 reflutec@etb.net.co			

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

R911310666

Printed in Germany
DOK-SUPPL*-VIC*BED*02-AW02-EN-P