Operating and Display Units

Control Terminal BT2 / BT5N / BT20N Programming (Part 1 and Part 2)

Edition 101

BOSCH
Automation

Operating and Display Units

Control Terminal BT2 / BT5N / BT20N Programming (Part 1 and Part 2)

1070 083 630-101 (00.05) GB

Software release: 1.3

Programming

TesiMod

Operating Terminals

Version 1.3 issued 17th April 2000

First edition	Version 1.0	issued 1st May 1996
Enlarged edition	Version 1.1	issued 16th September 1997
Revised edition	Version 1.2	issued 17th November 1998
Revised edition	Version 1.3	issued 17th April 2000

Table of Contents

1 TesiMod Operating Concept

1.1 Operating Terminals

All of the terminals offer the same functions, they differ only in design, display size and number of function keys. The aluminium front panel and zinc-coated plate steel casing ensure noise immunity, making the terminals suitable for employment even in a harsh industrial environment. All of the terminals are provided with an IP65 degree of protection on account of the sealed front mounting.

Extensive tests have proven the terminals extremely reliable and ideal for industrial applications. Tests such as mechanical strength tests, electromagnetic compatibility tests, temperature and climate tests ensure a high quality of the products.

All terminals are equipped with highly readable, multiple-line, backlit or luminescent displays (up to 16 lines and 42 characters) and are thus suitable for a wide scope of applications. The keyboard is protected by a front foil and is provided with mechanical short-stroke keys with defined tactile touch. Function keys with integrated LEDs eliminate the need for additional external switches and pilot lamps. All function keys are supplied with blank identification strips for individual labelling.

The communication with the terminals is supported by standardized interfaces. A version with PROFIBUS-DP interface is available for each terminal.

1.1.1 Loadable Protocol Drivers

With a standard terminal it is possible to exchange data with any of the PLCs which are currently supported and which is not a fieldbus protocol. The interface standards TTY / 20 mA Current Loop, RS485 and RS232c are combined in one connector.
For fieldbus connection you must order a terminal with the special interface PROFIBUS-DP.

The applications of all terminals include output, input and modification of data in PLCs, microprocessor-based controllers, process computers or PCs.

Communication occurs either by means of the controller-specific protocol or a standard protocol. A controller-specific protocol ensures access to all system and user data as well as inputs and outputs, flags, timers, counters etc. In case of connections to PLCs, the terminals are, in general, connected via the programming unit interface.

1.1.2 Networking the Terminals

Multipoint connections are supported, provided the transmission protocols allow this method of connection. With some protocols it is possible to address multiple CPUs on the network or in the mounting rack. Only fieldbus connection allow multiple operating terminals to be connected to one PLC.

The DIN-Meßbus in accordance with the DIN 66348 offers a more cost-effective solution for such requirements. Multipoint connections of this type are configured such that one operating terminal operates as bus master and as gateway to the controller. All other operating terminals operate as DIN-Meßbus slave stations. Addressing of the data occurs in accordance with the syntax of the connected controller. The bus protocol allows access to all standard functions of the TesiMod. Appropriate measures ensure a high efficiency between the two protocols used. Every operating terminal can be operated as bus master. The interface X3 of terminals operating as bus master is, however, not available for connecting a printer but is required for connecting the bus (RS485 interface). This connection can be carried out in several different ways, depending on the terminal type.

1.1.3 Programming Unit Interface

Since hardware and protocol software are already available in every PLC for the programming unit, there is no need for an additional communications processor, thus providing for a low-cost connection of the terminal to the PLC. The application programmer is not required to develop and install communications software for his PLC as would be the case with a communications module. The protocol software is integrated into the PLC's and terminal's operating system.

1.1.4 What is TesiMod and which functions does it provide?

With TesiMod, the Sütron electronics company has designed an operating concept offering every technical feature of an advanced operating technology. It is a standardized and functionally uniform operating concept which completely frees the connected controller from operating tasks such as operator guidance and data display. In standard mode, decoding of the keys or menu control through the connected controller is not required.

1.1.5 Operating Modes

Both modes of operation, the standard as well as the transparent mode of operation, are available in every TesiMod operating terminal.

1.1.5.1 Transparent-Mode

TesiMod terminals operating in the transparent mode represent full-size ANSI-terminals. Every key generates a press and release code which is transmitted via the interface. ESC-sequences are used for controlling the display and LEDs. Various character sets and attributes are available depending on the type of display.

1.1.5.2 Standard-Mode

The full performance of the terminal is reached in standard mode. The key decoding or the menu control with the connected PLC is not required in standard mode. Without much control requirement a operator guidance with password administration, recipe administration, scaling of variables, graphic elements, tables and messages is made. The user can be guide through the operating structure with softkeys, selection menus, funktion keys or the PLC (compulsory).

1.1.6 Costomized Definition

The project data of the customized definition consist principally of three kinds of data.

Global data, valid for the whole project.
Language specific data, only valid for a fixed language in a project.
PLC specific data, only valid for the selected PLC in a project.

Global data:
- System configurations
- Graphics

Language specific data:
- Mask definition
- Text lists
- Help text (help masks)

PLC specific data:
- Communication configurations
- Variable list

The splitting of the project data in three parts makes it easier to enlarge the project with a mask definition in an additional language or to change the connected PLC or PLC producer. The whole customized definition may be done with the confortable PC software (TSdos or TSwin).

Later the costomized definition appears for the operator only as a text, message,table,diagram,graphic and value which the terminal displays. The displayed elements are composed for one display page. This page is called mask. Mask types for different demands are developped for TSdos. In TSwin the content of the mask defines the function.

1.1.7 Storing the Customized Definition

The customized definition of all operating and terminal functions is stored in a FLASH-Memory. Programming and erasing of this memory occurs directly in the terminal. A programming unit is not needed for this process. The FLASH-Memory is programmable while being mounted in the terminal. It is not necessary to open the terminal.

It is, of course, possible to replace the FLASH-Memory with a UV-EPROM. UV-EPROMS can, however, neither be programmed nor be erased in the terminal. The FLASH-EPROM offers a power failure safe storage of masks and operator guidance without battery!

1.1.8 Addressing of the Variables

The variable listing serves as a reference listing for the connection to the controller hardware. This is were the symbolic names of the variables are assigned hardware addresses such as flags, inputs, outputs or data words. The assignment occurs in the notation of the corresponding controller manufacturer.

1.1.9 Data Release

To change a value in the terminal the data release must be grant. It is possible to aktivate the data release for all variables in a mask automatically (mask parameters).
If the data release is not automatically activated, the operator must request it with the data release key at the terminal. The status-LED of the data release key shows if the data release is grant (LED on) or not (LED flashing).
The PLC can deny the data release (external data release). For that create variables for the poll area (write coordination byte) and the read coordination byte. The bits in the variables must set or reset appropriate.

1.1.10 Password Protection

It is possible to assign up to eight passwords to protect data and the contents of masks. Each password is allocated a view and an edit level. Each mask is allocated a specific access level. Depending on the password levels of the user, access to masks or even entire menu trees may be allowed or restricted. Any information contained in a mask is subject to the same access authorization. The access authorization of the passwords is defined in the TS software. The access authorization is reset when the terminal is initialized through the operator, from within the controller or when the password entered is incorrect. Default passwords and access levels are defined during the programming of the mask definition. The modification of passwords is possible in the operating terminal. A master password can be defined which allows the default values to be restored. All masks and variables are accessible if a password is not assigned.

1.1.11 Editors

In the I/O-mask it is not only possible to modify controller variables, it is also allows the modification of terminal-specific system variables.

For every type of data, a correspondingly powerful editor is available. All variables are checked for plausibility as they are entered. The limits are determined by the user in the list of variables. In addition, the user has the option of defining a variable-specific help text of the size of one display. This text may, for example, contain a description of the function of variables and their permissible range of values.

Variables are entered as follows: if the I/O mask contains an editable value, it is possible to switch to the editor by pressing the data release key (provided the external data release is given by the PLC and in case of password protection, the correct code word has been entered). Once in the edit mode, the LED of the data release key will light up and the cursor will be located on the first editable variable. Now a modification of this value is possible through the numerical keypad including the sign keys, decimal point and delete key. After pressing the enter key, the value is accepted. The value will then be checked for plausibility and a system error message will be generated, if necessary.

Particularly powerful editors such as the table editor and the selection field editor allow the handling of larger amounts of data. The table editor permits entire columns of subscribed variables to be edited. For the column editor type, a large variety of individual editors are available for selection such as the fixed point editor, floating point editor, coded text editor etc. It is also possible to select different variable sizes and variable types for each column.

Powerful editors require additional key functions such as PgUp, PgDn, column left, column right etc. They are generated individually by means of system variables and soft keys.

1.1.12 Help Texts

A display-sized help text can be assigned to every mask and every editable value. If no such text has been assigned, a default text applicable for the entire system will be displayed. The default help text for masks and variables can be defined in the TS software.

A flashing help key indicates either a malfunction or receipt of a message. Upon pressing the help key, the system message received will be displayed.
In case of an error, the enter key can not be used to exit the editing field, it can only be used to enter a valid value.

The cursor keys, however, can be used to exit an editing field even if the value is invalid. In this case, however, the value which was entered last will be replaced by the old value (i.e. the original value is restored). Within the editing mode, the cursor keys allow selection of editable fields. The editing mode can be exited by pressing the data release key once more (the LED is off).

1.1.13 Function Keys

The function keys and their LEDs play an important role in the TesiMod operating concept, in addition to the masks.
Individually labelled identification stripes are available for designating the function keys as desired. The unit is delivered with labelled identification strips and blank identification strips. The function keys are programmable by the user. They can be used either as direct selector keys for choosing further masks and/or as control keys in the machine. The controller will evaluate the LEDs integrated into the terminal´s cursor keys as functional feedback.

Function keys programmed as direct selector keys allow direct access to the masks or entire menu structures behind those masks. In addition, they permit the experienced operator a more speedy operation compared with the menu structure.

It is also possible to assign function keys default assignments for an entire mask definition. However, this does not exclude a simultaneous use of the function keys as soft keys. The definition as soft keys possesses a higher priority than the default function.

Another factor in providing a more speedy operation is the editor. The editor memorizes the position in a mask which was edited last and upon activating the mask again, the editor will return to this position.

1.1.14 Soft Keys

All function keys can also used as soft keys. Soft keys can change the function in one mask. The topical function must be displayed on the Display. For example: it is possible to switch a pump on and off with the same soft key.

1.1.15 Variables

The type of variables allowed such as bit, byte, word, double word or field is governed by the connected controller. The concept allows direct as well as indirect read and write access to all types of data available within a controller.

Scaling and formatting of the output and input of a variable is possible. The following data formats are available: binary, integer, fixed point, floating point, alphanumerical and coded text. All data formats and variable types can be used for input and output. The scaling facilitates the conversion of millimeter to inch, degree Celsius to degree Fahrenheit etc.

The input is possible in different formats. The output of variables can occur once or cyclic. The output of editable variables can occur once or cyclic, too. The cycle time can be set via system parameters within the programming system.

1.1.16 Graphic

The employment of the graphic displays is sufficient for easy graphic process visualising. The enlarged diagnose possibilities, the easier and clearer process control and the language independence of the symbols are important.
Partly the translation in other languages is not necessary. Graphics enables an operator guidance with pictograms and its possible to make graphic copys. The graphic display send a quick information with the supported trend displays if no exact numeric data required. Its easy understandable to visualize the reaction of the operator, the PLC and the process. The used graphic provides a quick overview about the process status in the unit or in parts of the unit. The continue of this possibilitys is almost leading to a textless, symbolic operator guidance.

A view Examples for graphic use:

- Graphic copy of processes
- Liquid level indicators
- Inpurity display in the machine image
- Display a frequency distribution
- Thermal characteristics
- Transient reactions
- Trend display
- View the company logo
- Unified mask backround

All texts, variables and graphics can be positioned anywhere.
Its possible to create a graphic in all the programs which supported from Windows95 and WindowsNT. The User has the option to choose between embedded objects and seperate administrated objects.

1.1.17 Recipes and Data Records

With the term "recipes" we refer to the storage of data records in the operating terminal. Recipes consist of the selected components and customizable texts.
The creation of the contents of the recipes is subject to the same options available for the masks.
The number of recipes in an operating terminal is theoretically limited to a maximum of 250 recipes.
A recipe may contain maximum 250 data records.
Every data record may contain:
255 lines with
255 variables and
255 text elements.
The limits of the system also depend on the memory available.

Recipes differ from masks in that they can be edited in an I/O mask within a selectable recipe window regardless of their length. Setting of the position and size of the recipe window is possible in the TS software.

A mask may contain other input and output variables, in addition to the recipe. Recipes may contain

input variables only. The variables are stored in the battery-backed CMOS-RAM of the operating terminal. The input variables can be formatted, scaled and be assigned help texts just like mask variables.

Data records predefined in the TS software are stored in the Flash and can be copied into the RAM to be modified. Data records can be created, deleted, copied within the operating terminal and be transferred from and to the PLC. The data records are handled in the terminal memory in a dynamic fashion thus making optimal use of the memory area available.

The transmission of the data records to the controller can be effected either directly or via a communications buffer. The transmission of the data record to the operating terminal is effected in a direct manner. The handling of the recipes and addressing of the variables occurs independently from the language. The texts in the recipes are retrieved from the language-specific mask definition. The data records of a recipe are printable with the interface X3. It is posssible to transfer the data from PC to terminal and from terminal to PC with system variables.

1.1.18 Messages

Messages are operating states which transmitting to the terminal. Messages are stored in the message memory either according to their priority or in chronological order. The current software version allows management of approximately 3000 messages within the message buffer. The maximum size of the message memory can be predefined in the TS software. Statistics, messages, recipes and loadable character sets are stored in the CMOS-RAM memory.
There are two methods of transmitting messages to the operating terminals, one being the parallel and the other being the sequential transmission. Both systems are treated with equal priority and can be used simultaneously. In addition to the message text, it is also possible to integrate messages of the size of one display into the system. Display-sized messages can be created and used like masks.

Every message text may contain one formatted and scaled variable. Upon receipt of a message, the current value of the variable is entered into the protocol storage. This value will not be updated.

There is the option of processing the entire contents of the message memory or of processing individual blocks or lines of the message memory. The second interface allows the entire message including message number, date, time, message text and variable value to be printed out. The print-out can be obtained directly but may also be obtained at a later point of time.

1.1.19 System Messages

System messages are generated by the operating terminal as a result of various monitoring functions. The texts of these system messages are defined by the user. Each text may comprise up to the size of one display. Upon generation of a system message, the help key LED will begin to flash thereby providing an optical signal. "Blank" system messages will automatically be suppressed by the system.

1.1.20 Programming System

Our extremely user-friendly and easy-to-use programming environment allows convenient use of the great variety of functions provided by our terminals. The program runs on all PCs with Windows95 or WindowsNT.

This program greatly facilitates the generation and management of your projects. If you have any additional questions or difficulties in operating this programming system, please make use of our extensive help system or call our Hotline. Even while creating the program, the mask is displayed in its true format. Using the programming system, all terminals can be programmed and combined with all protocol formats. All terminals are programmed in the same manner and offer the same functions.

The definition of the masks and representation of the variables is carried out independently of the protocol. The reference to the PLC is made only by the reference listing of the variable definition.

While generating the mask definition using the programming system, it is possible to operate the terminal without a PLC. This allows checking and testing of the masks and the operating interface directly in the terminal.

If you wish to carry out initial tests, do not hesitate to contact us for a demo-software and demo-terminal.

Do you have any more questions?

If you have additional questions we will be glad to be of assistance.

Our Hotline number is: ++49 (6062) 78-426 !

1.2 Common Features

1.2.1 Hardware

- Multiple-lined, backlit or luminescent displays
- Filters provide glare suppression and increase the contrast
- Temperature compensation of the display (if required)
- Function keys with integrated green LEDs
- Identification strips for individual labelling of the function keys
- Editing keys
- Cursor keys
- Mask memory programmable within the terminal (Flash-Memory)
- Battery-backed RAM for message buffer
- Universal interface
- TTY /20 mA, RS485, RS232c or PROFIBUS-DP
- Battery voltage monitoring function
- Real time clock with date, time
- User-mode switch

1.2.2 Software

- Functions have been standardized for all terminals
- Two operating modes: standard and transparent mode
- Application identifier
- Free defineable operating system
- Softkey function on all function keys
- Password protection with different levels
- Integrated help system
- All variable formats
- Pixelgraphic: Backround pictures, bar diagram, changing picture and curve display
- All terminals with recipes
- Loadable protocol driver for all important PLCs
- Protocol independent hardware
- Communication supervision
- Connection of several terminals at one PG interface
- PROFIBUS-DP or DIN-Meßbus connection
- One programming software for all terminals

Table of Contents

2 Operating Modes

There are two types of operating modes available in all TesiMod operating terminals, the standard mode of operation and the transparent mode of operation.

TesiMod terminals operating in the transparent mode represent full-size ANSI-terminals. Each key generates a press and release code, which will be transmitted via the serial interface in the form of an ASCII character. The displays and the key LEDs are controlled via ESC sequences. The number of character sets and character attributes varies with the type of display. For a detailed description please refer to the chapter "Transparent Mode".

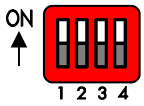
In the standard mode of operation, the entire operating system is integrated in the terminal through the TesiMod. The standardized operating concept completely frees the connected controller from any operator guidance tasks as well as data display. In standard mode, a decoding of the keys or selection of masks from within the controller is not required.

2.1 Setting the Operating Mode

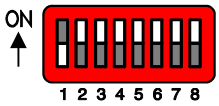
The operating mode can be set by means of the user-mode switch. The terminals are factory-set to the standard mode of operation. The user-mode switch of the BT5N is placed at the side of the unit. The user-mode switch of the other units is placed at the rear side.

User-mode switch

4 switches
BT2
BT5N



8 switches
BT20N



The switches S5 to S8 can be used by the operator as needed. The switch positions are stored at initialization time and afterwards they can be overtaken to the controller.

S1	S2	S3	S4	S5	S6	S7	S8	Function
I	X	-	-	X	X	X	X	Standard-Mode with PLC (delivery state)
I	X	I	-	X	X	X	X	Standard-Mode without PLC
-	I	-	-	X	X	X	X	Transparent-Mode with start and stop code of the keys
-	-	-	I	X	X	X	X	Transparent-Mode without stop code of the keys
I	-	-	I	X	X	X	X	Activate download (deletes application memory) und default contrast setting

Legend of above table:

I = Switch position ON
\- = Switch position OFF
X = Switch position irrelevant

/000-0108/
Bosch_T_eng_V13_3002000QK0

Table of Contents

3 TesiMod Standard Mode

In addition to the transparent mode of operation, all operating terminals incorporating the TesiMod operating concept are equipped with the standard mode of operation. As the name indicates, this is the mode of operation in which the terminals are most commonly used and that provides the highest performance. In this mode of operation, the terminals act as intelligent peripheral devices for the controller by performing controlled pre-processing of the visualization data, thus completely relieving the connected controller of all operating tasks. This reduces the programming effort required by the user, the run time and memory required in the PLC.
Communication processors have been dispensed with in the PLC and programming unit interfaces have been used throughout, making the PLC connection very economical. The operating concept is a universal one, thanks to the large number of supported protocols. Operation of the machine has been made independent of the type and manufacturer of the controller.
The standard mode is supported by every operating terminal. The system's flexibility means you can customize all menus and dialogs according to your particular application. Extremely sophisticated operator guidances can be configured with the aid of masks, system variables and control and function keys. In standard mode, all functions which can be executed with the terminal, mask contents, texts, messages and variables are stored in the user description (mask definition). After the programming phase is complete, the user description is stored in the operating terminal's Eprom or Flash memory.

In the description which follows, the term "*user*" refers to individuals who configure or program the operator interface, while "*operator*" refers to individuals who monitor and operate the data of the installation on site. In addition, a distinction is made between TSdos and TSwin with respect to their difference in the scope of programmable functions. For reasons of simplicity, the prefix "Sys" of the system variable names has been omitted.

3.1 Setting the Operating Mode

The standard mode of operation can be set with the user-mode switch. Turn off the terminal before selecting the mode of operation.
The position of the mode selector switch is described in the manual for the particular operating terminal.
All terminals are factory-set to the standard mode.
The ON/OFF positions on the mode selector switch are marked.
The selected mode of operation becomes active as soon as power is supplied.
Position of the switch for the standard mode:

S1	ON	S5	not used
S2	not used	S6	not used
S3	OFF	S7	not used
S4	OFF	S8	not used

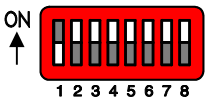


Fig. 1: Mode selector switch

3.2 Startup Process

All LEDs in the terminal are initially lit when the power is applied. A system-test is then performed that includes testing and initialization of the modules in the operating terminal. Various system and error messages may be output during this system-test. If the application memory contains a valid user description, the first mask, the Startup Mask (TSdos), or the mask specified in the language parameters as the mask to be used as the startup mask (TSwin) will appear on the display.
This mask is displayed for 5 seconds (fixed time setting). This time can be used by the operator to visually inspect the LEDs and the screen for proper functioning. After this time period, the Main Mask (TSdos) or the mask specified in the language parameters as the main mask is displayed. This mask is the first mask of the operator guidance.
If the **Enter key** is pressed while the Startup Mask is displayed, the Setup Mask will appear on the display. This Setup Mask can be used to set the parameters for the interface and the operating terminal.
If the Enter key is pressed before the Startup Mask is displayed, an error message will be generated during the keyboard test.

```
KEYBOARD ERROR
PLEASE RELEASE KEY
```

..... The self-test performed on the keyboard during the terminal startup has detected that a key is pressed. Comply with the request and release the key. If this message is generated when no key is pressed, this indicates a defective keyboard. Generally, the message will disappear after releasing the key/keys.

A longer delay may occur before the Startup Mask is displayed if the statistics memory contains a large number of messages from the serial message system. This time period (initialization period) is required to set up the structures for message management, which will then enable faster sorting of messages later. During the initialization phase, the following message is displayed:

```
INITIALIZING
CPU    XX MHz
Flash XXX kByte
XXXXXXXXX  YYYYYYYY
```

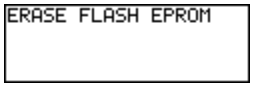
..... When the terminal is switched on, the messages that are present in the terminal are sorted. This process requires a certain length of time (initialization time).

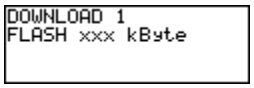
3.2.1 Startup Process without a Valid User Description

The system-test performed on the application memory will detect whether a module is missing, defective or of the wrong type. This is then indicated to the operator by the following message:

```
NO FLASH EPROM
```

If the memory test establishes that the right type of Flash memory is used but that it does not contain a valid user description, the Flash memory will be erased and the terminal will automatically switch to the download operating mode. The following messages appear on the display indicating the various phases:





The message "Download" (TSdos) or "DOWNLOAD 1" (TSwin) remains on the display to indicate that the terminal is now ready to receive a valid user description via interface X3.

Until the memory contains a valid user description, communication with a controller connected to the X2 interface will not take place, nor will the keyboard be operational. Once a valid user description has been downloaded, the terminal becomes active.

3.3 Communication in the Standard Mode

In standard mode, any interface except the interfaces for the logging printer and the parallel outputs can be used for communication between the PLC (host computer, etc.) and the operating terminal. The interface labeling, however, always depends on the connected counter part or the network.
See relevant chapters in the operating terminal manuals for a more detailed description of the various interfaces. For information on possible connections to various PLC types and networks refer to the chapters 5.x on the *controller and bus connections* in this manual.
To ensure a reliable connection, a 3 m long standard cable is available for every connection type as an accessory.

3.4 Operating Concept

The operating concept of the TesiMod terminal has been designed to allow the operator to access all masks - and thus all the data - quickly and easily.
A number of means are available to the user to help guide the operator through the hierarchical mask system.

3.4.1 Hierarchical Mask Structure in TSdos

The basic elements - the node mask and the I/O mask - provide the means for creating a hierarchically structured operator guidance. The mask parameters which can be set for each mask can not just be used to select masks but also provide access to further menus. Figure 2 shows how node masks are used to provide mask menus. The same functionality can be achieved with I/O masks which have been set up with selection texts that will call up other masks.

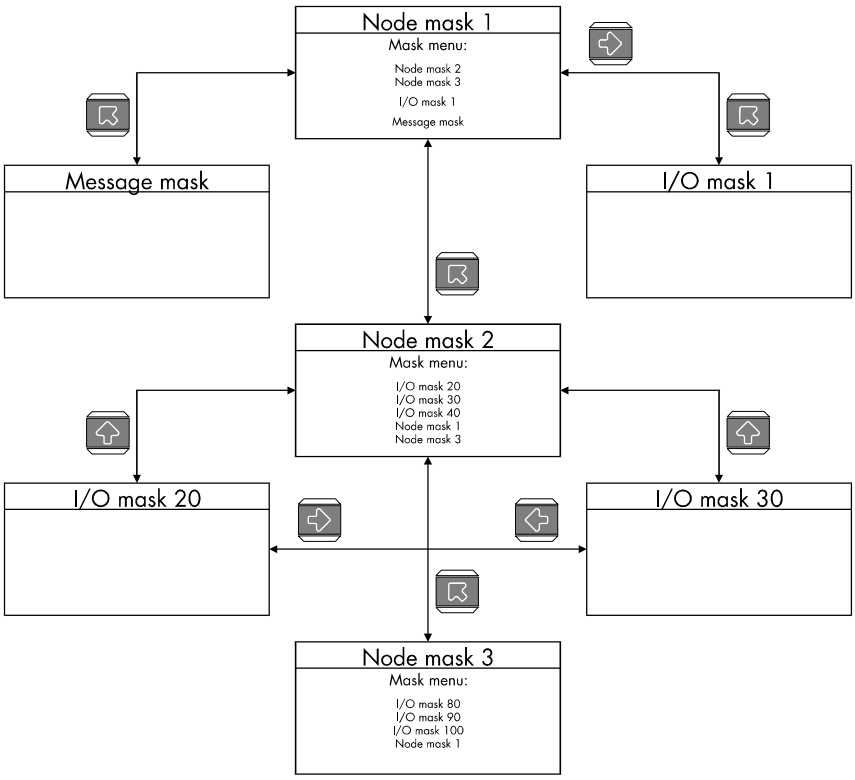


Fig. 2: Hierarchical Mask Structure

The various mask types offer the following different means for establishing a connection to other masks:

Node masks:	Cursor Right/Left/Home keys, function keys and selection items
I/O masks:	Cursor Right/Left/Home keys, function keys and selection texts
Message masks:	Cursor home key and function keys
Status message masks:	Cursor home key and function keys

The entire operator structure can be made subject to access control. Access to nodes and the masks behind this node can be prevented by implementing passwords. The direct selection of masks via function keys can also be controlled.
The controller is notified whenever a new mask is called up. Process data required by the terminal in conjunction with masks and messages are automatically obtained from the controller. If an attempt is made to call up a password-protected mask, the associated password-mask will be displayed but data will not be retrieved from the controller.

3.4.2 Mask Structure in TSwin

In TSwin, a network of I/O masks without a real hierarchical structure is formed. I/O masks are located at the nodes in the network and contain a selection field which is used to call up other masks by their names. It is possible to access any other I/O mask from an I/O mask by using the control and function keys. This feature thus eliminates the difficulty of returning from message masks or status message masks.

3.4.3 External Mask Selection

Mask calls from the controller are handled in the same way as messages. This requires simply that 8000H is added to the desired mask number prior to transfer.

When the same number is assigned to a mask and message, both the mask and the message are called up during external mask selections.
This effect can be used to provide help in a mask. Otherwise, ensure that different numbers are used for masks and messages.

A mask called up externally is considered selected once the desired mask number appears in the variable <Image of Mask Number>. The acknowledgement from the sequential data channel is not a reliable confirmation of the mask output.

Example 1: Mask number and message number are not identical
A project comprises masks with numbers ranging from 100 to 200; the first message has the number 10.
Mask 118 is to be called up by the controller.
The following adding operation must be performed in the controller:
118 + 8000H = 76H + 8000H = 8076H.
The value 8076H must be written to the address of the serial message channel.
Only mask 118 is displayed.

Example 2: Mask number and message number are identical
A project comprises masks with numbers ranging from 1 to 100; the first message has the number 10.
Mask 50 and serial message 50 are to be called up by the controller.
The following adding operation must be performed in the controller:
50 + 8000H = 32H + 8000H = 8032H.
The value 8032H must be written to the address of the serial message channel.
Both the mask 50 is displayed and message 50 are written to the message buffer.

3.4.4 Password Protection, Access Authorization

The TesiMod operating concept incorporates a password protection function. Password protection prevents masks from being accessed and the data they contain from being altered without proper authorization. The protective function is available in every operating terminal. It is obtained by assigning access levels to masks and by using passwords.
Unless otherwise specified by the programmer, the access levels of all masks automatically default to the lowest level (=0), i.e. no password is required to access all masks with this access level.
Two authorization levels, referred to as the edit level and the view level, are assigned to every password.
View level means that the next mask can be viewed after entering the password but the values in it can also not be edited.

Edit level means that the mask can be viewed after entering the password and the values in it can be edited.

Up to eight different passwords with a length of up to 11 characters can be defined. When defining the passwords, the access authorizations should be entered in a hierarchical structure. Example:
- Password for the manufacturer of the system, machine, etc.
- Password for servicing on site.
- Password for the machine setter, master craftsman, foreman etc.
- Password for the system operator.

The following figure illustrates how the access levels, edit and view levels work:

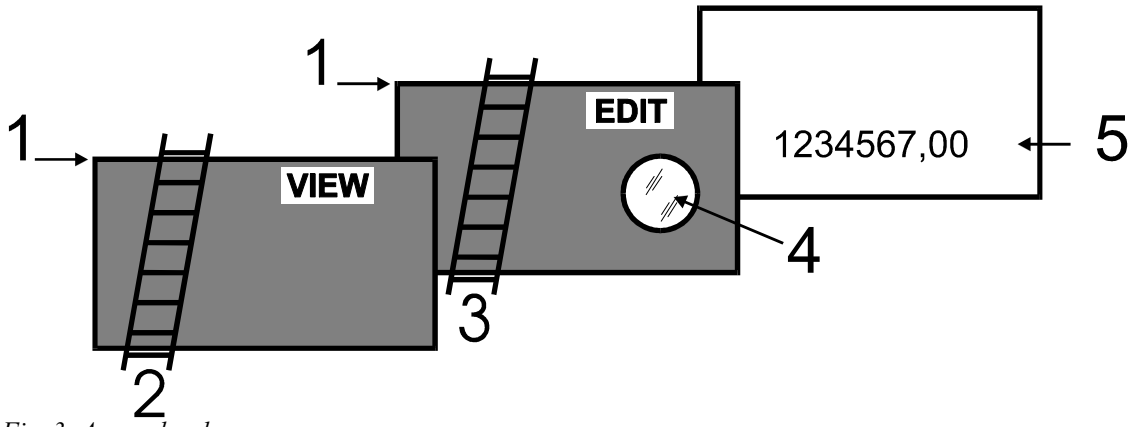


Fig. 3: Access levels

An access level (represented here by the height (1) of the walls) can be assigned to every mask and a view level (length of ladder 2) and edit level (length of ladder 3) can be assigned to every password.

A password must be entered for **viewing only** of a mask (5) that has been assigned an access level (access level is greater than 0). The password must have been assigned a view level (ladder 2) high enough to be able to get over the access level (view wall). The mask (5) can then be viewed (through window 4) but the variable values can not be edited.

For **viewing and editing** the variable values in a mask (5), a password must be supplied which has a view level (ladder 2) and an edit level (ladder 3) that are both high enough.

The following rules apply to passwords:
- Access is permitted if the view and edit level values are greater than or equal to the values specified for the access level.
- The edit level must be equal to 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 Data Release key will have no function if pressed with an insufficient edit level. The password can be entered in any I/O mask. Note that the Setup Mask is an exception. The system variable **MskchgPasswd** is designed to prompt for password entry. If the Password Editor is selected in the programming system, passwords will not appear as they are entered on the terminal. Instead the character "X" appears for every character that is entered.

Example:
Enter PASSWORD: XXXX

This example illustrates hidden entry of a 4-digit number. If an invalid password is entered, the authorization levels will automatically reset to 0.
It is recommended that at least one password - the master password - is programmed with the highest authorization level. The first password entered in the password list via the programming system is the master password and as such has a special function. The master password is unique in that it can not be changed on the operating terminal. Furthermore, it can be used to reset all modified passwords to the default values entered in the programming system.

Example:

Mask 5		access level = 10	
Mask 6		access level = 20	
Mask 7		access level = 30	
Password	4712	Edit level = 15	View level = 25

Once the password "4712" has been entered, the following accesses are permitted:
- Mask 5 will be displayed, editing of values is authorized
- Mask 6 will be displayed, editing of values is **not** authorized
- Mask 7 will **not** be displayed, editing of values is **not** authorized

The access level for the Startup Mask is always 0.
The Setup Mask is an exception with regards to the password and external data release functions. Since no communication is taking place when the Setup Mask is displayed, the external data release function is not applicable. To restrict access, passwords must be used. By defining the first editable variable in the Setup Mask as a Password Editor, all further variables can be protected against unauthorized access. The view level does not apply when accessing the Setup Mask. Viewing is always permitted if a value less than or equal to 254 is selected for the access level of the Setup Mask.
The edit level for all variables of the Setup Mask, with the exception of the Password Editor, is the same as that defined as the access level.
Access to the mask is always denied if an access level of 255 is defined for the setup mask. This means that it will no longer be displayed during initialization of the terminal and can therefore not be selected. However, all terminal-specific parameters can also be edited in any I/O mask. The new parameters become effective by restarting the terminal or with the system variable **Boot**.

3.4.4.1 Reactivating the Password Protection

The access authorization for a mask or variable (TSwin only) is reset whenever
- the operating terminal is switched off and turned back on
- an incorrect password is entered
- bit 2 in the <Write Coordination Byte> is set
- the system variable **MskchgResPasswd** is activated
- the option **Reset Password** in the mask parameters of the password-protected mask is selected.

3.4.4.2 Password Management

Passwords are stored in the operating terminal's Flash memory. These are the default passwords that are used when the terminal is started up initially after a download. At the same time, the passwords are stored in the operating terminal's RAM.
The passwords stored in the flash memory can be reactivated by writing to the system variable **FlashPasswd**.

Every password (except for the master password = first password in the list) can be edited from the terminal. For this process, the password to be edited is written to the system variable **MskchgPasswd** first. The new password is then written twice to the system variable **ChangePasswd**. If both entries for the new password are identical, the password will become effective immediately; otherwise, a system message will be displayed and the password reset. Passwords are stored and compared as 11-character strings. The passwords are entered with the Alphanumerical Editor.
Passwords are only globally programmable (not language-specifically).

3.4.4.3 Password Mask and Password Functionality

- It is possible to create a special mask which requests entry of a password (mask 3 in TSdos). This password mask is then displayed whenever an attempt is made to call up a password-protected mask without first entering a password with sufficient authorization. When a password with sufficient authorization is entered into this mask, the previously selected password-protected mask is called up once the Data Release key is pressed. There are no restrictions with respect to the remaining mask contents (texts, further variables, soft keys, etc.).
- You can specify for every mask separately whether the password protection is to be reactivated after the mask is exited.
- For those cases where no valid password has been entered, an option to exit the mask must be provided. The Cursor home key can be programmed to perform this function, for example.
- If no such mask has been created to prompt for password entry, the operator will be required to enter the password in masks specifically designed for this purpose.
- The entire password protection can be deactivated by setting the system variable **PasswdInactive** to the value 1. The operating terminal will then act as if all masks were created with an edit and view level of 0. The system variable is battery-backed, so deactivation will therefore still be in effect after the operating terminal is restarted.

3.5 Masks

In conjunction with the TesiMod operating system, the term "mask" always refers to the contents of one screen. The size of masks therefore varies from operating terminal to operating terminal. Masks with a specific functionality form the basic elements of the operator guidance. The first step when programming a mask is to specify its functionality. The process of designing the operator guidance is greatly simplified by the restricted number of different mask types.
The following mask types are available in TSdos:

- Setup mask (system mask)
- Startup mask (system mask)
- Password mask (system mask)
- Main mask (system mask)
- I/O mask (user mask)
- Node mask (user mask)
- Message mask (user mask)
- Status message mask (user mask)

3.5.1 Mask Parameters

The number of mask parameters that is available for a specific mask depends on the mask type. The mask parameters determine the functionality of the control keys in the operator guidance. Any mask parameters which are not required may remain unassigned. If no mask parameters at all are programmed, only the function keys or external mask selection can be used to exit the mask in question.
The functionality of the control keys specified in the mask parameters is subject to access control of the password system. This prevents unauthorized access to masks via the control keys.
In addition to the control key functionality, the mask parameters also contain the access level definition. The access level indicates the minimum value that a password (view level, edit level) must have to be authorized to access a mask and edit its values. The default value used for the access level is "0" (meaning free access).
A selected Automatic Data Release option in the mask parameters will allow the operator to supply input into the mask without having to press the Data Release key first.
An option for automatic reactivation of the password protection is also available in the mask parameters to ensure that the password has to be reentered when a password-protected mask is called up again.

3.5.2 System Masks

System masks facilitate initial programming. They also get your system up and running directly. This makes the initialization phase an integral part of the user description. In TSdos, fixed mask numbers are assigned to system masks. In TSwin, any mask can be selected as a system mask. System masks are basically I/O type masks with a few restrictions. These restrictions result from the operator-prompted initialization phase and the fact that communication to the controller is not yet established.

The system masks Setup Mask and Startup Mask can not be accessed via external mask selection!

The following TSdos system masks have a fixed function:

Mask 1	Setup mask
Mask 2	Startup mask
Mask 3	Password mask
Mask 4	Main mask (first user mask)

3.5.2.1 Setup Mask

Only terminal-specific parameters can be defined in the Setup Mask. This is because no communication takes place with the connected controller while the Startup Mask and Setup Mask are displayed. The external data release therefore has no function. To protect data, a password must be assigned.

From the operator guidance, the Setup Mask and Startup Mask can be reached through the function keys, provided they have not been assigned an access level. After the masks are exited, however, the terminal is not reinitialized.

Examples of terminal-specific parameters:
- Printer interface settings
- Default contrast/default intensity setting of the display
- Date and time settings
- Activation of the download function.

Fig. 4: Example of a setup mask for the BT20

3.5.2.1.1 Password Protection - Setup Mask

The Setup Mask is an exception with regard to password protection.
To password-protect the Setup Mask, the system variable **MskchgPasswd** must be set up as the first variable which can be edited in the Setup Mask. A password can be input regardless of the access level (with the exception of the access level 255).
For the setup mask, the access level applies to the edit level only, meaning that its contents are always visible to the operator.

3.5.2.1.2 Function Without the Setup Mask

If the Setup Mask is not needed, its access level can be set to the value 255, thus preventing access to the Setup Mask via the Startup Mask (using the Data Release key).

3.5.2.2 Startup Mask

The startup mask is displayed for around 5 seconds after the terminal is switched on. This is a fixed time setting; the startup process can not be changed.
With regards to the mask design, the user can only design the text to be displayed in the mask. Any combination of characters, character sizes and text attributes can be chosen in the programming system for this purpose.
Only system variables can be output in the Startup Mask. Variable input is not possible due to the time restriction. While the Startup Mask is displayed, the Setup Mask can be called by pressing the Data Release key. However, this is not possible if the access level for the setup mask is set to 255.
Some examples of information that may be chosen to be displayed in the Startup mask are listed below:

- address for servicing
- machine type
- version number of the program

Fig. 5: Example of a startup mask for the BT20

3.5.2.3 Password Mask

The password mask is an I/O type of mask. In TSdos, the mask that prompts for password entry (password mask) when attempting to call up a password-protected mask always has the mask number 3. The functionality is as illustrated in the password description.

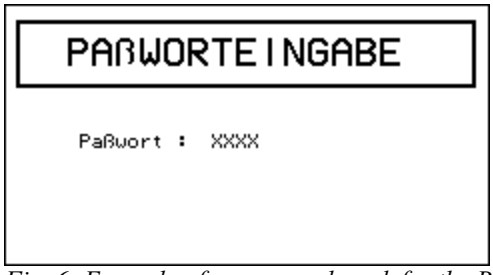


Fig. 6: Example of a password mask for the BT20

3.5.2.4 Node Mask, I/O Mask with Selection Text

The node mask as a basic element of the operator guidance structure is now available in TSdos only. This mask is no longer available in TSwin now that the same functionality can be achieved by using a selection text (linked to a text list and the system variable **NewMask**) together with the automatic data release option.
The node mask consists of a heading section and a selection part.
The heading section is used to specify the menu heading and additional information on the menu.
The number of display lines for the heading can be defined in the programming system.
The text attribute selected for the heading will automatically be applied to all of the text elements in the heading.
The selection part describes the submenus that can be selected. These submenus can be selected with the Cursor up or Cursor down key and the selection can then be confirmed with the Enter key. Access to the submenus can be protected with passwords.

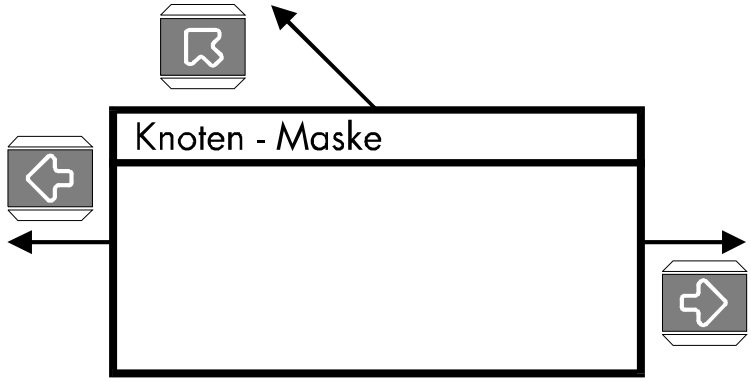


Fig. 7: Optional control keys in the node mask

A node mask can also be exited with the Cursor left, Cursor right, Cursor home keys and any function keys and soft keys that have been programmed accordingly. The cursor keys can be used for direct selection of menus at the same level. The Cursor home key may also be used to select higher-level menus.
It is not possible to input or output variables in node masks. Node masks are only used to branch to further node masks or I/O masks. To function efficiently, an operator guidance should therefore be made up of a combination of node masks and I/O masks.

Key Functions in the Node Mask

Key: Cursor up	Moves the cursor up one selection item
Key: Cursor down	Moves the cursor down one selection item
Key: Cursor left	Can be programmed freely to select adjacent masks
Key: Cursor right	Can be programmed freely to select adjacent masks
Key: Cursor home	Can be programmed freely to select adjacent masks Exits the node mask for the higher-level menu
Key: Data Release	No function
Key: Enter	Initiates the jump to the selected mask
Key: ?, Help	Displays the help text specified for the mask for as long as the key is held down

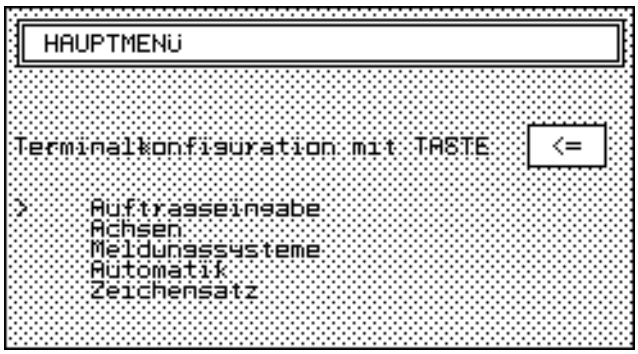


Fig. 8: Example of a node mask

3.5.2.5 I/O Mask

As a basic type of mask, the I/O mask offers a broad range of functions - enough in fact for configuring complete operator guidances based entirely on it.
I/O masks provide the following options:

- Selection of menus
- Definition of data formats
- Scaling of values
- Output of values once or cyclically
- Text display
- Message output
- Direct selection of up to five adjacent masks with the control keys
- Display of large mask contents over several screen pages
- Display of values in tables

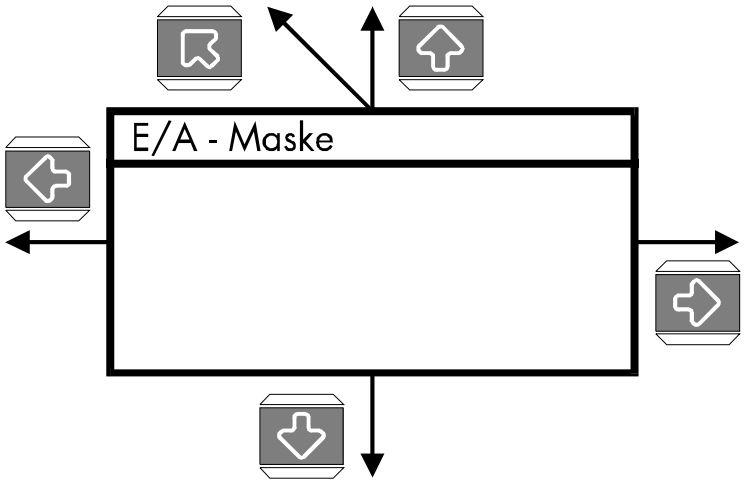


Fig. 9: Possible control keys in the I/O mask

Key Functions in the I/O Mask

Key: Cursor up	Can be programmed freely to select adjacent masks
Key: Cursor down	Can be programmed freely to select adjacent masks
Key: Cursor left	Can be programmed freely to select adjacent masks
Key: Cursor right	Can be programmed freely to select adjacent masks
Key: Cursor home	Can be programmed freely to select adjacent masks. Exits the I/O mask for the higher-level menu.
Key: Data Release	Switches into the Editor and exits the Editor
Key: Enter	Only has a function in conjunction with the Editor
Key: ?, Help	Displays the help text specified for the mask for as long as the key is held down

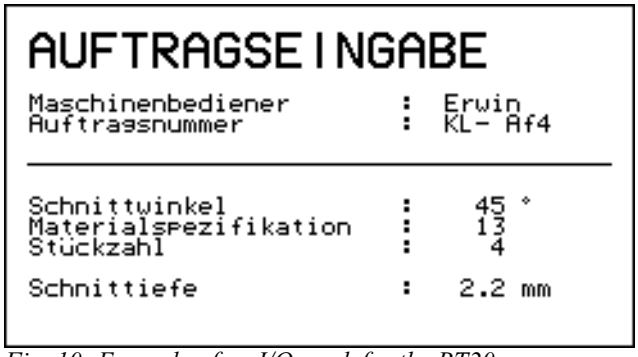


Fig. 10: Example of an I/O mask for the BT20

3.5.2.6 Message Mask, Mask with a Message Field for Serial Messages

This mask type is now only available in TSdos. In TSwin, this mask type has been replaced by the I/O mask with a message field for serial messages.
The message mask consists of a heading section and a message field.
Before a serial message is displayed on the operating terminal, the message number must be written to the poll area. A data word (two bytes) is reserved in the poll area for transferring of serial message numbers (serial message channel).
Incoming messages can be sorted directly by the operating terminal according to predefined criteria before being displayed.
Messages can optionally include the following information when displayed on the operating terminal:
- message number
- message date
- message time
- any combination.

The number of message numbers that can be stored in the operating terminal depends on the system parameter settings and the size of the RAM mask memory in the operating terminal.
Serial messages are displayed in the message mask until they are deleted by the operator from the message memory of the operating terminal.

Heading section of the mask:

The heading section is used to specify the heading and additional information, if required, on the messages (for example, information on how to view the remaining part of the message). The number of display lines for the heading can be specified in the programming system.
The text attribute selected for the heading will automatically be applied to all of the text elements in the heading. A help mask can be programmed to provide information on how to operate the message mask.

Message field of the mask:
All incoming messages are stored in the message memory and are displayed in this mask according to specific sorting criteria.
In TSdos, any messages longer than one display line on the operating terminal are truncated when displayed. The message contents are not lost, however.
To view the entire message, select the message and press the Enter key and it will be displayed in a separate mask (zoom function).
For TSwin projects, the entire message text can be displayed. The message text can, however, also be displayed in a separate mask using the zoom function.
Zooming of messages is provided as a part of the standard message mask functions. No special programming effort is necessary. (See chapter 3.9.2.2.3 *Zooming Messages*.)
The type of message representation can be determined by defining the appropriate settings in the programming system or online using system variables in a configuration mask.

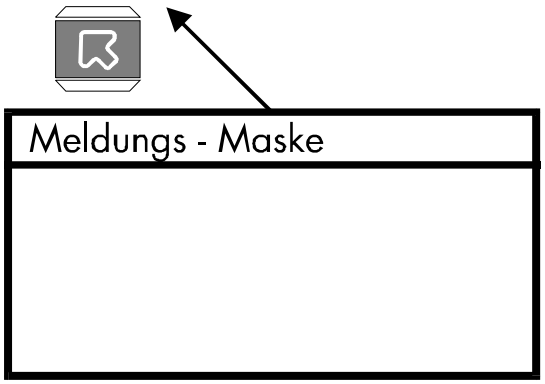


Fig. 11: Possible control keys of the message mask

Key Functions in the Message Mask

Key: Cursor home	Freely assignable to call up any mask. It exits the message mask for the higher-level menu
Key: Cursor up	Moves the cursor (>) upwards
Key: Cursor down	Moves the cursor (>) downwards
Key: Cursor right	Moves the cursor (>) down one message field
Key: Cursor left	Moves the cursor (>) up one message field
Key: ?, Help	Displays the help text specified for the mask for as long as the key is held down
Key: Enter	Zooms the message. All of the characters in a message are displayed
Key: Data Release	Switches to the Message Editor and exits it again

If the Message Editor is running, messages can be selected and then deleted with the Delete key or printed by means of the system variable **BlockPrint**. The settings used to print the messages are the same as those selected for the message display.

3.5.2.7 Status Message Mask

The status message mask offers the same functions and display options as the message mask. However, the contents of the status message mask refer primarily to message texts of the parallel message system. This type of mask is now available in TSdos only. In TSwin, this mask type has been replaced by the I/O mask with a message field for parallel messages.
Status messages are reported to the operating terminal by means of a separate data area/address range which can be freely selected. Every bit of this data area represents one status message. If the bit is set, the message is displayed on the operating terminal; once the bit has been reset, the message is removed from the display again.
Status messages can optionally include the following information when displayed on the operating terminal.
- message number
- message date
- message time
- any combination.

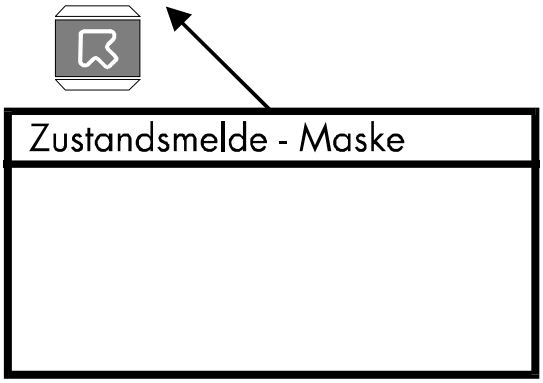


Fig. 12 : Possible control keys of the status message mask

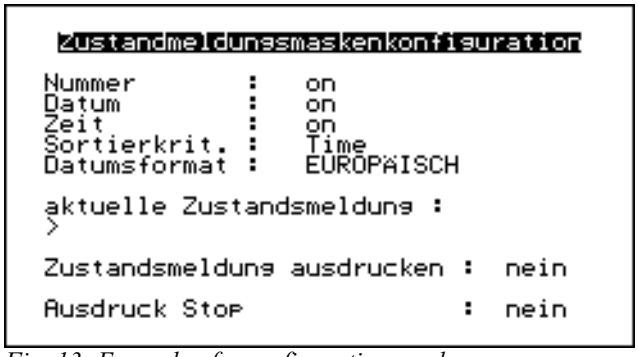


Fig. 13: Example of a configuration mask

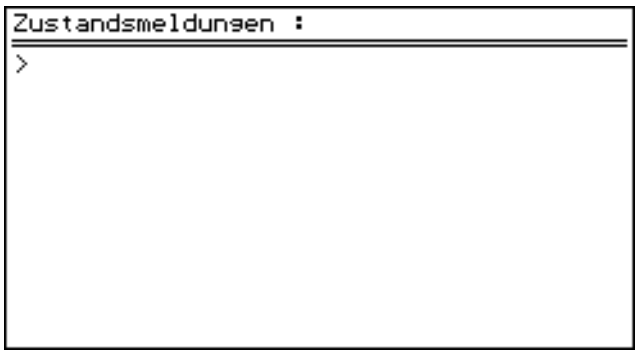


Fig. 14: Example of a status message mask

Key Functions in the Status Message Mask

Key: Cursor home	Freely assignable to call up any mask. It exits the Message Mask for the higher-level menu
Key: Cursor up	Moves the cursor (>) upwards
Key: Cursor down	Moves the cursor (>) downwards
Key: Cursor right	Moves the cursor (>) down one message field
Key: Cursor left	Moves the cursor (>) up one message field
Key: ?, Help	Displays the help text specified for the mask for as long as the key is held down
Key: Enter	Zooms the message. All of the characters in a message are displayed.
Key: Data Release	Switches to the Message Editor and exits it. If the Message Editor is running, messages can be selected and then be deleted with the Delete key or printed by means of the system variable **BlockPrint**. The settings used to print the messages are the same as those selected for the message display.

3.6 Variables

The number of valid variable types is determined by the connected controller. All standard variable types which are commonly used are supported by the operating terminals. The data type that is used also determines the valid range of values and number of significant digits.

Variable 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	[0to 65535]
Lword	4 bytes	[-2147483648 to +2147483647]
Lword	4 bytes	[0 to 4294967295]
Lword	4 bytes	[$\pm 1.2 * 10^{-38}$ to $\pm 3.4 * 10^{+38}$]
ASCII	42 bytes	[0 to 255]

3.6.1 Output Variables

Output variables are numerical or alphanumerical data stored in the memory of the connected controller. The variables are retrieved from the controller whenever required, and output on the display at the programmed location according to the defined method of representation, formatting and scaling. Generally, a distinction is made between displaying one-time variables (variables are transferred only once and are then displayed) or at cyclic intervals (variables are continually updated while displayed). The character attributes that are available depend on the display type of the operating terminal.

One-Time Output Variables or Cyclic Output Variables:

Pure output variables are transmitted by the controller only once and are displayed in the mask. This method offloads the communication process and is suitable for all variables such as setpoint values, constant values and parameters that change either rarely or never. All output variables can be displayed on a scaled and formatted basis.
Cyclic output variables are used to display actual values, in other words values which vary continually while a mask is displayed. The cycle time (and thus the intervals at which the displayed actual values are refreshed) corresponds to the polling time predefined by the user.
Data transfer between the operating terminal and the PLC can be offloaded by using variables of the same data type only (data word, flag word, input word, flag word) and by using continuous addresses within one mask.
Cyclic output variables can also be scaled and formatted. Note that the display of floating point numbers requires more computing time when the values are scaled. These will therefore not be displayed in "real time". For this reason, the cycle times selected here should be > 500 ms.
Also note that the larger the amount of cyclic data for transfer, the longer the response time to new values from the PLC. As a result, longer polling times are required.

Formatted Output:

Formatting a numerical variable allows the display format of the variable to be adapted to the output range of values. Formatting includes the number of digits to be displayed, the number of post-decimal places (fractional digits) and whether signs are to be used or not. The number of post-decimal places gives the operator the impression that a division has been performed. A division is not actually carried out, however. The variable in the controller must be available in the appropriate (higher) resolution.
Example:
Actual value of a length in the controller:

Word variable	Range of values 0 to 65535
	Resolution 1/100 mm

Display of actual values on the operating terminal:

Display range	0.00 to 655.35 mm

The output is in millimeters, so a conversion in the controller is not necessary. The following format has been selected in the variable definition:

PLC address:	Data type Word
TSdos:	Positive integer (without sign)
TSwin:	Decimal number (without sign)
Length:	6 digits
Post-decimal places:	2 digits

Selecting "with sign" would cause the output to change as follows:

Display range -327.68 to 327.67 mm

Because of the sign, the output length has increased to 7 digits. The format specified in the variable definition must be adapted accordingly.

Format	Data type	formatting	scaling
Binary	Bit, Byte, Word, Lword	x	-
Hexadecimal	Byte, Word, Lword	-	x
Decimal	Bit, Byte, Word, Lword	x	x
BCD	Byte, Word, LWord	-	x
Floating point	Lword	x	x
Selection text (coded text)	Bit, Byte, Word	x	-
Text	ext. ASCII font	-	-

The format corresponds to scaling without the need for calculation.

Scaled Output

When a variable is scaled, its range of values can be adapted to the operator guidance. Scaling applies to data input and output. The operands can be entered in the programming software.

Scaling for integers:

Operand	Range of values
Factor	-32768 to +32767 (excluding the value 0!)
Divisor	1 to +32767
Summand	-32768 to +32767

The operands factor and divisor must be >0!

Scaling for floating point numbers:

Operand	Range of values
Factor	+/-999999999,99999999 (excluding the value 0!)
Divisor	+/-999999999,99999999 (excluding the value 0!)
Summand	+/-999999999,99999999

The operands are stored in IEEE-format
The variable also retains its entire value range for formatting.

Formula for Scaling the Output:

Scaling is always performed on the variable in the operating terminal. Through scaling, the measured values collected in the controller are adapted to the operator guidance. The output representation is determined by *formula 1* only. Unlike the factor and divisor, the summand (representing an offset) can be set to the value 0.
Before a variable or constant is output on the display, it is converted as follows:

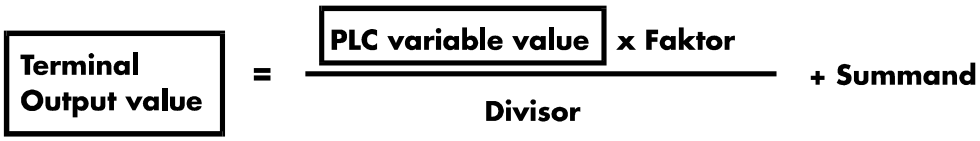


Fig. 15: Formula 1

Output Representation Types:

Certain types of variables can be recognized more easily if they are displayed in their own specific format. A type-specific representation makes interpretation of the variable content easier. For this reason, a wide range of representation formats is provided.
Examples:

- Input statuses of an input module	Binary representation
- Filling level of a container	Bar (bar chart)
- Temperature	Curve (line graph)
- Valve states	Graphics showing valves

Example of a coded text used for outputting an end position condition:

Binary	Hex	Decimal	Selection Text (Coded Text)
0	00	0	Not in end position
1	20	32	In end position

Representation With Leading Zeros:

The option of displaying leading zeros can be used in conjunction with every integer. Leading zeros are required in particular for displaying dates and times and hexadecimal or binary formats. The "Display Leading Zeros" option can be selected in the variable definition.

	Representation without Leading Zeros	Representation with Leading Zeros
Binary Number	10 0101	0010 0101
Time	8: 2:33	08:02:33
Date	5. 3.1997	05.03.1997

3.6.1.1 "Decimal Number" Representation

The decimal representation of numbers is the most frequently used representation method. TSwin differentiates between the following decimal number types of representation: Standard, Timer, Counter and BCD-Format.
Decimal representation includes integers and floating point numbers.

3.6.1.1.1 "Standard" Variable Type

The positional significance of the displayed positions increases from right to left. Leading zeros or the decimal point can be displayed as an option. Decimal representation is suitable for the data types Bit, Byte, Word and Lword. The maximum length depends on the data type. There are no blanks between the characters. In the controller, this variable is in the binary format or special timer or counter formats.

10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	Positional significance
0	1	2	3	4	5	Display ($123,45_D$)

3.6.1.1.2 "Timer" Variable Type

The variable type Timer has a special function only when used in combination with Simatic S5 controllers.
The kind of formatting of the variable type Timer depends on the memory area of the PLC where the value was read. If the value is read directly from a timer word, the 10 bit binary time value contained in it and the 2-bit time base are converted to the time value to the base 10 ms. If read from a different memory area, i.e. data word (DW), flag word (MW), input word (EW) or output word (AW), it is then assumed that the value is BCD-coded (3 digits BCD-code and 2 bit time base). This value will also be converted to a time value to the base 10 ms.
The resulting time value to the base 10 ms can now be formatted in the same way as a fixed point number, i.e. use of post-decimal places is possible and scaling can be applied.

Example:
A setpoint value is entered on the operating terminal: Address MW100
The current actual value is displayed on the operating terminal: Address MW200

Representation Input Variable: Decimal Number / Timer / 7 Digits / 2 Post-Decimal Places /
Factor 1 /Divisor 1 / Summand 0
Representation Output Variable: Decimal Number/ Timer / 6 Digits / 1 Post-Decimal Place/
Factor 1 / Divisor 10 / Summand 0

The command sequence

```
L  MW 100
SI T  1
```

has caused the BCD-coded time value to be loaded into flag word 100.

The command sequence

```
LC T  1        (Load current time value as a BCD-number)
T  MW 200
```

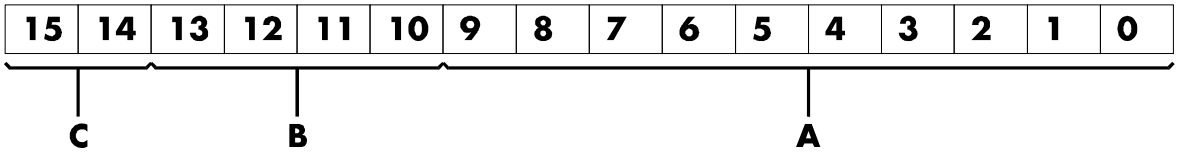
now causes the current time value to be loaded into flag word 200. Before being output, the operating terminal reads this value as a BCD-coded time value and interprets it.
For this process, the time value is converted to the time base 10 ms first and is then scaled. The example lists the output values with one post-decimal place (fractional digit). This produces the following to be displayed on the operating terminal:

Input Value	Output Value	Resolution	KT Values (S5)
0000,01 to 0000,09	0000,0 to 0000,0	0,01 s	001.0 to 009.0
0000,10 to 0000,99	0000,1 to 0000,9	0,01 s	010.0 to 099.0
0001,00 to 0009,99	0001,0 to 0009,9	0,01 s	100.0 to 999.0
0010,00 to 0099,90	0010,0 to 0099,9	0,1 s	100.1 to 999.1
0100,00 to 0999,00	0100,0 to 0999,0	1 s	100.2 to 999.2
1000,00 to 9990,00	1000,0 to 9990,0	10 s	100.3 to 999.3

The resolution and as a result, the precision of the input and displayed values can be influenced by modifying the post-decimal places and scaling.

3.6.1.1.3 "Counter" Variable Type

The variable type Counter can only be used in conjunction with controllers that support this variable type. This will be explained in more detail using the variable type Counter in combination with the Siemens S5 as an example.



A *Dual-coded count value, 0 to 999*
B *Edge trigger flag for setting, releasing, counting up and counting down*
C *Auxiliary flag for queries*

Fig. 16: Structure of the counter word with the Siemens SIMATIC S5-115U

The kind of formatting of the variable type Counter depends on the memory area of the PLC where the value was read. If read directly from a counter word, the 10 bit binary count value contained in it will be displayed directly, so conversion is not necessary.
If read from another memory area, i.e. data word (DW), flag word (MW), input word (EW) or output word (AW), it is assumed that the value is BCD-coded (3 digits BCD-code). The value will then be converted to a binary count value first.
The resulting binary count value can then be formatted in the same way as an integer, i.e. scaling is possible.

3.6.1.1.4 "BCD-Number" Variable Type

The BCD format is partially used to represent numbers in the PLC. This output is required in special cases. The positional significance of the displayed digits increases from right to left. The numbers 0 to 9 are used for the representation; leading zeros can be included optionally. The BCD mode of representation is suitable for the data types Byte, Word and Lword. The length is limited to a maximum of 8 digits. There are no blanks between the characters. The variable value is stored in the controller in BCD format. The range of numbers for a byte is 00 to 99.

10^4	10^3	10^2	10^1	10^0	Positional significance
0	1	2	3	4	Display (1234_D)

3.6.1.2 "Alphanumerical" Representation

In the alphanumerical mode of representation, ASCII strings are read from the controller in byte format and are represented on the display. The number of characters which can be displayed depends on the capabilities of the type of terminal involved. One display line is the maximum permissible length for a variable; longer texts are truncated. The address in the variable list indicates the beginning of the character string. A definition of the variable size is not included and is not necessary. The alphanumerical representation provides another means of editing mask texts during runtime.

3.6.1.3 "Selection Text" (Coded Text) Representation

In the Selection Text (coded text - TSdos) mode of representation, a text string is assigned to a numerical value.
This allows two different tasks to be performed:
1. visualization or selection of statuses by means of text strings
2. change of mask (TSwin).
The selection text in TSwin is used the same way as the selection field in TSdos.
In TSdos, the display of coded texts is limited to one line.

An example of using a one-line selection text is the option of selecting the parity for a serial interface.

In this case, the text list will contain three entries:

Value	Text
0	No Parity
1	Odd
2	Even

A variable (**ComParityA** / **ComParityB**) is created in the mask.
Only these three options are selectable on the operating terminal by means of the Standard or Mix-Mode Editor; invalid inputs are therefore prevented.

Multiple-line selection texts are primarily used for menu control (replacing the node mask). Unlike one-line selection texts, multiple-line selection texts display the text list either in full or in part. If the value for the selection text field height is less than the number of text elements to be displayed, the cursor keys can be used to scroll through the text elements. After the final text list entry is reached, the selection proceeds with the first entry.
To indicate that a text is selected, the entire selection text line is represented in the inverse format.
With the representation type Selection Text, text lists are also used to assign mask names to the mask numbers:

Value	Text
10	Machine parameters
20	Serial message mask
30	Status messages
40	Message configuration
50	Interface parameters

A variable (**NewMask**) of the type Selection Text is created in the Main Mask (mask 4) with the maximum height of 5 lines and a length of 25 characters. The values in the text list and the mask numbers must match.
One of these entries can then be selected from the selection field displayed on the operating terminal: the desired mask is then displayed.

3.6.1.4 "Selection Image" (Coded Image) Representation

In the Selection Image (coded image - TSdos) mode of representation, an image (pixel graphic) is assigned to a numerical value. This image is assigned in an image list. The image list is linked to the variable whose values are to be represented as images. In this way, language-independent visualization of operating states, inputs and outputs, etc. is possible.
Numbers may be freely assigned to the images. They need not necessarily be contiguous or sorted in a consecutive order. In addition, a default image exists for every image list which is displayed, whenever the variable assumes a value that does not exist in the image list. The images used are cut to the size that applies to the output format. This representation is limited to bits, bytes or words. Selection image variables can be both displayed and edited using the +/- key (as with the Selection Text Editor). Any modifications that are made are directly transferred to the controller.

The example below illustrates how images are assigned to numerical values:

Value	Image Name
120	Symbol1
34	Symbol2
7	Symbol3
1201	Symbol4

The names for the images represent the images in the image list. The actual list will look something like this:

This image list has been defined with four entries. The graphic selected by the controller will be displayed.

All of the images in an image list should have the same output size to ensure that they overwrite (overlap) each other completely. An I/O mask can contain multiple selection image variables. These selection image variables can be of the types: one-time output variable, cyclical output variable, or input variable. When the cyclical output of selection images ("animation") is chosen, it must be remembered that the rate of change will be slow due to the limitations of the hardware performance. The more images that are output, the smaller the output performance. The selection image variable should therefore mainly be used for switching states or nearly static processes.

3.6.1.5 "Floating Point Number" Representation

The rules that apply to the Floating Point Number mode of representation are basically the same as those applying to the representation type Decimal Number - variable type Standard. The only difference is that with this representation type, the factor that is used to achieve scaling can be a floating point number; a divisor is therefore not required. With floating point numbers, the reciprocal value can also be generated, before the value is displayed.
Not every controller type supports the floating point format. There are different floating point formats available for processing in the controller (for example, the IEEE format).

3.6.1.6 "Hexadecimal Number" Representation

The hexadecimal representation of numbers is frequently used to display addresses when the PLC is programmed. This format is aimed for use by more experienced operators! The positional significance of the displayed digits increases from right to left. Numbers are expressed by the characters 0 to 9 and A to F. Only capital letters and leading zeros are used for representation. The hexadecimal representation is suitable for the data types Byte, Word and Lword. The length is limited to a maximum of 8 digits. There are no blanks between the characters.

16^4	16^3	16^2	16^1	16^0	Positional significance
0	E	4	5	A	Display ($0E45A_H$)

3.6.1.7 "Binary Number" Representation

The binary format permits the representation of single bits, bytes, words or Lwords. The value for the "length" entered in the variable definition corresponds to the number of bits to be represented. Counting always begins with bit 0. The number of blanks indicates how many gaps (spaces) are inserted between each bit. Output is always in the horizontal position. The direction of the positional significance can be set in the variable definition.

Bit 3	Bit 2	Bit 1	Bit 0	
0	1	0	0	Direction = 76543210 (TSwin)
Bit 0	Bit 1	Bit 2	Bit 3	
0	0	1	0	Direction = 01234567 (TSwin)

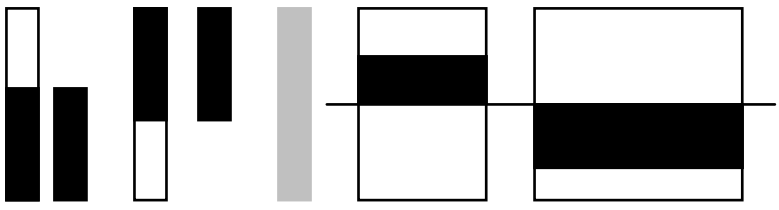
```
0100          Blanks = 0
0 1 0 0       Blanks = 1
0  1  0  0    Blanks = 2
```

3.6.1.8 Bar Representation

Variables in I/O masks can be represented as bars. They can be programmed to be either horizontal or vertical. The bars can start at a reference point (e.g. in the center of the display) and extend in both the positive and the negative direction. The range of values of the bars can be defined by specifying the upper and lower limit (corner values). The bars are represented on the terminal with the aid of four different graphical objects, the simplest of which is a fill pattern. Separate fill patterns can be specified for:

- the empty part of the bar
- the filled-in bar
- the "lower limit exceeded" bar
- the "upper limit exceeded" bar

The programming software includes seven default fill patterns for representing bars on the operating terminal. The other, above-mentioned graphical objects can also be used to design your own fill patterns and shapes.



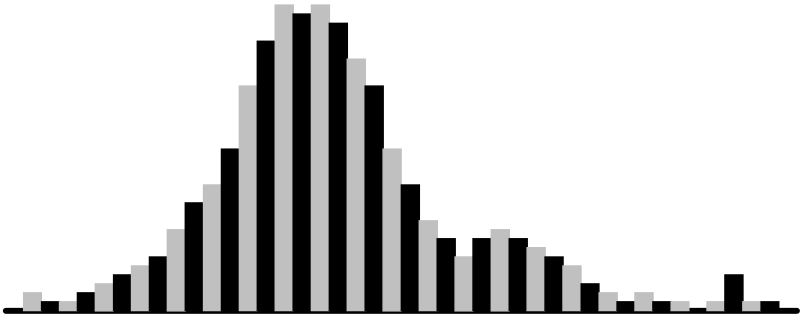


Fig.17: Different forms of bar representation

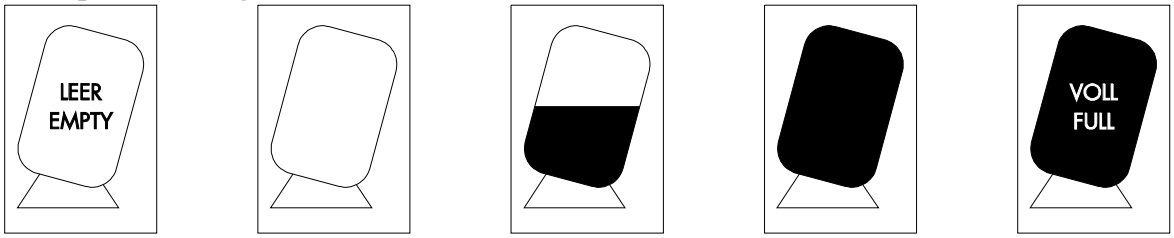
Bar charts can be output once only, cyclically or after defined events (event-controlled). They are used purely to indicate actual values.

The range of values for outputs extends from -32768 to +32767.

Each bar varies between a lower limit and an upper limit. Outside these limits, it changes to the predefined fill pattern for Upper or Lower Limit Exceeded.

Possible applications:
- Vertical and horizontal trend displays
- Visualized monitoring of limit values
- Histograms
- Filling-level indications

Example of a filling-level indication:



Bitmap: Lower Limit Exceeded | Bitmap: Lower Limit Exceeded | Intermediate Value | Bitmap: Upper Limit | Bitmap: Upper Limit

Fig. 18: Example of a filling-level indication

The bar mode of representation always means a slower output rate. Each bar therefore always corresponds to an integer multiple of a character. The smallest possible bar is equivalent in size to a single character, while the largest bar covers the entire display. The bars „grow“ one pixel at a time.

If your mask contains several bars, you should address the variables such that they can be transferred contiguously.

3.6.1.9 Curve Representation (Trendline)

The curve mode of representation permits value tables to be output as dotted lines. A „curve“ variable must be defined in a mask in order to represent a curve. The size of the curve is determined by its length and height. If a coordinate system is required, it can be displayed with the aid of background images.
The address of the curve variable represents the start of a value table in the PLC. Each value in the table describes one pixel on the curve. The graphical representation of the value table resembles that for cyclic output variables.

Examples of a variable time history:
- Output of one-time processes
- Memory function of a point recorder
- Filling-level curves

A curve is limited by the following parameters:
- Maximum height: Height of the display
- Maximum width: 54 pixels per curve variable

If you need a width of more than 54 pixels, you can output several curves directly adjacent to one another.

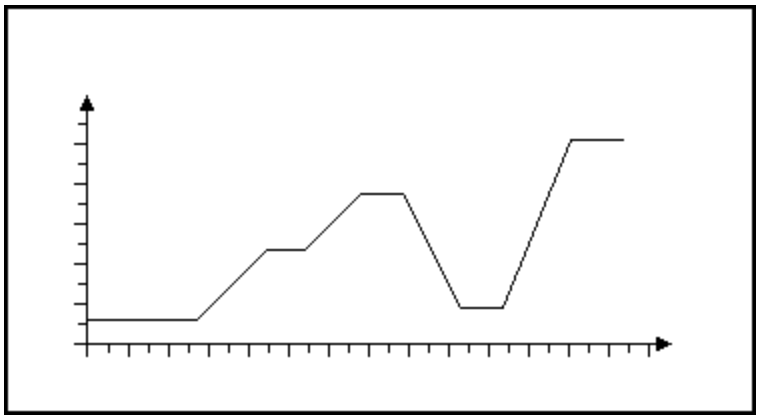


Fig. 19: Example of a curve representation

The height information, in other words the variable values, is read from the controller in a contiguous field by means of one read job. The height information in the field element with the starting address *(address + 0)* is displayed on the far left. All subsequent height information (*address + n*) is shifted one pixel position to the right. The curve height information is refreshed cyclically.
The output is thereby deleted and written again one pixel at a time.
Only two parameters are required to represent a curve:
- Curve width as a multiple of the character width (normal font)
- Curve height as a multiple of the character height (normal font)

3.6.1.10 Selection Field (TSdos) Representation

The Selection Field mode of representation is comparable to coded text in that a text string is assigned to a numerical value from the controller. This facilitates visualization of operating states, inputs, outputs, etc. Unlike the coded text, the selection field displays the text list in full or in parts. The chosen text is selected once the entire selection text line is displayed in the inverse format.
The example taken from the sample projects illustrates how the names of machine operators are assigned to numerical values. Numbers can be freely assigned to the texts. They need not necessarily be contiguous or sorted in a consecutive order. Each line is stored in a text list of the programming system and can be used multiply. This representation is limited to bits, bytes and words. The interpretation of Lword is possible, but requires considerable memory space.

Value	Text
9	Alfred
1	Bernd
7	Detlef
4	Erwin
2	<< blanks were entered here!

The controller value contains the value = 7. Result:

Four entries have been defined for the selection field. If the controller addresses numbers that are not defined in the text list, question marks (that fill the entire length of the field) appear on the display.

Example: The variable contains the value = 5

whereas blanks are displayed for the value 2.

3.6.2 Input Variables

Input variables, when displayed for the first time (when the mask is activated), are treated in the same way as one-time output variables, i. e. the same representation options are available. This includes the scaling effect.

Scaling is performed on the value in the PLC.

Input variables can be modified in the terminal by means of Editors. Their functionality is determined by the type of Editor that is selected.
The Editors that are supported for variable input are the same as those supported for variable output. Editing of the variables is subject to certain conditions which must take into consideration by the user when creating the project, such as the password protection and the external data release, for example.

The following must be observed when entering timers and counters:

Timer
When writing to a timer variable, any scaling that may have been performed is first reversed to give the time value to the base 10 ms again. The BCD-coded value is then calculated so that the lowest possible time base is used.
Avoid writing to a timer word in the PLC, as this has uncontrolled results on the control bits.

Counter
When writing to a counter variable, any scaling that may have been performed is first reversed. The BCD-coded value is then computed and transferred to the PLC.
Avoid writing to a counter word in the PLC, as this has uncontrolled results on the control bits.

Formula for Scaling Input Values

Input values transferred by the PLC are processed according to *formula 1* before being displayed. After the values have been edited, they are processed in accordance with the inverse function (*formula 2*) before being transferred back to the PLC. The inverse function will automatically be generated in the terminal. Operands must be defined by the user on the basis of the values in the PLC including the input value.

$$\text{Controller Value} = \frac{\boxed{\text{Input Value of the Unit}} - \text{Summand}}{\text{Factor}} \times \text{Divisor}$$

Fig. 20: Formula 2

During conversion, the last digit is automatically rounded. This must be kept in mind when defining the upper limit.

$$\left(\boxed{\text{Input Value of the Unit}} \times \text{Factor}\right) < \left(\text{Upper Limit} - \text{Divisor} \;/\; 2\right)$$

Fig. 21: Formula 3

Plausibility Check

A plausibility check is performed on all input variables. As a part of this check, the entered value is validated against the upper and lower limits specified in the variable definition. System messages are generated if the entered value is outside these limits. In this case, the invalid value will not be transferred to the controller and the previous (valid) value will be retained.

If the system message texts "Value too large" and Value too small" are deleted, the following applies:
- if the value drops below the limit, the value corresponding to the lower limit is used
- if the value exceeds the upper limit, the value corresponding to the upper limit is used.

A system message is not output during this process.

Help Text for Input Variables

A screen-sized help text can be created for each input variable. This help text can be displayed in the Editor by pressing the Help key. Important information to be inserted in a help text may include the upper and lower limits, the impact of variables on the process to be carried out, or any interactions with other variables.

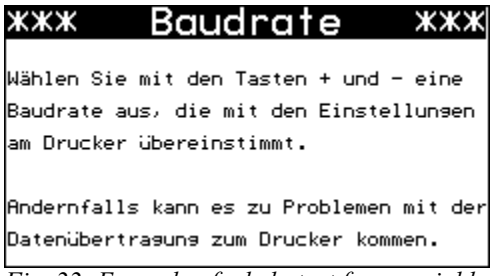


Fig. 22: Example of a help text for a variable

If a variable-specific help text is not defined, the default help text will be displayed instead. This is also a screen-sized text which can be edited in the programming software. If a default help text is not defined, a blank mask will be displayed.

3.6.3 System Variables

System variables can be used to control terminal-internal functions.
System variables can be referenced to function keys and soft keys. They can also be used in masks as normal variables for input and output processes.

The following applies if a system variable is linked to a function key or soft key:
- **do not** link a change of mask function and a system variable to the **same key**
- it is not necessary that the same function key or soft key is used to set (1) and reset (0) a system variable
- jogging mode is obtained by using the same function key or soft key to set (1) and reset (0) a system variable.

Do not include the names of system variables in the variable list. If the name of a system variable is referenced to a PLC address, it will loose its terminal-internal function.
The following subchapters list the system variables, organized into separate groups of functionality, and describes their functions.
In TSdos, all system variables are identified by the syllable **Sys** that precedes the variables. In TSwin, this syllable is not used. Within the following description, the syllable **Sys** is omitted.

3.6.3.1 Basic Functions

IntEraseEprom
Allows the user description to be erased from the Flash memory and prepares the terminal for a new download via interface X3. There will be no communication with any connected PLC during this process. Writing of a "1" to this variable erases the user description unconditionally. Subsequently, the variable is automatically reset to "0".

Data type:	Numeric		
Editor:	Selection text (coded text), positive decimal number		
Output:	---		
Possible values:	(0)	Inactive	Initial state
	(1)	Active	Erases the user description

MainVersion
Allows the user to display the current firmware version of the operating terminal.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric, 8 characters
Possible values:	Format specified by manufacturer

ComVersion
Is used to display the type and version number of the current communication protocol.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric, 8 characters
Possible values:	Format specified by manufacturer

UserVersion
Indicates the version number of the project description to the operator. The user enters this number into the programming system during programming time.

Data type:	Numeric
Editor:	---
Output:	Numeric, 3 digits
Possible values:	---

Boot
This variable can be used to initiate the terminal to boot (system restart). The variable is suitable for integrating the setup function into the regular operator guidance, making it possible to modify terminal parameters which require a subsequent reinitialization (system restart) in any I/O mask. The terminal is booted if a "1" is written to this variable. Subsequently, the variable is automatically reset to "0".

Data type:	Numeric		
Editor:	Selection text (coded text), positive decimal number		
Output:	---		
Possible values:	(0)	Inactive	Initial state
	(1)	Active	System restart

LCDContrast
Is used to adjust the contrast for terminals with liquid crystal displays.

Data type:	Numeric
Editor:	Integer
Output:	---
Possible values:	-127 to +127 (to be limited further based on terminal type)

LCDBackground
Is used to adjust the background of the display. This variable is only relevant for operating terminals fitted with a corresponding display. For further information see the relevant technical manual.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Normal representation
	(1) Inverse background

LCDBackLight
Is used to adjust the background lighting of the display. This variable is only relevant for operating terminals fitted with a corresponding display. For further information see the relevant technical manual (available from firmware 6.40).

Data type:	Numeric
Editor:	Any
Output:	---
Possible values:	(0) Background lighting OFF
	(1 to x) Background lighting ON (dimmed)

TurnOnTemp
Is used to ensure that the liquid crystal display of the operating terminal is not turned on until a certain ambient temperature is reached. This variable is only relevant for operating terminals provided with a corresponding functionality in conjunction with the corresponding display. For further information see the relevant technical manual (available from firmware 6.40).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) LCD-display OFF
	(1 to x) value of the temperature at which the LCD-display is turned ON

OsLanguage
With multilingual user descriptions, this variable is used for online language selection.

Data type:	Numeric
Editor:	Selection text (coded text)
Output:	---
Possible values:	(0) First language
	(n) nth language

3.6.3.2 Communication Area X2

ComDataLenA
Defines the number of data bits for the communication via interface X2 (X2.1).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) 5bits/Char
	(1) 6bits/Char
	(2) 7bits/Char
	(3) 8bits /Char

ComParityA
Determines the creation of and check method of the parity for the communication via interface X2 (X2.1).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) No Parity
	(1) Odd Parity
	(2) Even Parity.

ComStopBitsA
Defines the number of stop bits for the communication via interface X2 (X2.1).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) 1 bit
	(1) 1.5 Bits
	(2) 2 bits

ComBaudrateA
Defines the baud rate for the communication via interface X2 (X2.1).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) 300 Bd
	(1) 600 Bd
	(2) 1200 Bd
	(3) 2400 Bd
	(4) 4800 Bd
	(5) 9600 Bd
	(6) 19200 Bd
	(7) 38400 Bd
	(8) 57600 Bd (BT35 / BT35C / TP35 only)

ComHandshakeA
Specifies the type of handshake for the communication via interface X2 (X2.1).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) No handshake
	(1) RTS/CTS hardware handshake
	(2) XON/XOFF software handshake

ComDefaultA
This variable can be used to program new parameters for the interface (X2).

Data type:	Numeric		
Editor:	Selection text (coded text), positive decimal number		
Output:	---		
Possible values:	(0)	Inactive	Initial state
	(1)	Active	The data entered into the associated system variables are accepted as the new parameters.
	(2)	Active	The data stored in the Flash memory by the programming software are accepted as the new parameters.

ComTimeout
Specifies the monitoring period for the interface X2 (X2.1). The value (0) deactivates the timeout monitoring function.

Data type:	Numeric	
Editor:	Positive decimal number	
Output:	---	
Possible values:	(0) Inactive	Initial state
	(1 to 65535)	Timeout monitoring active (time in ms)

ComRetryTimeout
Specifies the period of time, in milliseconds, that the terminal allows to elapse after an attempt to establish a connection via the communication interface X2 failed, before making another attempt. This allows the time period required for the PLC-specific powerup phase to be bridged, thereby avoiding the generation of an error message.

Data type:	Numeric
Editor:	Positive decimal number
Output:	---
Possible values:	0 to 65535 ms

ComSlaveNr
Contains the slave number used for the device on the network. This address can be used to address the operating terminal on the bus.

Data type:	Numeric
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 255

3.6.3.3 Error Statistics Interface X2

ComParityCount
Error counter for monitoring of parity errors on the interface X2 to the PLC. Is erased on every download.

Data type:	Numeric
Editor:	(Possible)
Output:	Positive decimal number
Possible values:	0to 65535

ComOverrunCount
Error counter for monitoring of overrun errors on the interface X2 to the PLC. Is erased on every download.

Data type:	Numeric
Editor:	(Possible)
Output:	Positive decimal number
Possible values:	0 to 65535

ComFrameCount
Error counter for monitoring of framing errors on the interface X2 to the PLC. Is erased on every download.

Data type:	Numeric
Editor:	(Possible)
Output:	Positive decimal number
Possible values:	0 to 65535

ComStatisticsTab
The system variable points to the beginning of the protocol statistics table with 6 elements of 4 bytes each.

Data type:	Positive decimal number
Editor:	---
Output:	Table
Possible values:	---

ComErrorTab
The system variable points to the beginning of a table with 16 elements which contain the most recent 16 communication errors.

Data type:	Positive decimal number
Editor:	---
Output:	Table
Possible values:	---

ComSubcodeTab
The system variable points to the beginning of a table with 16 elements which contain the subcodes of the communication errors.

Data type:	Positive decimal number
Editor:	---
Output:	Table
Possible values:	---

3.6.3.4 Communication Area X3

ComDataLenB
Defines the number of data bits on the interface (X3).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values	(0) 5 bits/char
	(1) 6 bits/char
	(2) 7 bits/char
	(3) 8 bits/char

ComParityB
Determines the creation of and check method for the parity bit for the interface (X3).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) No parity
	(1) Odd parity
	(2) Even parity

ComStopBitsB
Defines the number of stop bits on the interface (X3).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) 1 bit
	(1) 1.5 bits
	(2) 2 bits

ComBaudrateB
Defines the baud rate on the interface (X3).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) 300 Bd
	(1) 600 Bd
	(2) 1200 Bd
	(3) 2400 Bd
	(4) 4800 Bd
	(5) 9600 Bd
	(6) 19200 Bd
	(7) 38400 Bd
	(8) 57600 Bd

ComHandshakeB
Specifies the type of handshake on the interface (X3).

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---

Possible values:	(0) Inactive	Initial state
	(1) Active	The data entered into the associated system variables are accepted as the new parameters.
	(1) Active	The data stored in the Flash memory by the programming software are accepted as the new parameters.

3.6.3.5 Real-Time Clock

RTCSec
This variable is used to display and set the seconds of the clock.

Data type:	Positive decimal number
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 59

RTCMin
This variable is used to display and set the minutes of the clock.

Data type:	Positive decimal number
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 59

RTCHour
This variable is used to display and set the hours of the clock.

Data type:	Positive decimal number
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 23

RTCDay
This variable is used to display and set the day of the month.

Data type:	Positive decimal number
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 31 (varies from month to month, clock corrects incorrect entries on the next change of date)

RTCMonth
This variable is used to display and set the month of the year.

Data type:	Positive decimal number
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 12

RTCYear
This variable is used to display and set the year.

Data type:	Positive decimal number
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 99 (only the year and the decade are influenced)

RTCDayofWeek
This variable is used to display and determine the day of the week. The variable assumes the values 0 to 6. This is a modulo counter. A coded text should be used for the display. The assignment and starting point can be specified as needed.

Data type:	Positive decimal number
Editor:	Selection text (coded text)
Output:	Selection text (coded text)
Possible values:	0 to 6

RTCDateFmt
Is used to enter the format according to which the date is to appear in displayed messages.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) European DD MM YY (1) US MM DD YY (2) JAPAN JJ MM DD

RTCYear2000
This variable is used to display and set the year in a 4-digit format.

Data type:	Numeric
Editor:	Selection text (coded text), selection image, alphanumeric, decimal
Output:	---
Possible values:	0 to 9999

3.6.3.6 Serial Message System

RepmanSortCrit
Is used to enter the sorting criterion for the serial message output.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) By priority of message number
	(1) By time of arrival (newest first)
	(2) By time of arrival (oldest first)

ClearRepBuf
Is used to erase the entire serial message buffer. Optionally direct control via function keys, soft keys or Editor. If necessary, passwords or special masks should be implemented to control erasure of the message buffer.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Maintain data
	(1) Erase message buffer

RepmanRepPrint
Switches the output of messages via the printer on and off.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Off
	(1) On

RepoutNr
Switches the output of the message number in the message mask on and off.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Off
	(1) On

RepoutDate
Switches the output of the date of the message in the message mask on and off.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Off
	(1) On

RepoutTime
Switches the output of the time of the message in the message mask on and off.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Off
	(1) On

RepoutAnzYear
Determines the number of digits that is to be used to represent the year.

Data type:	Numeric
Editor:	Selection text (coded text), selection image, decimal
Output:	---
Possible values:	(0) year represented by 2 digits
	(1) year represented by 4 digits

RepoutRepText
Is used to display the most current serial message. The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	Any

RepoutRepText21
Is used to display the most current message of the serial message system - from the 21st digit with a variable length (20, 40 or 60). The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	Any

RepoutRepText41
Is used to display the most current message of the serial message system - from the 41st digit with a variable length (40 or 60). The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type: Alphanumeric
Editor: ---
Output: Alphanumeric
Possible values: Any

RepoutRepText61
Is used to display the most current message of the serial message system - from the 61st digit. The last 20 characters of a message are displayed. The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type: Alphanumeric
Editor: ---
Output: Alphanumeric
Possible values: Any

3.6.3.7 Parallel Message System

RepmanSortCritP
This variable indicates to the terminal the sorting criterion according to which the messages are to be displayed. The default value is defined in the user description. This variable can be adapted during editing processes at a later date.

Data type: Numeric
Editor: Selection text (coded text), positive decimal number
Output: ---
Possible values: (0) By priority of message number
(1) By time of arrival (newest first)
(2) By time of arrival (oldest first)

RepoutNrP
This variable indicates to the terminal whether or not the message number is to be displayed along with the message text of messages in the parallel message system. The default value is defined in the user description. This variable can be adapted during editing processes at a later date.

Data type: Numeric
Editor: Selection text (coded text), positive decimal number
Output: ---
Possible values: (0) Off
(1) On

RepoutDateP
This variable indicates to the terminal whether or not the date is to be displayed along with the message text of messages in the parallel message system. The default value is defined in the user description. This variable can be adapted during editing processes at a later date.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Off
	(1) On

RepoutTimeP
This variable indicates to the terminal whether or not the time is to be displayed along with the message text of messages in the parallel message system. The default value is defined in the user description. This variable can be adapted during editing processes at a later date.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Off
	(1) On

RepoutAnzYearP
Determines the number of digits that is to be used to represent the year.

Data type:	Numeric
Editor:	Selection text (coded text), selection image, alphanumeric, binary, decimal, hexadecimal
Output:	---
Possible values:	(0) year represented by 2 digits
	(1) year represented by 4 digits

RepoutRepTextP
The terminal uses this variable to display the most current message of the parallel message system.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	Any

RepoutRepTextP21
Is used to display the most current message of the parallel message system - from the 21st digit with a variable length (20, 40 or 60). The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	Any

RepoutRepTextP41
Is used to display the most current message of the parallel message system - from the 41st digit with a variable length (40 or 60). The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	Any

RepoutRepTextP61
Is used to display the most current message of the parallel message system - from the 61st digit. The last 20 characters of a message are displayed. The message is displayed in the same way as it would appear in the message mask, i.e. in accordance with the selected message parameters.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	Any

3.6.3.8 Printer Control

StopPrint
When the system variable is activated, the print process currently in progress is terminated.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Terminate print process

BlockPrint
Upon activation of this system variable, the block selected in the message mask is printed.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Print block

PrintAllRep
Upon activation of this system variable, the entire serial message memory is printed. The output is as predefined.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Formatted printout (2) Full-length printout

PrintAllState
Upon activation of this system variable, the entire parallel message memory (status messages) is printed.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (2) Print message buffer

BlockPrintLong
Upon activation of this system variable, the entire area selected in a message mask is printed in full length. The settings chosen at programming are ignored.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Print selected area

3.6.3.9 Menu Control / Keys

NewMask
If a mask number is written to this system variable, the system will switch to the corresponding mask.

Data type:	Numeric
Editor:	Selection text (coded text), selection field
Output:	---
Possible values:	1 to 9999

VarTablenR0
Generates a consecutive numbering in tables beginning with the number "0". This system variable is also used to output constant table texts.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text), positive decimal number
Possible values:	0 to n

VarTablenR1
Generates a consecutive numbering in table beginning with the number "1". This system variable is also used to output constant table texts.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text), positive decimal number
Possible values:	1 to n

HardCopy
Is used to upload a hard copy in PCX or ASCII format via interface X3 to the connected PC.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Activate hard copy (always currently displayed screen)

TabLeft
In tables, this system variable can be used to shift to the column on the left.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) To the left by one column

TabRight
In tables, this system variable can be used to shift to the column on the right.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) To the right by one column

TabPgUp
This system variable enables soft keys to be used in tables to shift from page to page (in an upward direction).

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Move up one page

TabPgDn
This system variable enables soft keys to be used in tables to shift from page to page (in a downward direction).

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Move down one page

Shift
When the system variable is set, alphanumerical characters can be entered. When the keys on the numerical keypad are pressed, the associated alphabetical letters are automatically provided. By pressing repeatedly, you can proceed through the associated letters. Only uppercase letters are provided.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state, numbers only
	(1) Shift mode 1 active, uppercase letters

Key	**Letters (Characters)**
Point	: ? ! .
Minus	\ * / -
Plus	< = > +
Zero	( ) ° 0
One	S T U 1
Two	V W X 2
Three	Y Z % 3
Four	J K L 4
Five	M N O 5
Six	P Q R 6
Seven	A B C 7
Eight	D E F 8

ShiftCase

When the system variable is set, alphanumerical characters can be entered. When the keys on the numerical keypad are pressed, the associated alphabetical letters are automatically provided. By pressing repeatedly, you can proceed through the associated letters. Uppercase and lowercase letters are provided.

Data type:	Numeric
Editor:	Soft key, Function key
Output:	---
Possible values:	(0) Initial state, numbers only
	(1) Shift Mode 2 active, lowercase and uppercase letters

Key	**Letters (Characters)**
Point	: ? ! .
Minus	\ * / -
Plus	< = > +
Zero	( ) ° 0
One	S T U s t u 1
Two	V W X v w x 2
Three	Y Z % y z % 3
Four	J K L j k l 4
Five	M N O m n o 5
Six	P Q R p q r 6
Seven	A B C a b c 7
Eight	D E F d e f 8

KeyCursLeft
This system variable allows soft keys to be used as a *Cursor left* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Cursor left* key.

KeyCursRight
This system variable allows soft keys to be used as a *Cursor right* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Cursor right* key

KeyCursUp
This system variable allows soft keys to be used as a *Cursor up* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Cursor up* key

KeyCursDown
This system variable allows soft keys to be used as a *Cursor down* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Cursor down* key

KeyHome
This system variable allows soft keys to be used as a *Cursor home* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Cursor home* key

KeyHelp
This system variable allows soft keys to be used as a *Help* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Help* key

KeyDot
This system variable allows soft keys to be used as a *Decimal point* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Decimal point* key

KeyClear
This system variable allows soft keys to be used as a *Delete* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Delete* key

Key0
This system variable allows soft keys to be used as the key *0 (zero)*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *0*

Key1
This system variable allows soft keys to be used as the key *1* .

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key*1*

Key2
This system variable allows soft keys to be used as the key *2*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *2*

Key3
This system variable allows soft keys to be used as the key *3*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *3*

Key4
This system variable allows soft keys to be used as the key *4*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *4*

Key5
This system variable allows soft keys to be used as the key *5*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *5*

Key6
This system variable allows soft keys to be used as the key *6*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *6*

Key7
This system variable allows soft keys to be used as the key *7*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1 Treat soft key as the key *7*

Key8
This system variable allows soft keys to be used as the key *8*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1 Treat soft key as the key *8*

Key9
This system variable allows soft keys to be used as the key *9*.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as the key *9*

KeyPlus
This system variable allows soft keys to be used as a *Plus* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Plus* key

KeyMinus
This system variable allows soft keys to be used as a *Minus* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Minus* key

KeyEnter
This system variable allows soft keys to be used as an *Enter* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as an *Enter* key

KeyEdit
This system variable allows soft keys to be used as a *Release* key.

Data type:	Numeric
Editor:	Soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Treat soft key as a *Release* key

3.6.3.10 Password

MskchgPasswd
Is used to enter the password into a mask.

Data type:	Numeric (alphanumeric - only terminals with corresponding keys)
Editor:	Alphanumeric
Output:	---
Possible values:	11 characters

MskchgResPasswd
Is used to erase the password currently entered; resets the access authorization. The access levels are reset on every power-up.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state
	(1) Reset access authorization

ChangePasswd
This system variable can be used to modify the passwords in the terminal.

Data type:	Numeric (alphanumeric - only terminals with corresponding keys)
Editor:	Alphanumeric
Output:	---
Possible values:	11 characters

FlashPasswd
This system variable can be used to reset the passwords to the code values specified in the programming software. Useful in the case of a loss of the passwords. Make sure to make a note of the master password first.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Reset passwords

Passwdlnactive
Is used to deactivate the password protection. The system variable is battery-backed. The most recent setting is retained when the device is turned off.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Password protection active (initial state when first initialization) (1) Password protection inactive (edit and view level = 255)

3.6.3.11 Recipes

SelectDSNr
This variable contains the number of the active data set. To edit the variable, the associated Selection Text Editor must be used.

Data type:	Numeric
Editor:	Selection text (selection field for TSdos)
Output:	Positive decimal number
Possible values:	0 to 250

SelectDSName
This variable contains the name of the active data set. To edit the variable, the associated Selection Text Editor must be used.

Data type:	Alphanumeric
Editor:	Selection text (selection field for TSdos)
Output:	Alphanumeric
Possible values:	15 characters

DestDSNr
When copying data sets, this variable contains the number of the destination data set.

Data type: Numeric
Editor: Positive decimal number
Output: ---
Possible values: 1 to 250

DSCopy
This variable can be used to copy the active data set to the destination specified in **DestDSNr**.

Data type: Numeric
Editor: Selection text (coded text), positive decimal number, soft key, function key
Output: ---
Possible values: (0) Initial state
(1) Copies to destination in **DestDSNr**
(2) Copying and automatic search for a free data set
(3) Copies to destination in **DestDSNr** and overwrites any existing data set.

DSDelete
This variable can be used to delete the active data set. The first data set from the same recipe will become the new active data set.

Data type: Numeric
Editor: Selection text (coded text), positive decimal number, soft key, function key
Output: ---
Possible values: (0) Initial state
(1) Deletes the data set

ActDSName
This variable contains the name of the current data set. It can be read and in the case of RAM-data sets, it can additionally be written.

Data type: Alphanumeric
Editor: Alphanumeric
Output: Alphanumeric
Possible values: 15 characters

SelectRezeptNr
This variable contains the active recipe. The variable can also be modified outside the recipe mask.

Data type:	Numeric
Editor:	Numeric, Selection text (coded text)
Output:	Numeric, Selection text (coded text)
Possible values:	1 to 250

DSDownload
This variable can be used to write the active data set to the controller.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Writes the data set to the controller

DSDnloadBreak
This variable can be used to terminate a data transfer to the controller currently in progress.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number, soft key, function key
Output:	---
Possible values:	(0) Initial state (1) Terminates data set transfer

DSDnloadState
This variable can be used to monitor the data transfer to the controller.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text)
Possible values:	(0) Initial state (1) Request for data transfer issued, but not yet released by the controller (2) Data transfer active

LoadDSName
This variable contains the name of the last data set which has been transferred to the controller. If the data set has already been deleted, question marks are displayed.

Data type:	Alphanumeric
Editor:	---
Output:	Alphanumeric
Possible values:	15 characters

StartSave
This variable can be used to transfer data sets to the PC.

Data type:	Numeric
Editor:	Selection text (coded text)
Output:	---
Possible values:	(0) Initial state
	(1) Transfer one data set to the PC
	(2) Transfer all data sets of a recipe to the PC
	(3) Transfer all data sets in the terminal to the PC

SaveState
During a transmission to the PC, this variable indicates the current status of the transmission process.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text)
Possible values:	(0) Initial state
	(1) One data set is transferred to the PC
	(2) All data sets of a recipe are transferred to the PC
	(3) All data sets in the terminal are transferred to the PC

StartRestore
This variable controls the download from the PC to the Terminal.

Data type:	Numeric
Editor:	Selection text (coded text)
Output:	---
Possible values:	(0) Initial state
	(1) The terminal switches to the ready-to-receive state
	(2) The terminal terminates a transmission currently in progress.

RestoreState
This variable indicates the status of the transmission from the PC to the terminal.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text)
Possible values:	(0) Initial state
	(1) Data transmission in progress

RestoreLineNr
This variable indicates the current line number of the data set file. It is used to verify the progress of the receive process and in the case of an error, for error localization.

Data type:	Numeric
Editor:	---
Output:	Numeric
Possible values:	1 to 255

StartRezPrint
This variable can be used to print data sets.

Data type:	Numeric
Editor:	Selection text (coded text)
Output:	---
Possible values:	(0) Initial state
	(1) Starts data set printout
	(2) Terminates print process

RezPrintState
When printing data sets, this variable indicates the current printing status.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text)
Possible values:	(0) Initial state
	(1) Data set printout in progress

StartUpload
This variable can be used to read, for the active recipe, a data set from the controller and to store it in the terminal.

Data type:	Numeric
Editor:	Selection text (coded text)
Output:	---
Possible values:	(0) Initial state
	(1) Variables are read individually from their specified addresses
	(2) Variables are read as a block from the buffer specified for the recipe
	(3) Variables are read individually from their specified addresses. A free data set is automatically being searched for. If no free data set is available, system message 18 is displayed.
	(4) Variables are read as a block from the buffer specified for the recipe. A free data set is automatically being searched for. If no free data set is available, system message 18 is displayed.

UploadDSNr
This variable specifies the data set number to which the uploaded data set is to be written in the operating terminal.

Data type:	Numeric
Editor:	Numeric
Output:	---
Possible values:	1 to 250

UploadState
When uploading data sets to the operating terminal, this variable indicates the upload status.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text)
Possible values:	(0) Initial state
	(1) Upload of data set in progress

3.6.3.12 Running Time Meter

Counter1
Counter2
Counter3
Counter4
Counter5
Counter6
Counter7
Counter8
Is used in conjunction with the start/stop function of the associated bit as a running time meter. The counter is incremented while the bit is set.

Data type:	Numeric
Editor:	Positive decimal number
Output:	Positive decimal number
Possible values:	0 to 4.294.967.295

3.6.3.13 Loop-Through Operation

Pg2Sps
Activates and deactivates the loop-through operation (toggle function). This function must be provided by the PG protocol.

Data type:	Numeric
Editor:	Selection text (coded text), positive decimal number
Output:	---
Possible values:	(0) Initial state
	(1) First 1 activates, second 1 deactivates the loop-through operation

Pg2SpsState
Indicates the status of the loop-through operation.

Data type:	Numeric
Editor:	---
Output:	Selection text (coded text)
Possible values:	(0) Initial state (OFF)
	(1) Issue request for loop-through operation
	(2) Loop-through operation possible
	(3) Loop-through operation active

3.6.3.14 Loadable Font

ChrsetName
Is used to display the current font name.
The name of the font is limited to a maximum length of 8 characters.

Data type:	Alphanumeric
Editor:	---
Output:	---
Possible values:	(Standard) Display using terminal font
	(Font name) Display using own font

3.6.3.15 Maintenance

User1
User2
User3
User4
User5
Universal variables that permit the user to store information in the terminal. The data are stored in the battery-backed RAM.

Data type:	Any
Editor:	Any
Output:	Any
Possible values:	16 bit

LCDADCInput
Is used to read the current input value from the AD-converter that is used for LCD contrast control. This value is only designed to be used for diagnosis purposes.

Data type:	Numeric
Editor:	Any
Output:	---

LCDDACOutput
Is used to read the current output value from the DA-converter that is used for LCD contrast control. This value is only designed to be used for diagnosis purposes.

Data type:	Numeric
Editor:	Any
Output:	---

Break
The current Editor is interrupted and the values entered are not transferred to the controller. Trigger the Break function with a function key in a soft key.

Data type:	Numeric
Editor:	Function key, soft key, selection text, decimal number, binary number, hexadecimal number
Output:	---
Possible values:	(0) Initial state (1) Editing process is terminated

3.6.3.16 Editors

EditInvers
Specifies if the inverse representation is to be used for the Editor for inputs.

Data type: Numeric
Editor: Any
Output: ---
Possible values: (0) Normal representation
(1) Inverse representation

EditEnter
Specifies the behavior of the Editor when the Data Release key is pressed.

Data type: Numeric
Editor: ---
Output: ---
Possible values: (0) Editor proceeds to the next input field (default setting)
(1) Editor remains at the current position. To proceed, the cursor must be used.

StatePerm
Specifies the status of the status LED in the Data Release key.

Data type: Numeric
Editor: Any
Output: 0, 1, 2
Possible values: (0) Status LED OFF
(1) Status LED ON
(2) Status LED FLASHING

3.6.3.17 Help

StateHelp
Specifies the status of the status LED in the Help key.

Data type: Numeric
Editor: Any
Output: 0, 1, 2
Possible values: (0) Status LED OFF
(1) Status LED ON
(2) Status LED FLASHING

Message
In the case of a system error, the number of the system message is written to this variable..

Data type:	Numeric
Editor:	Any
Output:	---
Possible values:	(0) Initial state
	(1 to 29) System message number

QuitMessage
Acknowledges the system message that is currently displayed by the system variable **Message**.

Data type:	Numeric
Editor:	Any
Output:	---
Possible values:	(0) Initial state
	(1) Acknowledge

3.6.4 Editors

Various Editors are available in I/O masks for the various data types. With the term Editor, we refer to program parts designed to provide a convenient method of editing input variables. This sequence control required for the editing process is completely integrated into the terminal. An additional "software" in the controller is not required for this purpose.
The Editors are influenced by the:
- data release (particularly from the connected controller)
- cyclic value refreshing
- the help system and
- the plausibility check.

The Editors are operated with the aid of editing and control keys. For details on the respective functions consult the corresponding Editor description.
The variables of the controllers can be modified using the Editors. Upon entry, the numerical variables are checked for plausibility. The limits are determined by the user in the variable definition. It is also possible to include a variable-specific help text. This text may, for example, provide a description of the variable function and its valid range of values.

Variables are entered as follows:
The Data Release key will allow the Editor to be activated if the I/O mask contains an editable value. When in the editing mode, the status LED in the Data Release key lights up and the cursor points at the first editable variable. This value can now be edited by using the numerical keyboard and/or the Plus/Minus keys. The value is stored by pressing the Enter key. The value is checked for plausibility during this process. If an error is detected, the status LED in the Help key will begin to flash.

A flashing status LED in the Help key indicates a malfunction to the user. A description of the malfunction will be displayed if the Help key is pressed. In the case of an error, the Enter key can not be used to exit the editing field.
The control keys, on the contrary, can be used to exit an editing field even if the value is invalid. In this case, however, the last value which was entered will be discarded and the old value (original value) will be restored.
In the editing mode, control keys can be used to select editable variables. If the Data Release key is pressed again, the editing mode will be exited and the status LED will go off.
The following Editors are currently available for selection in the variable definition:

TSdos Editors	**TSwin Editors**
Integer	Decimal Number / No Post-Decimal Places
Real	Decimal Number / Post-Decimal Places
Floating Point	Floating Point Number
Selection Item / Coded Text	Selection Text
Coded Image	Selection Image
Selection Item	Selection Text
Curve Diagram	Curve
Beam Diagram	Bar
Alphanumerical	Alphanumeric
Hexadecimal	Hexadecimal Number
Binary	Binary Number
BCD-Number	Decimal Number / Attribute BCD
Timer	Decimal Number / Attribute Timer
Counter	Decimal Number / Attribute Counter
Password	Alphanumeric / Password

The functions of the keys are identical for all numerical Editors.

Key Functions in the Numerical Editor

Key: Cursor up	Moves the cursor up on the display by one editable variable and selects it in the process. After the cursor reaches the editable variable at the top, the variable at the bottom is selected next.
Key: Cursor down	Moves the cursor down on the display by one editable variable and selects it in the process. After the cursor reaches the editable variable at the bottom, the variable at the top is selected next.
Key: Cursor left	Moves the cursor within the editable variable to the left by one position until it reaches the end of the field.
Key: Cursor right	Moves the cursor within the editable variable to the right by one position until it reaches the end of the field.
Key: Plus	1. Variable currently selected: Value is deleted, a new value can be entered 2. Cursor has been moved within a positive value: No change 3. Cursor has been moved within a negative value: Deletes the negative sign
Key: Minus	1. Variable currently selected: Value is deleted, negative sign is entered at the least significant position, a new value can be entered 2. Cursor has been moved within a positive value: Negative sign is placed in front of the value 3. Cursor has been moved within a negative value: No change
Key: Delete	Deletes the position where the cursor is currently located (the sign also).

3.6.4.1 Decimal Number Editor

The Decimal Number Editor is a numerical Editor, optionally for positive decimal numbers only or for positive/negative decimal numbers.
In addition, the following variable types can be selected for decimal numbers:

- Standard Type Integers / fixed point numbers
- Timer Timer values in S5 format
- Counter Counter values in S5 format
- BCD Number Decade switch / incremental Editor

The number of digits (field length) is limited to the highest displayable 4-byte number. Decimal numbers can be output with leading zeros. Fixed point numbers can be represented by specifying a certain number of post-decimal places (fractional digits).

Other selectable options are scaling factors, limits, message transfer mode and display attributes. This Editor can be used to access bit, byte, word and Lword variables. The maximum limits are determined by the memory map.

The variable types Timer and Counter are represented and can be edited on the operating terminal in the same way as the variable type Standard Type.

The variable type BCD-Number has a special feature.
The Editor for BCD-numbers is also referred to as the “Decade Switch”.
With scaled variables, the value in the PLC changes by +1/-1, although the input value which is actually displayed also depends on the defined scaling factor! This means that special care must be taken whenever scaling is required. The Editor is also suitable for making fine adjustments or as a decade switch with decimal carry-over.
Possible Data Types:
- Positive integers (no entry of signs)
- Integers
- Positive fixed point numbers (no entry of signs)
- Fixed point numbers

Key Functions in the BCD-Number Editor

Keys: 1 to 9	1.)	Standard Direct input of numerical values
	2.)	Mix-Mode Direct input of numerical values
	3.)	Incremental No Function
Key: Plus	1.)	Standard No Function
	2.)	Mix-Mode and Incremental Increases the value at the cursor position and influences the more significant digits when the range is exceeded (with a dynamic repeat function)
Key: Minus	1.)	Standard No Function
	2.)	Mix-Mode and Incremental Reduces the value at the cursor position and influences the more significant digits when this value falls below the range (with a dynamic repeat function)
Key: Cursor left		Moves the cursor within the editable variable to the left by one position until it reaches the end of the field.
Key: Cursor right		Moves the cursor within the editable variable to the right by one position until it reaches the end of the field.

3.6.4.2 Floating Point Number Editor

The Floating-Point Number Editor supports Lword variables stored in IEEE format. This variable type is not supported by every controller type. In this case, an alternative would be to resort to fixed point numbers in Lword format. Scaling factors are also suitable for displaying precise fractional numbers.
The functions of the control and editing keys are identical to those of the Decimal Number Editor.

3.6.4.3 Hexadecimal Editor

The Hexadecimal Editor permits inputs of hexadecimal values. The sign keys have a special function here. Since there are no alphanumerical keys, alphanumerical characters can be entered with the sign keys instead.

Key Functions in the Hexadecimal Editor

Keys: 1 to 9	Direct entry of numerical values
Key: Plus	Enters the ASCII-characters 0 to 9, A to F in ascending order at the selected position
Key: Minus	Enters the ASCII-characters F to A, 9 to 0 in descending order at the selected position
Key: Cursor left	Moves the cursor within the editable variable to the left by one position until it reaches the end of the field.
Key: Cursor right	Moves the cursor within the editable variable to the right by one position until it reaches the end of the field.

3.6.4.4 Alphanumerical Editor

Entering alphanumerical characters with the numerical keyboard is possible by means of a special Incremental Editor. This Editor permits text strings that consist of lower and upper-case alphanumerical characters to be edited.

Key Functions in the Alphanumerical Editor

Keys: 1 to 9	Direct entry of the numbers 0 to 9.
Key: Plus	Enters the ASCII-characters 0 to 9, A to Z, a to z in ascending order at the selected position.
Key: Minus	Enters the ASCII-characters z to a, Z to A, 9 to 0 in descending order at the selected position.
Key: Cursor left	Moves the cursor within the editable variable to the left by one position until it reaches the end of the field.
Key: Cursor right	Moves the cursor within the editable variable to the right by one position until it reaches the end of the field.
Delete	Deletes the character at the cursor position. As the character is deleted, the text to the right of the cursor is moved to the left by one character.

Field Type "Password"

In conjunction with the field type Password, the Alphanumerical Editor allows hidden entry of numerical values for the password. The field type Password has the same functions as the standard Alphanumerical Editor except that the characters do not appear when they are typed in. The entered character is represented by a letter "X". Entries are made in insert mode. The function of the field type **Password** is linked to the system variable **MskchgPasswd**. To be able to enter the password such that it also appears when it is entered, an alphanumerical variable without the field type Password must be used and linked to the system variable **MskchgPasswd**.

3.6.4.5 Selection Text Editor (Coded Text)

The Selection Text Editor offers texts contained in a text list for selection. Every text in a text list is assigned to a numerical value. In the case of input variables, the numerical value associated with the corresponding text is written to the PLC variable. For output variables, the text associated with the numerical value is displayed on the operating terminal.

Key Functions in the Selection Text Editor

Key: Plus	Selection in ascending order (after the final value is reached, the value at top of the text list is selected next.)
Key: Minus	Selection in reversed order (after the first value is reached, the value at the bottom of the text list is selected next.)

3.6.4.6 Selection Image Editor (Coded Image)

The Selection Image Editor offers images contained in an image list for selection. A numerical value is assigned to every image in an image list. In the case of input variables, the numerical value associated with the corresponding image is written to the PLC variable. For output variables, the image associated with the numerical value is displayed on the operating terminal.

Key Functions in the Editor for Coded Images

Key: Plus	Selection in ascending order
Key: Minus	Selection in reversed order

3.6.4.7 Table Editor

Definition of Terms:

Number of rows:	Number of rows in the table that appear on the screen.
Number of elements:	Number of variables that are stored in the PLC for every column.
Table offset:	Offset of the variable represented in the first row in relation to the top of the table. When a mask is refreshed, the table offset is always set to 0. After paging down once, it is for example equal to the number of rows. The table offset is always the same for every column.
Single variable:	Every variable in a mask located outside of the table.
Column variable:	Every variable placed in a table creates one column.
Mask variable:	Column variable or single variable.

Row variable: Every row in the column of a table contains one row variable.
Last row variable: Row variable in the bottom row of the table with the maximum table offset.
First row variable: Row variable in the top row of the table with a table offset of 0.

Functional Description:
- Every row variable is read from the PLC again just before being released for editing.
- Input and cyclic output variables in the table are also continuously updated. An exception to this is the row variable that is currently being edited. One-time output variables are updated whenever the mask is refreshed (for example, when the Help key is released) or whenever the table offset changes.
- Cursor keys and soft keys linked to system variables can be used to move from one variable to another. The order of movement between the mask variables corresponds to the order in which the variables were created in the mask (single variables) and in the table field. The following table illustrates the system variables (and their functions) that are relevant for navigating in table fields.

System Variable	Function in the Table Field	Key/Position
TabPgDn	Page Down	Function Key / Soft Key
TabPgUp	Page Up	Function Key / Soft Key
TabLeft	Shifts to the next column on the left When in the leftmost column, shifts to the previous mask variable	Function Key / Soft Key
TabRight	Shifts to the next column on the right When in the rightmost column, shifts to the next mask variable	Function Key / Soft Key
VarTablenR0	Displays row number beginning with 0	On the left side of the table field
VarTablenR1	Displays row number beginning with 1	On the left side of the table field

- Once the data release has been set (the status LED in the Data Release key lights up), the variables in a table field can be viewed and edited by means of the *Cursor up*, *Cursor down*, *Enter*, *Page up* and *Page down* keys. If the data release has not been set (the status LED in the Data Release key is off), the variables can only be viewed (using the *Page down* and *Page up* keys) and not edited.
- It must be possible to read all of the variables in one column of a table with one read request. When using 4-byte variables, this can result in a restriction of the number of rows.

Key Functions in the Table Editor

Key: Cursor up — Shifts to the previous mask variable if the cursor is located at a single variable or at the first row variable.
Shifts to the previous row variable if the cursor is located at a variable other than the first row variable. To be able to scroll row-by-row, the cursor must already be positioned at the row variable at the top.

	When shifting to a new column variable, the cursor is placed at the bottom row variable (i.e. the last row variable). When shifting to a single variable, the row number remains unchanged.
Key: Cursor down	Shifts to the next mask variable if the cursor is located at a single variable or at the last row variable. Shifts to the next row variable if the cursor is located at a variable other than the last row variable. To be able to scroll row-by-row, the cursor must already be positioned at the bottom (not the last) row variable. When shifting to a new column variable, the row number is set to 0 and the cursor is placed at the top row variable (i.e. the first row variable). When shifting to a single variable, the row number remains unchanged.
Soft key: TabLeft	Shifts to the next column on the left in the same row. Shifts to the previous mask variable if the cursor is located in the column at the left-most position.
Soft key: TabRight	Shifts to the next column to the right in the same row. Shifts to the next mask variable if the cursor is located in the column at the right-most position.
Key: Page up	Reduces the table offset by the number of rows scrolled up. The row number is reduced by the corresponding number of rows (otherwise the remaining number of rows). Once the row number has also reached the minimum value, the key is no longer effective. Shifting to another mask variable in particular is not possible.
Key: Page down	Increases the table offset by the number of rows scrolled down. The row number is increased by the corresponding number of rows (otherwise the remaining number of rows). Once the row number has also reached the maximum value, the key is no longer effective. Shifting to another mask variable in particular is not possible.
Key: Enter	Function is the same as that of the Cursor down key, but the variable is additionally written to the PLC.

3.6.5 External Data Release

The PLC can prevent the input of variable values by the operator using the external data release. To be able to make use of this data release function, the following variables must have been created:
- a variable for the Read coordination byte
- a variable for the poll area.

When the Data Release key is pressed in an I/O mask, the terminal will enable the editing mode. If the controller has set the external data release, the status LED in the Data Release key lights up. The external data release is controlled by bit 0 in the first byte of the cyclic poll area of the controller (see chapter Poll Area). Data editing is inhibited whenever the PLC sets this bit to 0. In this case, the status LED in the Data Release key will begin to flash.

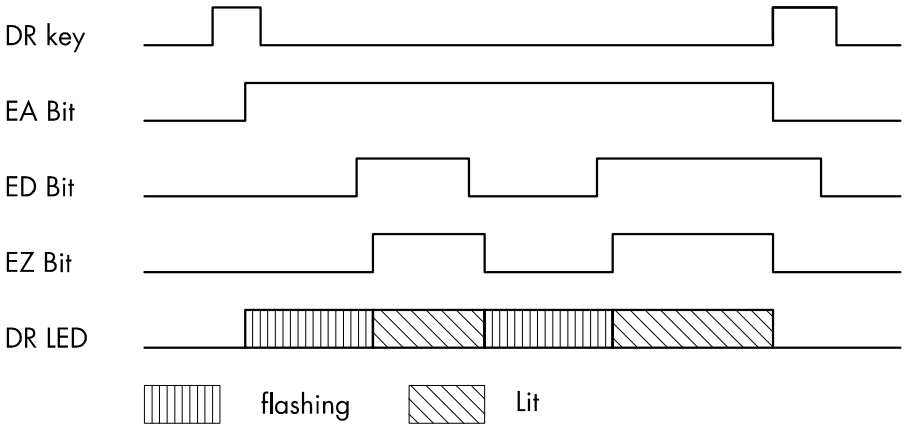


Fig.23: Pulse diagram - external data release

DR Key	Data Release key
EA-Bit	Editing request bit in the Read coordination byte
ED-Bit	External data release in the Write coordination byte
EZ-Bit	Editing status bit in the Read coordination byte
DR LED	Status LED in the Data Release key

A flashing status LED in the Data Release key always indicates that no external data release has been set. Editing is allowed if the PLC has set bit 0 to 1. In this case, the status LED in the Data Release key will light up. A blinking cursor will appear in the first entry field in the mask. The Editor specified by the user will be activated, depending on the selected variable.

(1) Windows is a registered trademark of the Microsoft Corporation

External data release handling:

- The PLC controls the external data release by switching the ED bit in the Write coordination byte.
- If the Data Release key is pressed, the EA bit in the Read coordination byte will be set and the status LED in the Data Release key will begin to flash.
- Approximately 100 ms later, the terminal will read the Write coordination byte from the controller. Editing is allowed if the external data release has been set (the ED bit in the Write coordination byte is set). In this case, the status LED in the Data Release key will light up and the EZ bit in the Read coordination byte will be set.
  Editing is not allowed if the external data release has not been set (the ED bit in the Write coordination byte is not set). In this case, the status LED in the Data Release key will continue to flash.
- During the editing process, the PLC can inhibit data entry at any time via the ED bit. Whenever the operating terminal detects that the ED bit has been reset, data input is inhibited, the status LED in the Data Release key begins to flash and the EZ bit is also reset.
  Whenever the operating terminal detects that the ED bit has been set, data input is allowed, the status LED in the Data Release key lights up and the EZ bit is also set.
- When the Data Release key is pressed again to exit the Editor, the last value that was edited is confirmed (i.e. it is transferred to the controller); the EA bit and EZ bit in the Read coordination byte are deleted.

3.6.6 PLC-Handshake

Handshake handling:

- If the terminal writes to a variable for which the PLC handshake option has been selected, the RA bit (refresh request bit) in the Read coordination byte will be set and data entry inhibited.
- The PLC can now, for example, provide a new data set for other input variables. The PLC will then set the RQ bit (refresh acknowledgement bit) in the Write coordination byte.
- Once the operating terminal detects that the RQ bit has been set, the terminal reads every input variable contained in the currently selected mask from the PLC again, then allows data inputs again and resets the RA bit.
- If the RQ bit is already set when writing to the variable for which the PLC handshake option has been selected, then the other input variables will be refreshed immediately and data entry will not be inhibited.

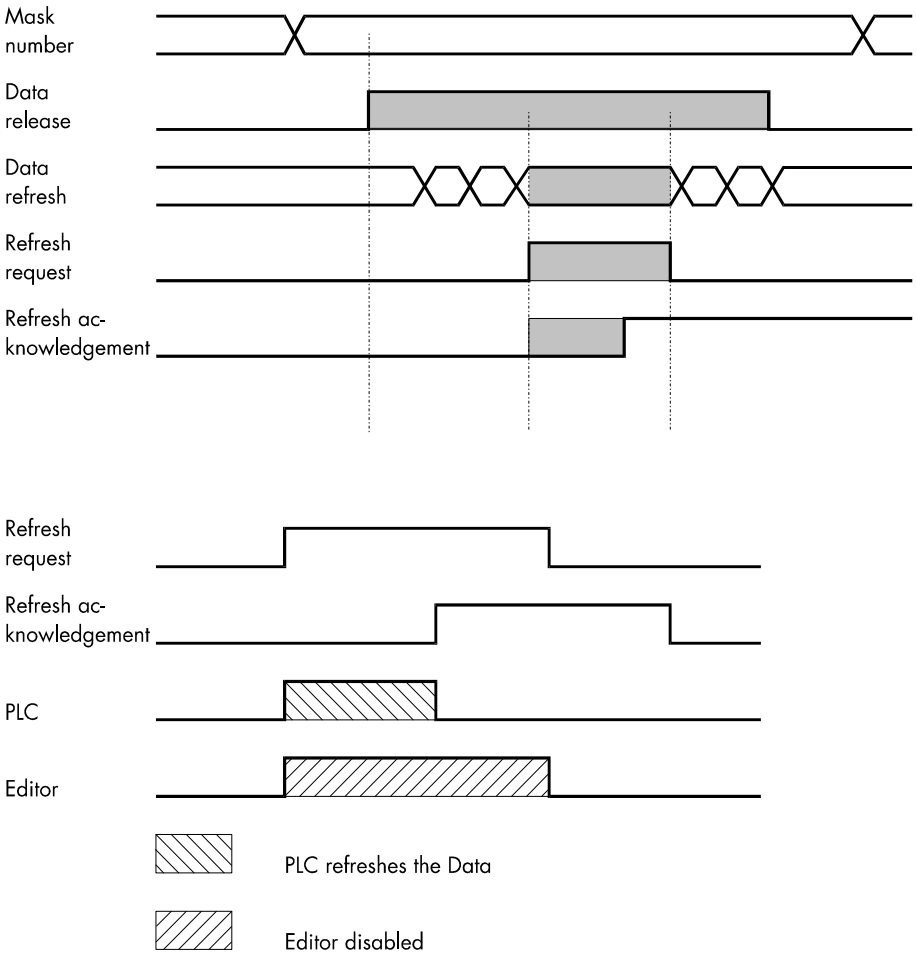


Fig. 24: Pulse diagram - PLC handshake

3.6.7 Refreshing One-Time Output Data

To refresh input or one-time output variables (variables are read only once and are displayed) that a mask contains, simply reactivate the mask (external mask selection).

3.6.8 Modified Data

The conditions that determine the transfer of modified data from the operating terminal to the controller are preset by the user in the variable definition. The following options are available, allowing the user to select the point of time at which the data are transferred:

- with the Enter key (standard)
- with the +/- keys or the Enter key (Incremental Editor)
- transfer continuously (upon modification), i.e. every intermediate status is transferred

In the Incremental Editor, every intermediate value is transferred.

The option to **transfer continuously** (upon every modification) is limited to values that are within the defined limits. If a value is entered that is outside the upper and lower limit, the operator will get a system message to this effect.
If a value with multiple digits is entered that

- exceeds the upper limit
  the digit entered most recently will not be transferred to the controller.

- falls below the lower limit,
  the previously valid value will be restored.

- falls below the lower limit (until the last digit has been entered) and then exceeds the upper limit (after the last digit has been entered),
  the previously valid value will be restored.

3.6.8.1 Input Plausibility Check

The plausibility check is performed on every editable numerical variable. This requires that an upper and lower limit is specified in the variable definition. These limits are displayed in the corresponding output format. System messages are generated when invalid values are entered. Further guidance can be found in the chapters Editors and System Messages.

3.7 Graphics

3.7.1 Graphical Objects (TSdos)

Images or other graphical information can be used in the terminal in the form of graphical objects. Each graphical object is based on a source image file stored in PCX format.

The images or graphics can be created using standard development tools, such as under Windows (1). Graphics programs that are already installed on the computer can also be used without any problem. The fact that standard tools can be used means that no additional operator training is necessary. The bitmaps (graphics) to be used are managed as monochrome PCX files in the TS programming software. These bitmaps can be defined as a series of details if necessary.

The specified bitmaps are managed by the programming software as "graphical objects" in a library. Each graphical object must be programmed with a unique, symbolic name. The objects can then be integrated into the masks or image lists by selecting these names.

The graphical objects are reusable and can be selected in a library list. They are thus made available to the user automatically, not only in different masks but also in different projects. The advantage of project independence is that once a graphical object has been defined in the system, it remains permanently usable. A bitmap can be any size from font (e.g. 6 x 8, 6 x 9) to display size (e.g. 256 x 128, 240 x 128, 256 x 64). It is always an integer multiple of a character (character coordinates).

Graphical objects are always managed "language-neutrally". The advantage here is that the same graphical objects are available in all languages.

A graphical object corresponds to a detail of the source image. This detail is defined relative to the top left-hand corner of the source image by means of its height and width (character units). The graphical object can be addressed anywhere in the programming system by means of a freely definable, 8-character, symbolic name.

When the PCX file is assembled into a loadable user description, it is customized by the Assembler to the particular display and display controller and then compressed. The output time and the amount of memory needed for the graphic are optimized at the same time. In order to shorten the CPU time, the bitmap object file is only generated if the PCX file acquires a more recent date or if the terminal identifier in the object file is no longer the same as the terminal type for which the Assembler is currently assembling.

3.7.2 Images (TSwin)

Any program installed by the user can be used by the programming software TSwin as a server for images.

Images can principally be made use of in two ways in TSwin:
1.) Using existing images: The images created in another program can be selected. A copy of the source image will then be stored in the TSwin project database and converted to a pixel graphics of the specified dimensions.
2.) Creating new images: The program to be used to create the new image can be launched from within TSwin. Once the image has been created, the server is exited and the image is stored in the TSwin project database. To display the image in TSwin and on the operating terminal, the image is converted into a pixel graphics of the specified dimensions.

By double-clicking the image in TSwin with the mouse, the server program used to create the image is opened automatically and the image can be edited. The scope of functions available to design the images depends on the features provided by the server program. The placement of images in TSwin can be pixel or grid oriented depending on the operating terminal type.

3.7.3 Graphics on Operating Terminals

Graphics are linked into the user description in the form of static background images for each I/O mask, dynamic variables on a bar chart, dotted lines on a curve or selection images. These basic elements allow the user to design a customized, full-graphics user interface. The curves (trendlines), bar charts and selection images are updated cyclically, so that a simple kind of animation is also possible.

3.7.3.1 Background Images

Background images can be created for masks on the terminal with the aid of graphical objects. Up to ten different graphical objects can be specified as background images for each mask. Background images are defined by means of the name of the graphical object and its position in the mask. You can also select display attributes for each background image. The following attributes are available: normal, inverse or flashing. The size of the displayed image is determined by the definition of the graphical object.

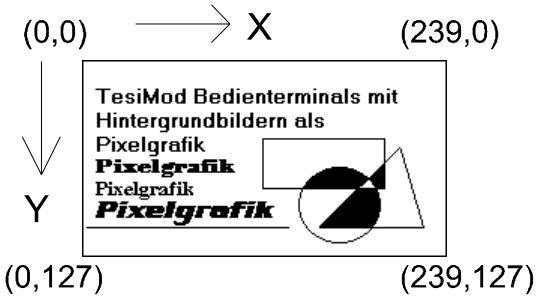


Fig. 25: Example of coordinates

Background images are output on the display one at a time. Their order is determined in the programming system by the order in which they are entered. The images are copied to the display memory in the selected mode. The maximum resolution and size of the bitmaps for background images are equivalent to those of the type of display that is used. Splitting the images into modular

segments has the advantage that they can be reused within the system, which above all enables memory space to be optimized. Background images can be used for all static applications, such as:
- Fixed display arrangement
- Representing the soft key row as icons
- Coordinate systems
- Information about production processes
- Static images or icons
- Representing overviews of plants or processes

If several different background images have been defined for a mask, they can also overlay one another. The write-over mode to be used for this process can be specified.

3.8 Recipes

Various logically related variables can be organized into units known as recipes. Unlike mask variables, recipe variables are not transferred to the controller immediately after being entered, but are stored in the terminal 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 terminal's Flash memory together with the user description or can be entered online on the operating terminal and are then stored in the battery-backed RAM. Data sets stored in the Flash memory must be copied to the RAM first before they can be edited. Data sets that have been edited remain in the battery-backed RAM.

Example for the usage of recipes:
Settings of a machine for manufacturing various products:

	Variable	**Value**	**Unit**
The product *clamp* requires:	Material	ST37-3	
	Feedrate	25.00	mm/s
	Setpoint Value Axis 1	43.50	mm
	Setpoint Value Axis 2	56.30	mm
	Cutting Angle	30	°
	Cutting Speed	110	mm/s

	Variable	**Value**	**Unit**
The product *shaft* requires:	Material	X20Cr13	
	Feedrate	20.00	mm/s
	Setpoint Value Axis 1	45.60	mm
	Setpoint Value Axis 2	51.20	mm
	Cutting Angle	45	°
	Cutting Speed	76	mm/s

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 while 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 mask for the recipe
- Recipe field in the mask
- Recipe buffer (address for the data area in the controller)
- Variable Data Set Number for Transfer from the Terminal
- Variable Recipe Number for Transfer from the Terminal
- Variable Data Set Number for Request from the Controller
- Variable Recipe Number for Request from the Controller
- System variables:

Variable Name	Linkage Partner	Description
SelectDSNr	Selection Text/Decimal Number	Display/Selection of the Data Set Number
SelectDSName	Selection Text Variable	Display/Selection of the Data Set Name
DestDSNr	Positive Decimal Number	Destination Data Set Number for Copy Process
DSCopy	Soft Key/Selection Text Variable	Activation of Data Set Copy Process
DSDelete	Soft Key/Selection Text Variable	Deletion of Data Set
DSDownload	Soft Key/Selection Text Variable	Loading of Data Set to Controller
ActDSName	Alphanumerical Variable	Entering the Name for RAM-Data Set
SelectRezeptNr	Selection Text/Decimal Number	Display/Selection of the Recipe Number
TabPgUp	Soft Key	Page up
TabPgDn	Soft Key	Page down
Break	Soft Key	Cancel input

3.8.1 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 mask, 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 Page up / Page down keys and the cursor keys can be used to scroll through long recipes in the recipe field.
All one-line formats and Editors that are available in I/O masks can also 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.

3.8.2 Processing 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.

3.8.2.1 Selecting a Recipe

All recipes are assigned a number from 1 to 250 when they are programmed.
You can select a recipe as follows:

- By means of a fixed assignment between the recipe and a mask. Whenever this mask is opened, the recipe window will contain the recipe that was specified when the mask was originally programmed. If a fixed assignment to a recipe has not been defined when the mask with the recipe window was programmed, the last recipe that was processed appears in this window when the mask is opened.

- By means of the system variable **SelectRezeptNr**. This variable can be edited using any Editor. It is a good idea, however, to use a selection text (coded text) or a selection field (TSdos only) and assign meaningful names to each recipe number.

3.8.2.2 Selecting a Data Set

Data sets can be assigned both a number from 1 to 250 and a name.
The data set numbers and names are allocated when the data sets are created, in other words either in the programming system for data sets stored in the Flash-Eprom or on the terminal 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).

A data set can be selected in one of the following ways:
- By selecting a new recipe. The associated data set with the lowest number is then selected for it automatically.

- By means of the system variable **SelectDSNr**. This variable can only be edited as a selection text (coded text) or a selection field (TSdos only). 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**. This variable can only be edited as a selection text (coded text) or a selection field (TSdos only). In this case, only the names of those data sets that are available for the active recipe are displayed.

3.8.2.3 Copying a Data Set

Only the active data set can be copied. To do so, the number of the destination data set is written to the system variable **DestDSNr** and then 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 between 1 and 250 (DSCopy = 2 searches for a free data set, while DSCopy = 3 overwrites the existing data set).
- 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 terminal.

If any of these conditions is not satisfied, the data set is not copied and a system error message is output.

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. A new name can be defined with the system variable **ActDSName**.

3.8.2.4 Deleting a Data Set

Only the active data set can be deleted. To do so, the value 1 is written 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 data set is not deleted and a system error message is output.

After the deletion, the data set with the lowest number in the current recipe becomes the active data set.

3.8.2.5 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 accept them. If the new values are not to be accepted, the system variable **Break** can be set to 1 to discard the contents of the buffer. It is a good idea to program one of the soft keys to this variable, in order to save time.

Another data set can not be selected until the buffer contents has either been accepted or discarded.

If the controller changes to a different mask while a data set is being modified, or if the external data release is cancelled again before the Data Release key is pressed, the buffer contents will likewise be discarded.

The modified data set is not transferred to the controller automatically. An explicit command from the operator or the controller is necessary first.

3.8.3 Data Set Transfer to / from a Controller

3.8.3.1 Transfer to a Controller

A data set can be transferred to the controller in the following ways:

- The active data set can be transferred to the controller by setting the system variable **DSDownload** to 1.
- The controller can issue a request for any data set. It first sets the variable addresses for the recipe and data set numbers to the required values. It then writes the value **7FFBH** to the address of the serial signaling channel (see section 3.22 *Cyclic Poll Area*).

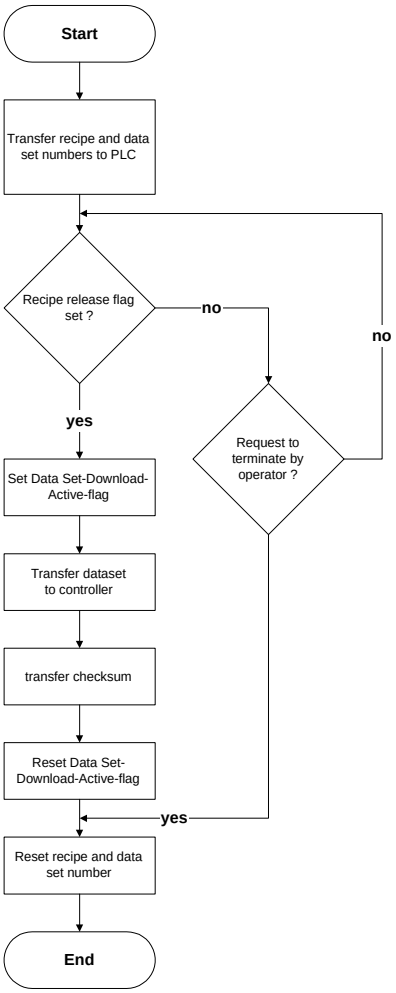


Fig. 26: Data transfer sequence to the controller

The sequence of a terminal-initiated data set transfer is recapitulated below in chronological order.
The process is described from the point of view of the terminal:

1. Transfers the recipe number to the variable addresses reserved for recipe numbers.

2. Transfers the data set number to the variable address reserved for data set numbers.

3. Waits until the data set transfer is released by the controller. For this process, the controller must set the Data Download Release bit (DDF bit) in the Write coordination byte to logical 1.
   Until this happens, the transfer can be cancelled again by setting the system variable **DSDnloadBreak** to 1. Once the transfer has been released by the controller, it can not be cancelled again by resetting the Data Download Release bit in the Write coordination byte.

4. Sets the Data Download Active bit in the Read coordination byte.
   This bit can be used by the controller to identify an incomplete data set transfer (communication interrupted, power failure). An incomplete transfer is, however, repeated the next time the terminal is booted.

5. Transfers the data set to the recipe buffer defined in the controller. A separate recipe buffer can be programmed for each recipe. The recipe buffer must be at least one byte larger than the data set (on account of the checksum) and consist of an even number of bytes (in other words, an extra byte must be added if necessary).

6. Transfers the checksum. The byte-wise XOR checksum (for the complete data set) is written to the address that follows the last byte of the data set. It provides the controller with additional protection against errors caused by incompletely transferred data sets.

7. Resets the Data Download Active bit in the Read coordination byte.

8. Resets the recipe and data set numbers to the variables that have been programmed in the controller for this purpose. The recipe number is set to 0 and the data set number to 255.

The various steps in this process can be monitored by the operator with the aid of the system variable **DSDnloadState**.
The name of the last data set to have been loaded in the controller is stored in the terminal for each recipe. This name can be displayed for the active recipe with the system variable **LoadDSName**. If a data set has not yet been loaded in the controller for this recipe, or if the data set which was loaded in the controller has already been deleted, a question mark appears on the display instead.

3.8.3.2 Transfer from a Controller

Certain applications may require that individual data set values in the controller are modified (for example, teaching) and that the modified data set is transferred back to the operating terminal. An option to transfer data sets from the controller to the terminal has therefore been provided.

A data set can be transferred from the controller to the terminal in either of the following ways:

- By writing a value to the system variable **StartUpload** to initiate a data set read process for the active recipe. If the value 1 is written to the system variable, the variables will be read individually from the addresses specified during variable programming. If the value 2 is written to the system variable, the variables will be read as a block from the data set buffer defined for the recipe. The number under which the data set is to be stored in the terminal can be specified via the system variable **UploadDSNr**. The default setting for this system variable is 0. If a destination data set number is not entered, the transferred data set will temporarily be stored under the number 0 and will then have to be copied to a valid number (1-250).

- The PLC issues a request for a data set read process. For this process, the controller writes the recipe number and data set number (the same as those for the transfer to the controller on request by the controller) of the destination data set to the variable specified for this purpose. Subsequently, one of the message numbers

  7FFDH (variables are read from their specified addresses individually) or

  7FFAH (variables are read from the defined data set buffer as a block)

  must be transferred. After the data have been transferred successfully, the recipe number is overwritten with 0. If there is no more free memory available in the operating terminal or if the specified data set number already exists in the Flash memory, the recipe number will be overwritten with 255. The data set number will not be overwritten in either case.

If the newly read data set overwrites a data set that already exists in the RAM with the same number, its name will be used for the newly read data set. If a data set does not yet exist with that number, the name will merely consist of blanks.

The read process can be monitored by the operator with the aid of the system variable **UploadState**.

3.8.4 Transferring Data Sets to / from a PC

It is possible to transfer data sets to or from a PC via the interface X3, in order to back up the data sets that have been stored in the terminal, process the data or supply the terminal with new data sets.
It is also particularly important to back up the data sets if a new user description is loaded in the 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 terminal again after the user description 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 terminal.
The data sets are transferred in a format that can be edited using a Text Editor (see section 3.8.4.3 *Structure of the Data Set File*).
The parameters for the X3 interface can be freely configured by means of the corresponding system variables. Merely make sure that the same parameters are set at the PC end. You can send or receive at the PC end with any suitable program, such as Windows Terminal (1).

(1) Windows is a registered trademark of the Microsoft Corporation

3.8.4.1 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**.

3.8.4.2 Transfer from a PC

The terminal 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 terminal 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 localized, at least approximately, with the aid of the system variable **RestoreLineNr**. This variable contains the number of the last line to have been received.
Data sets can only be stored in the terminal if their structure is still identical to the data set structure specified for the recipe concerned in the user description. This can be checked by the terminal 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 terminal, the newly received data set is similarly rejected without any warning to the operator.

3.8.4.3 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 terminal unchanged when it is needed.
If the data are to be processed further, for example, within the scope of production data acquisition, the operator 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
Parameters: none (date and time are output by the terminal as a comment)

End of file identifier:
Key: SFE
Parameters: none

Data set header:
Key: SDK
Parameters: Recipe number, data set number, data set name (as an ASCII string), data set size in bytes, recipe version number, write-over identifier

Data set variables:
Key: SDW
Parameters: Offset of the variables in the recipe, variable size in bytes, value of the variables (as a hexadecimal string)

End of data set identifier:
Key: SDE
Parameters: none

Explanations:
Recipe Version Number:
On creating or changing the recipe description in the programming system, 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 terminal, the downloaded version number and the version number stored in the operating terminal for the recipe involved must match. The downloaded data set will not be stored if they 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 terminal. 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 terminal together with the user description.

3.8.5 Printing Data Sets

The data set printout can be started from both the operating terminal and the controller.
To be able to initiate a printout from the operating terminal, either the system variable **StartRezPrint** must be placed into a mask or a soft key must be assigned accordingly. The active data set can be printed via the interface X3 by writing the value 1 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 indicated via the system variable **RezPrintState**.

To be able to control a print job from the controller, the data set number and recipe number must be entered into the appropriate variables first. The print job is then started 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 terminal) 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 terminal).

3.8.6 Memory Requirements for Storing Data Sets

The RAM in the terminal that is not required by the system (approximately 110,000 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 12,000 bytes for the default message buffer size (500 messages), so that a further 98,000 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
98,000 / (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.

3.9 TesiMod Message System

The message system is an integral part of the TesiMod operating concept. Messages are reactions to events that enable these events to be 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.

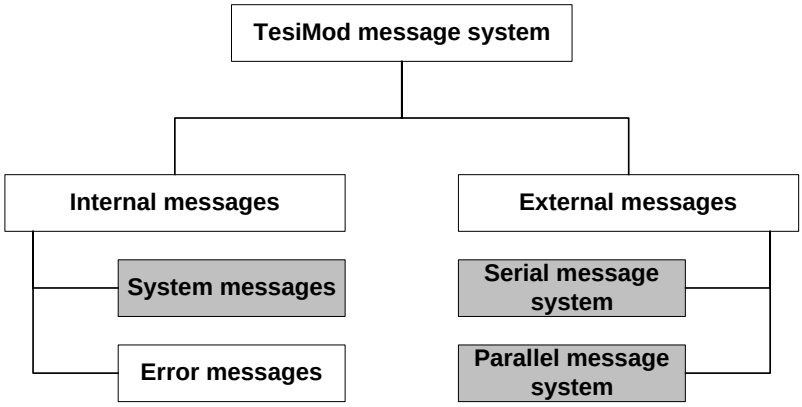


Fig. 27: Structure of the TesiMod message system

The grayed parts of the messages are freely configurable.

3.9.1 Internal Messages

Internal messages are all messages that are sent by the operating system to the operator. There is a further subdivision between system messages and error messages. The user (programmer) can not influence the generation of these messages.

3.9.1.1 System Messages

System messages are generated by the operating system as a result of internal plausibility checks. A system message is activated immediately after the corresponding event has occurred.
Pending system messages are signaled to the operator by a flashing Help key. If the Help key is pressed, the system message text will be displayed in its full length for as long as the Help key is held down. If several system messages are pending at the same time, they will be displayed in order of their system numbers, whereby the system message number "1" represents the highest display priority.
The user has the option of editing system message texts with the programming software. The size of one screen is available for each message text. The system message text can be freely designed using the terminal-specific font. Additional character attributes or graphics are **not** possible.
The output of the texts is language-specific, i.e. if the operator guidance is multilingual, the system messages are displayed in accordance with the selected language.
System messages are assigned in the programming software via the system message number. The system message number stands for a predefined event. A brief info 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 directly on one line, even on the smallest display in the TesiMod operating terminal series. When a selection is made in the programming software, the brief description is provided next to the number as a message name. A newly created system contains the following system messages with brief infos:

System Message No.	Brief Info
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 mask no
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	(Floating point) Number invalid
26	Loop through active
27	No data set address
28	Recipe unknown
29	Data set download

System Message 1 ..... "Wrong format"
This text indicates that an attempt has been made to enter an invalid data format into a variable field of the Numerical Editor. For example, the number of pre-decimal places entered exceeds the setting specified in the user description.

System Message 2 ..... "Value too large"
This indicates that an attempt has been made 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 description. If the text is deleted from the system message, no system message will be displayed if a value is entered that is too large. In this case, the maximum value that is valid is entered instead.

System Message 3 ..... "Value too small"
This indicates that an attempt has been made to enter a value into a variable field of the Editor that falls below the variable's lower limit. The lower limit is defined in the user description. If the text is deleted from the system message, no system message will be displayed if a value is entered that is too small. In this case, the minimum value that is valid is entered instead.

System Message 4 ..... "Replace battery"
This text is displayed when a test performed on the battery indicates that its capacity has fallen below the limit values. This capacity test is repeated every 60 minutes. To retain the recipe data and the data stored in the message memory, the battery must be replaced within 2-3 days of initial display of this message. To avoid loss of data when replacing the battery, the information in the respective operating terminal manual must be complied with. The same message appears when the battery is removed, switching the terminal off at this point will, however, result in the battery-backed data being lost.

System Message 5 ..... "Message overflow"
This text indicates that the system is unable to process the external messages quickly enough. Upon display of this message, one message has already been lost

System Message 6 ..... "New message"
This text is displayed when the Help key is pressed and the terminal has received a new external message whose priority exceeds the programmed threshold value and no direct selector key has been assigned to the message mask.

System 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).

System Message 8 ..... "Invalid mask no"
This text is displayed to indicate that a non-existing mask number has been transmitted by the PLC via the serial message channel.

System Message 9 ..... "Invalid message no"
This text indicates that the PLC has attempted to activate a non-existing message number.

System Message 10 ..... "Print log invalid"
The operator or the PLC has attempted to activate a non-existing print log.

System Message 11 ..... "Interface in use"
Interface X3 is already being used by another print job. An attempt has been made to transmit different types of data to the printer at the same time (e.g. to print recipes and messages).

System Message 12 ..... "Invalid Password"
The operator has entered a password which does not exist in the password list. With this message, the previous access authorizations (view and edit level) are reset.

System Message 13 ..... "Password unchanged"
The operator did not enter the same new password two times in a role.

System Message 14 ..... "Overvoltage"
The system has detected that the supply voltage is too high. Switch the terminal off immediately to avoid damage. Check supply voltage.

System Message 15 ..... "Data set protected"
An attempt has been made to modify individual values of a data set stored in the flash or to delete the entire data set.

System Message 16 ..... "Illegal data set"
The data set number specified as the destination for the data set copy process exists already or is outside the valid range (for example, flash).

System Message 17 ..... "Data set unknown"
A data set has been selected whose number does not exist in the data set list.

System Message 18 ..... "Data set memory full"
An attempt has been made to create a new data set, but the data set memory is full.

System message 19..... "Data set active"
An attempt has been made 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.

System Message 20 ..... "Data set transfer"
An attempt has been made to initiate a data set transfer to the controller even though the previously initiated transfer has not yet been completed.

System Message 21 ..... "Password missing"
An attempt has been made to switch to a password-protected mask or to edit a password-protected mask without having entered a password with sufficient authorization.

System Message 22 ..... "Editing mode active"
An attempt has been made to perform a change of mask while the terminal was in the editing mode.

System Message 23 ..... "Data set file error"
The data set file loaded from the PC to the terminal contains a syntax error. The error can be localized by means of the line number system variable.

System Message 24 ..... "Data set format"
The size or internal version identifier of a data set loaded from the PC to the terminal and the corresponding values in the programming software do not match.

System Message 25 ..... "(Floating point) Number invalid"
The bit pattern read from the controller is not valid for a floating point number. The number is displayed as 0.0.

System Message 26 ..... "Loop-through active"
The selected action was not performed due to an active loop-through operation.

System Message 27 ..... "No data set address"
The addresses for the data set transfer did not exist at the time of the controller's request.

System Message 28 ..... "Recipe unknown"
An attempt has been made to select a recipe that does not exist in the terminal.

System Message 29 ..... "Data set download"
A data set transfer to the controller (download) has been initiated, but the Data Set Download Release bit in the Write coordination byte (bit 4) has not yet been set by the controller.

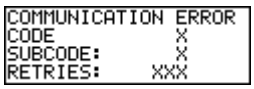
3.9.1.1.1 Suppressing the Display of System Messages

The operating concept provides you the option of suppressing the display of system messages. The display of a system message can be prevented by erasing the system message in the programming software.

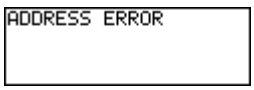
Example:
System message 7 "Message buffer full" is to be suppressed. Older messages or messages with a lower priority are to be overwritten.
The system message text in the programming software is erased.
By suppressing the display of this system message, the user agrees that incoming messages automatically overwrite the oldest messages or those with the lowest priority once the message buffer is full.

3.9.1.2 Error Messages

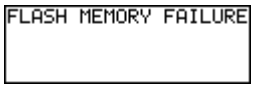
The messages listed below are displayed by the operating system. The message texts are part of the operating system and are displayed in English. The size of the texts has been chosen in such a way that they can be displayed on every terminal. The output of these texts can not be suppressed and their contents can not be modified. The term "error message" is used because the terminal does not operate in accordance with the true meaning of the TesiMod standard mode while these messages are displayed. In addition to true system errors, various conditions and processes are also described.



..... 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 application documentation. 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.



..... This message may be displayed during a download. The S3 file addresses physical addresses in the terminal. 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.



..... 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
```

..... An error has occurred during transmission of the user 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
```

..... Error during transmission of the user 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
```

..... Transmission format of the user 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.

```
1. TURN POWER OFF
2. RESET DIP-SW 4
OTHERWISE ALL FLASH-
DATA WILL BE LOST !!
```

..... The mode selector switch S4 was at the "on" position when the supply voltage for the terminal was switched on. The Flash data will be retained if the following instructions are complied with. Switch the terminal off, set S4 to "off" position, switch terminal on - data will be retained and the terminal will function as before. If S4 is set to the off-position while power is on - data will be lost, terminal switches to the download mode!

```
DIFFERENT MASK  VERS
          EPROM FLASH
VERSION    XXX   XXX
REVISION   XXX   XXX
```

..... The version of the programming system and the operating software of the terminal do not match. This error occurs if the wrong operating system version was selected for compilation of the user description. The two program versions must match.

```
DIFFERENT DRIV  VERS
          EPROM FLASH
VERSION    XXX   XXX
REVISION   XXX   XXX
```

..... The protocol driver loaded via the programming system and the terminal’s operating system do not match. The two program versions must match.

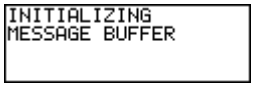
```
!!!!! WARNING !!!!!
NONE DEFAULT PARA-
METERS ON SERIAL
PORT X2 USED
```

..... The parameters of the interface 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.

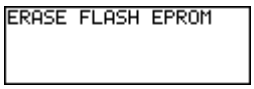
..... The operating system can not find a protocol driver in the user description loaded into the terminal. Select a protocol, recompile the user description and activate another download.

..... The protocol selected in the programming system when creating the user description and the terminal hardware are not compatible. For example, the Interbus-S protocol driver has been loaded to a device with standard interfaces.

..... A self-test is performed when the terminal is switched on. This error message is generated if a key is pressed while the keyboard is being checked. Please release the key. If this message appears when no key is pressed, the message indicates a defective key-board!

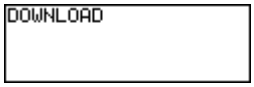


..... When the terminal is switched on, all messages in the terminal are sorted. This initialization process requires a certain length of time based on the number of stored messages. The message is always generated, but is only displayed for a very short time period or is not visible at all.

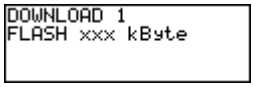


..... Is displayed while the mask memory is being erased. All of the programmed data are erased at this point.

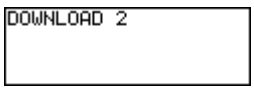
..... This message indicates that the Flash has been erased. Interface X3 is initialized for the download mode.



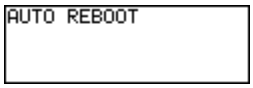
..... This message indicates that the terminal is ready to receive the new user description via interface X3 (TSdos only).



..... The operating terminal 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 (TSwin only).

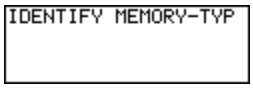


..... The operating terminal indicates that it is ready for a download with the new interface parameters. If no data are received within 20 s, the operating terminal will return to the DOWNLOAD 1 condition (TSwin only).

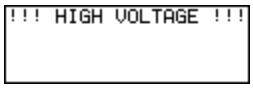


..... The operating terminal will reboot within the next few seconds.

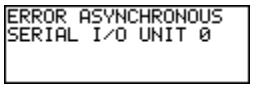
..... The operating terminal reported its parameters during the startup process:
- CPU frequency in MHz
- Size of the Flash memory in Kbytes
- Eprom version number XXXXXXXX
- Loaded PLC driver YYYYYYYY



..... The Flash memory type used is being identified.



..... The voltage applied to the operating terminal is too high. This message will not disappear until the specified supply voltage has been reached.

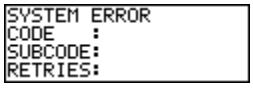


..... Initialization of the serial interface (unit 0 or unit 1) failed.

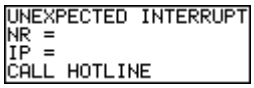
..... The program level of the SUCOnet K card and the current protocol driver are not compatible. Retrofit the operating terminal or use the appropriate driver version. The subcode specifies the level of the SUCOnet K card.

..... The program level of the keyboard card and the current firmware are not compatible. Retrofit the operating terminal. The subcode specifies the level of the keyboard card.

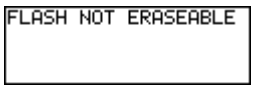
..... Indicates a successful update operation. The operating terminal reboots automatically.



..... A fatal error has been encountered. If this error message is generated, make a note of the firmware level and the hardware version and contact Sütron electronic GmbH, Kurze Straße 29, D-70794 Filderstadt, hotline no.: ++49 / (0) 7 11 / 7 70 98 55.

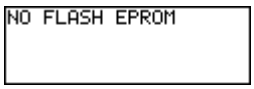


..... An unexpected interrupt has occurred. Make a note of the interrupt number (NR) and the level of the program counter (IP) and contact Sütron electronic GmbH, Kurze Straße 29, D-70794 Filderstadt, hotline no.: ++49 / (0) 7 11 / 7 70 98 55.

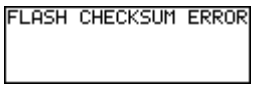


..... Is displayed after the terminal has been switched on or prior to a download to indicate that the Flash-Eprom can not be erased.

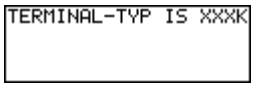
..... Is displayed at the beginning of a download to indicate that the S3 file is not the correct type for the terminal being used.



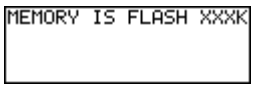
..... This message is displayed to indicate that no Flash supported by the programming algorithm is found.



..... The user description stored in the FLASH contains errors. This error may occur at the end of a transmission, e.g. if the transmission is not complete or after a terminal is switched on that has a defective memory.

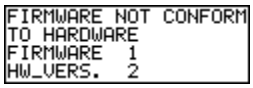


..... An attempt has been made to load a S3 file which was intended for another terminal type. When this error occurs, the correct type for this terminal is displayed at the "XXXX" position. Recompile using this selection in the programming system.

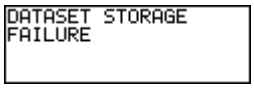


..... An attempt has been made to load a S3 file which was created for a larger mask memory. The amount of memory space requested by the S3 file and the memory available in the terminal do not match. When this error occurs, the memory size available in the terminal is specified, in kBytes, at the "XXX" position. This value must be specified in the programming system when compiling.

..... An error message that should not occur, but which exists. The operating system of the terminal generates this error if proper operation is no longer possible due to a lack of plausibility. To be able to reconstruct 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.



..... If this error message is displayed, contact Sütron electronic GmbH, Kurze Straße 29, D-70794 Filderstadt, hotline no.: ++49 / (0) 7 11 / 7 70 98 55. Before calling, make a note of the firmware and hardware version. The operating system of the operating terminal switches into an endless loop to prevent damage to the device.



..... A checksum error has been detected when checking the memory areas of the recipe data sets. Either the battery or the RAM-memory is defective.

3.9.2 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 terminal can be either serial or parallel, regardless of whether the messages are process 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 terminal and the controller by means of a message number. The associated texts and variable specifications are stored in the terminal together with the user description. The function of a message and its contents are determined by the user when the user 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. With TSwin projects, parallel messages can optionally be stored in the serial message memory and as a result also evaluated statistically. If the message contains a variable, its value will be frozen in the message memory.

3.9.2.1 Structure of an External Message

An external message is made up of:
- a message number between 1 and 9999
- a message text with up to 80 characters (TSdos) or
- a message text with up to 255 characters (TSwin)
- one variable at maximum.

When creating a new application, existing messages (one or all) are reusable.

3.9.2.1.1 Assigning Message Numbers

With external messages, message numbers also determine the priority of a message, with message no. 1 being the highest and message no. 9999 being the lowest priority. It is not necessary that message numbers are assigned contiguously; this allows messages with related contents to be grouped.
The assignment of message numbers for status messages always starts with "1". Make sure that the serial and parallel message system do not overlap. If the two message systems are to be independent from one another, ensure that the message numbers for the serial system start above those for the status messages. In addition, message numbers and mask numbers must be coordinated accordingly when the programming is carried out (see chapter 3.9.2.3.1 *Full-Page Message Output*).
The system allows status message texts to be used in the serial message system.

3.9.2.1.2 Message Buffer Size

The total message buffer size is designed for management of up to 3000 messages. Such a large number of data requires a corresponding computing capacity when the messages are sorted, resorted and initialized. Since this large number may not always be needed, the user has the option of setting the maximum system message buffer size to suit his own needs. The default buffer size allows 500 entries.
When determining the message buffer size, it must be considered that approximately 50 pages of paper are needed when the entire message buffer containing 3000 messages is printed!
The message buffer is output in the message mask. The message sorting criteria can be set via a system variable.

3.9.2.1.3 Message Texts, Variables

The maximum text length including a formatted variable is 80 (TSdos) or 255 (TSwin) characters. The programming system does not allow longer texts to be input. All characters that are available in the respective terminal can be used in standard size. One output variable can be included per message text. The variable output format is the same as that of one-time output variables (variables are transferred only once and are then displayed) in I/O masks. Individual messages can for example be modified with coded text or used for several statuses. The output format of the message line can be modified online in a configuration mask for the message mask.
The capabilities for serial and parallel messages are the same.

Example: Complete message format:

```
No.  Date     Time     Text1        Variable Text2
1234 25:08:92 11:30:00 Temperature  285 °C   at Meas.
    Pt.07
```

The message consists of the following elements:

1234	4-digit message number
25.08.92	Date, is stored in the terminal when the message is recorded by the terminal
11:30:00	Time, is stored in the terminal when the message is recorded by the terminal
Temperature	Text (1) preceding variable
285	Value of the variable at the time of the message
°C at Meas. Pt.07	Text (2) following variable

3.9.2.1.4 Sorting Messages

Messages can optionally be displayed in the message mask according to their time of arrival or priority. The desired option can be selected when the system is programmed. If both possible message systems are used, it is possible to select the settings separately. The settings are stored in the system variables **RepmanRepSortCrit** and **RepmanSortCritP**. They can be changed online on the operating terminal using a configuration mask, for example. If a configuration option is not offered on the operating terminal, the settings selected by the user will be used.

Sorting algorithm for the serial message system:
0 - by priority
1 - by time (newest first)
2 - by time (oldest first)

Sorting algorithm for the parallel message system:
0 - by priority
1 - by time (newest first)
2 - by time (oldest first)

3.9.2.1.5 Message Priority for Direct Display

The priority of a message is determined by its message number. The higher the message number, the lower the priority. The value that represents the upper limit for the message number that is to be signaled on arrival by a flashing LED or by outputting a system message can be entered, via the programming system, into the parameters file (TSdos) or the system parameters of the message system (TSwin).

If the value = 0 is entered, newly received messages will be not be signaled.

3.9.2.1.6 Printing the Message Memory

The memory contents of the serial and parallel message systems can be printed either in full or in part.

The entire contents of the **serial** message memory are printed if the system variable **PrintAllRep** is set to 1 (formatted printout) or 2 (full-length printout).

The entire contents of the **parallel** message memory are printed if a soft key that is linked to the system variable **PrintAllState** is pressed.

To print the message memory in part, the messages to be printed must be selected in the message mask. This is done by pressing the Data Release key in the message mask and selecting the messages in the message field using the Cursor up and Cursor down keys. The print job is started by pressing a soft key 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 mask and be edited online.

3.9.2.2 Message Mask, Status Message Mask

In TSdos, the message mask is a special I/O mask which has been modified to accommodate the output of messages. In TSwin, the same functions are achieved by inserting a message field into an I/O mask. The key assignment for a message mask or status message mask is specifically designed to allow convenient navigation within the extremely large message memory.

- Data release not active	Cursor up:	Moves the cursor up one message. The cursor can be moved upwards until the top message is reached.
	Cursor down:	Moves the cursor down one message. The cursor can be moved downwards until the bottom message is reached.
	Cursor left:	Moves the cursor up one screen (repeat function). The cursor can be moved upwards until the message at the top is reached.
	Cursor right:	Moves the cursor down one screen (repeat function). The cursor can be moved downwards until the message at the bottom is reached.
	Minus:	Moves the cursor to the bottom-most message.
	Plus:	Moves the cursor to the top-most message.
	Delete:	Inactive
	Data Release:	Enables the editing mode, provided the external data release has been set and the entered password has a sufficient access level.

	Enter:	The entire message at the cursor position will appear on the display.
- Data release active	Cursor up:	Selects individual messages from the current cursor position on upwards.
	Cursor down:	Selects individual messages from the current cursor position on downwards.
	Cursor left:	Selects messages in page-mode from the current cursor position on upwards until the topmost message is reached. No repeat function.
	Cursor right:	Selects messages in page-mode from the current cursor position on downwards until the bottommost message is reached. No repeat function.
	Delete:	Deletes all selected message entries. If no selection is made, the message at the cursor position will be deleted.
	Minus:	Selects all messages from the current cursor position down to the last message.
	Plus:	Selects all messages from the current cursor position up to the first message.
	Data Release:	Exits the editing mode, provided the external data release is still set.

3.9.2.2.1 Direct Selection of the Message Mask

In the programming software, a function key can be linked to a message mask. This function key can then be used to switch to the message mask from within any mask. This provides another means to reach the message mask, in addition to the access via a selection menu. In this case, the arrival of new messages will be signaled by the integrated function key LED which will begin to flash (instead of the Help key).
Direct access to the message mask can be obtained by pressing the flashing function key. If this function key is pressed again, the message mask will automatically be exited and the previous mask will reappear on the display.

3.9.2.2.2 Output Formats for Messages

The following information can be output with every external message:
- Message number
- Date
- Time of day
- Message text
- Value of a variable at the time the message was generated (if available only)

Various system parameters are available to influence the representation of a message in the

message mask or the message printout. These parameters can be set in a configuration mask, provided such a mask is programmed.
The selection or deselection of message elements is carried out by means of system variables.

Serial Messages	Parallel Messages	Influenced Element
- RepoutNr	**RepoutNrP**	Message number
- RepoutDate	**RepoutDateP**	Date
- RepoutTime	**RepoutTimeP**	Time
- RepoutAnzYear	**RepoutAnzYearP**	2 or 4 digit representation of the year

This allows the length of the message line to be influenced. Modifications to the output format have no impact on the stored information.

Possible output versions are as follows:

Complete message format:

```
No.  Date     Time     Text1       Variable Text2
1234 25:08:92 11:30:00 Temperature 285 °C   at Meas.
    Pt.07
```

Versions:

```
1234 25:08:92 Temperature 285 °C   at Meas. Pt.07
1234 11:30:00 Temperature 285 °C   at Meas. Pt.07
1234 Temperature 285 °C   at Meas. Pt.07
25:08:92 11:30:00 Temperature 285 °C   at Meas. Pt.07
11:30:00 Temperature 285 °C   at Meas. Pt.07
25:08:92 Temperature 285 °C   at Meas. Pt.07
Temperature 285 °C   at Meas. Pt.07
```

3.9.2.2.3 Zooming Messages

Messages are displayed in a one-line format in the message mask 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.

Message mask line, for example on the BT20:

```
1234 25:08:92 11:30:00 The measuring point in
```

Zoomed view:

```
1234 25:08:92 11:30:00
Measuring point 137 in the furnace has a
temperature of 285 °C
```

The zoomed view remains active for as long as the Data Release key is held down. With smaller displays (for example, the BT5 with 4 x 20 characters) only the message text is zoomed. The terminal type that is to be used must be considered when the text is programmed, to ensure the lines are wrapped correctly.

3.9.2.2.4 Acknowledging Messages

Message acknowledgement in the controller can be carried out by means of variables. Various Editors or function keys (soft keys) are suitable for this purpose. The acknowledgement enables the controller to delete the message and verify the status of the process.

3.9.2.3 Serial Message System

A 2 byte compartment is used in the cyclic poll area for the transmission of serial messages. The byte order depends on the selected data type of the poll area (see Poll Area). The PLC stores a 16 bit message number in this transmission compartment. The TesiMod operating terminal polls the entire poll area of the PLC at cyclic intervals and transmits the serial message in the process. Upon detecting a message (message number > 0), this message is stored in the internal message memory of the operating terminal and the compartment in the PLC is reset to zero. The value 0 in the compartment indicates to the PLC that the message has been fetched by the terminal. The polling time of the cyclic data area is configurable.
The same procedure is used to address external masks and message masks. Whenever the number transmitted corresponds to a mask number, this mask is displayed. If a mask and a message text exist for this number, the mask (message mask, 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 entered in the serial data compartment as 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.

3.9.2.3.1 Full-Page Message Output

The full-page message is a combination of message processing and external mask selection. For a full-page message mask output, a mask and a message text must be programmed under the same number. The controller calls up the "external mask" through the serial message channel. When it is called up, the mask is displayed and the associated message text is entered into the message memory. As the display contents can be chosen freely, it is possible to realize a message mask, a full-page error output or other contents types. To be able to return to the previous menu from here, at least one mask parameter must be programmed for "Previous mask". Message masks can also consist of several masks or even complete structures for troubleshooting.
It goes without saying that a separate, full-page help text can also be programmed for each full-page message.

3.9.2.3.2 Messages Directly to a Logging Printer

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.
The output of the messages to a printer can be influenced by 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 mask.
The settings for the printout can be changed on the terminal online.

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.

3.9.2.3.3 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 deletion variable must be specified in the Message System option of the parameters file using the programming software. Two bytes are needed in the controller for the variable.
The terminal always checks the deletion variable in the controller once it has received the deletion sequence (deletion code $\mathbf{7FFE_H}$ via the serial message channel). The internal message buffer is deleted if the deletion variable contains the bit pattern $\mathbf{E216_H}$. The deletion variable increases protection against unintentional deletion.
If deletion is not required, the variable should be reset or no symbolic name should be specified in the programming software.

3.9.2.3.4 Information about the Serial Message System

How do messages reach the terminal?
In the controller, a word variable, as a part of the cyclic poll area, is reserved for the message number. This variable is polled by the operating terminal. If the variable value is greater than zero, this indicates that this is a numeric value of a message, i.e. the message number. The message number is placed into the terminal's message memory and the variable in the controller is overwritten with zero. This is treated as the acknowledgement for the controller that the terminal has fetched the message. The next serial transfer of messages can now take place.

How is a new message recognized?
By flashing of the Help key or by a flashing function key, the terminal signals to the operator that a new message has been received. This visual indication only appears when a value has dropped below a limit (message priority for direct display).

How is the message mask configured?
The message mask is preceded by an I/O mask, the configuration mask, and output formats can be defined in it by means of various system variables.

How can the most current messages be output in any chosen I/O mask?
A system variable **RepoutRepText** is available which always contains the message most recently received or the one with the highest priority. The content is retained up to the next change. The output is always left-justified and as defined in the configuration menu. There are also the following system variables for complete or multiple-line output:

RepoutRepText21
RepoutRepText41
RepoutRepText61

See description of the system variables for more information.

Where are messages displayed?
In the message mask. The message mask can be reached via the node mask, the I/O mask, control keys or soft keys.
Owing to the fact that many control keys are used in the message mask, it can only be exited by pressing the Home key or function keys.

3.9.2.4 Parallel Message System (Status Messages)

The parallel message system complements the serial message system. The messages are transmitted in parallel and are evaluated in the terminal. During this process, the current message status is compared with the prior status in the terminal. These messages are automatically deleted from the memory once they are no longer pending. New messages are added to the memory. The current status of the messages can be output.
Date and time are included in every message to indicate the point of time at which a message has been generated.
The message buffer length is limited to a maximum of
- 64 data words or 128 bytes (TSdos) or
- 256 bytes (TSwin).

The length is to be set in the parameters file (TSdos) or in the system parameters for the message system (TSwin) of the programming system. There may be restrictions regarding the length depending on the protocol (see chapter on controller and bus connections).
Status messages are retained in the message memory only as long as they are being reported by the controller.
Status messages can be transferred on a time or event controlled basis.

3.9.2.4.1 Number of Bytes for Status Messages

This parameter specifies the number of bytes to be transferred with the parallel message system. The size depends on the number of status messages. The absolute size also depends on the defined data type (address). For example, if a word address has been defined and the number of bytes is odd, then this number will automatically be rounded up.

3.9.2.4.2 Image of the Status Messages

For status messages, the starting address of the storage area containing the message information is entered here. The symbolic name is assigned to a fixed address. The assignment of message number to bit information depends on the data type of the controller variable. The data type has been processed in a protocol-specific manner since the structures of byte, word and Lword are not identical in the supported controllers. It is important that once an assignment method is chosen, it is not changed!

Byte-oriented assignment:

Byte address + 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte address + 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte address + 2	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte address + 3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
:				:				
Byte address + n	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Message 1 = Byte 0, Bit 0
Message 2 = Byte 0, Bit 1
Message 3 = Byte 0, Bit 2
:
Message 8 = Byte 0, Bit 7
Message 9 = Byte 1, Bit 0
Message 10 = Byte 1, Bit 1

Word-oriented assignment:

Word address + 0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 3	Bit 2	Bit 1	Bit 0
Word address + 1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 3	Bit 2	Bit 1	Bit 0
Word address + 2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 3	Bit 2	Bit 1	Bit 0
Word address + 3	Bit 15	Bit 14	Bit 13	Bit 12	Bit 3	Bit 2	Bit 1	Bit 0
:				:				
Word address + n	Bit 15	Bit 14	Bit 13	Bit 12	Bit 3	Bit 2	Bit 1	Bit 0

Message 1 = Word 0, Bit 0
Message 2 = Word 0, Bit 1
Message 3 = Word 0, Bit 2
:
Message 16 = Word 0, Bit 15
Message 17 = Word 1, Bit 0
Message 18 = Word 1, Bit 1

The same rules apply to other data formats.

3.9.2.4.3 Time-Controlled Transfer of the Status Message

The transmission of parallel messages is activated either in a time-controlled mode by the terminal or an event-controlled mode by the controller. The entire message buffer is transmitted at once.
Both transmission methods can be implemented at the same time. This permits longer polling intervals (every 5 to 10 s) to be defined. Critical or important messages can additionally be transferred on an event-controlled basis. The PLC is automatically polled for the status messages after the polling time has elapsed. The selected polling time should not be too short since the status message transfer can take up a longer amount of time (depending on the number of messages and the protocol).
No event-controlled transfer will take place if the polling time is set to 0,0 s.

3.9.2.4.4 Event-Controlled Transfer of the Status Message

The event-controlled transfer is activated by the controller by writing the event code $\mathbf{7FFF_H}$ to the serial message channel. This code causes the parallel message system to be updated to ensure the messages in the operating terminal's parallel message buffer are current.
To be able to use the event-controlled transmission (also for polling times of 0,0 s), the parameters file must contain the symbolic variable name and the length of the status messages.

3.10 Help System

A screen-sized help text can be assigned to each mask and editable variable. If no specific text is programmed, a default help text will be displayed instead. This default help text can be freely specified by the user through the programming system.

3.10.1 Default Help Text

If no help text has been defined for a particular mask, the default help text specified in the programming system is displayed. This text is always present. If no text is programmed, a blank mask is displayed instead.

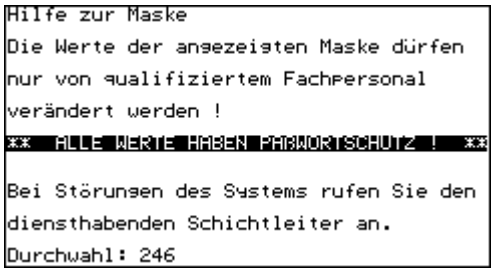


Fig. 28: Example of a default help mask

3.10.2 Help Text For Masks

A help mask can be created for every programmed mask. The link between the current mask and help text is specified in the parameter settings.
All of the valid characters can be used to design the help mask. Help texts can be called up directly by means of the Help key. As long as no data release has been requested, the help text for the mask is displayed. Once the data release is requested, the help text for the variable is shown.

3.10.2.1 Help Text for the Message Mask

Help masks can also be created for messages masks. Help texts are only available for the mask. Any variables that may be embedded can not be linked to a help mask.

3.10.3 Help Text For Variables

The functionality is the same as that of help texts for masks. It is, in particular, possible to display the valid range of values. The link between the help mask and the variable is specified in the variable parameters.

3.11 Function Keys

Another important feature of the TesiMod operating concept, in addition to the masks, are the function keys and their LEDs. Function keys are user-programmable. They can be used as direct selector keys for selecting more masks 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 masks as well as to entire menu structures.
If the operating terminal is fitted with parallel outputs, 8 function keys can be assigned to the outputs directly. The reaction time after pressing a key is approximately 30 ms. Before a function key signal is provided, the terminal debounces the key, thereby ensuring that it has actually been "pressed".
In the programming system, the combination of direct selection and control can be programmed for function keys and soft keys. 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 mask, the stop code may have already changed!

3.11.1 Direct Selector Keys

Direct selector keys are function keys which have been programmed to directly call up a specific mask. Pressing this function key causes a new mask to be activated.
This change of mask is **not** possible if the data release has been requested (status LED in the Data Release key is flashing or lights up) in a mask without automatic data release.
With direct selector keys, a speedy and convenient operation can be obtained.

3.11.2 Function Keys of the Controller

In addition to being programmed as direct selector keys, function keys can also trigger a function in the PLC. This requires that instead of a change of mask, the symbolic name of a controller variable is assigned to the function key. This assignment is performed in the user description. The function key can be defined to either "set" or "reset" a variable when pressed/released. If the key is assigned the function "set" the specified value will be transmitted to the variable in the controller.
If the number 1 is entered as a value,
- a flag bit will be set to logical "1"
- a flag byte will be set to the value 01_H
- a flag word will be set to the value 0001_H
- a double word will be set to the value 00000001_H

For values greater than "1", at least a byte address must be specified for the variable. Values greater
than "1" can be assigned in TSwin only.
If the number 3 is entered as a value,
- a flag byte will be set to the value 03_H
- a flag word will be set to the value 0003_H
- a double word will be set to the value 00000003_H

3.11.3 Soft Keys

In the TesiMod standard mode, soft keys are function keys that perform a mask-related function (they perform different functions in different masks). A description of the function that a soft key performs in a particular mask should be displayed in that mask. Several means are available to design this description such as images, background images, selection images, static texts and selection texts (coded text).
If a selection text (coded text) is used for labeling a soft key, the function key can be used for multiple functions in one and the same mask.
The action to be performed is determined in the controller by linking the
- mask number
- number of the selection text (coded text)
- variable value that is transferred with the soft key.

The number of keys that can be used as soft keys depends on the type of operating terminal used.

Example: A soft key (F1) is to be capable to switch a pump off and on in mask 10

1. Create a text list (pump) with two entries

Value	**Text**
0	Switch Pump OFF
1	Switch Pump ON

2. Define the variables
- Soft key labeling M 100.0
- Soft key status M 100.1
- Image of the mask MW 110

3. Create the mask (number 10)
- set up a controller variable (M 100.0) next to or above a function key:
  link a selection text variable for cyclic output with the text list (pump)
- link the function key F1 of the mask with the variable Soft Key Status (M 100.1), (set/reset)

4. Create the PLC program to perform the following: O 32.0 is to be used to control the pump
- Evaluate mask number (MW 110); (must have the value 10)
- Create the edge evaluation for M 100.1
- Create the ELTACO function for pump output O 32.0
- Set flag M 100.0 to 0 when the pump is on
- Set flag M 100.0 to 1 when the pump is off

3.11.3.1 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 (cyclic 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.

3.11.3.2 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 setting (value 1) or resetting (value 0) a variable (TSdos) or of assigning any values to it (TSwin). The transfer procedure is independent of the mask parameter assignment. Thus, if a control key is to carry out a specific function in a mask, it should not be programmed as a "mask selector key" at the same time (in other words, one which initiates a change of mask). The mask-specific evaluation is identical to that of the function keys.

3.11.4 Function Keys Controlling Parallel Outputs

Groups of 8 function keys can be linked to parallel outputs (semiconductor outputs). The keys are read in by the software, debounced and then mapped to the outputs. The reaction time to the outputs is around 30 ms. As the keys act on the PLC very quickly and independently of the protocol, they are ideal for controlling axes or for programming jogging mode.
The power output allows direct control of PLC inputs.
If a PLC variable has been programmed for the function key in addition to the output, it is of course also sent to the controller, though with a small time delay.

3.11.5 Status LEDs in the Function Keys

A 2-bit information is provided in the cyclic poll area for each status LED in a function key. One of the two bits is responsible for activating or deactivating the corresponding status LED while

the other bit represents the flash attribute for the status LEDs. Status LEDs can only be influenced by the controller.

Bit is on/off	Bit flashes	Status of the LED
0	0	LED is off
1	0	LED is on
1	1	LED flashes
0	1	LED is off, flash bit can remain set

This is not true if
- a function key has been programmed for direct selection of a message mask (direct selector key)
- a value greater than 0 (zero) has been entered for the message priority
In this case, the LED in this function key can not be influenced by the PLC but is controlled entirely by the message system.

If the number of status LEDs provided by the TesiMod operating terminal being used is smaller than could be controlled here, then the excessive bits have no function.
For the arrangement of the bits for the individual status LEDs see the section on the cyclic poll area.
To reduce transfer times, the length of the poll area should be selected such that only the bytes required for the status LEDs are transferred.

3.12 System Parameters

All system parameters are set to default values.
They are loaded into the operating terminal together with the project created. The system parameters contain the value settings for the:
- General parameters
- Poll area
- Terminal clock
- Running time meter
- Message system
- Variant buffer
- Password management
- Printer interface
- Gateway
- Data set transfer
- Parallel outputs (optional).

3.12.1 System Parameters: General Parameters

The general parameters refer to all generally applicable functions of the operating terminal.
- Polling time for cyclic variables
- Screensaver settings (optional)
- Automatic download (TSwin only)
- Symbolic addresses for: Image of the mask number
  Image of the DIP switch
  Read coordination byte
  Table index
  Keyboard image

3.12.2 System Parameters: Poll Area

System parameters for the poll area include the symbolic address of the variable, the polling time and the size of the poll area.

3.12.3 System Parameters: Terminal Clock

System parameters that can be specified for the terminal clock are the symbolic addresses of the variable, the polling time, the transfer parameters (date, time of day, day of week) and the variables for setting the terminal clock from the controller end.

3.12.4 System Parameters: Running Time Meter

System parameters that can be specified for the running time meter are the addresses for the control byte and reset byte as well as the polling time.

3.12.5 System Parameters: Message System

System parameters that can be specified for the message system are the general parameters. These include the size of the message buffer and the message priority for direct display of a message.

Parameters that can additionally be specified for the serial message system are the parameters for the logging printer and the symbolic name of the variable which can be used to delete messages from within the controller.

Parameters that can be defined for the parallel message system are the size of the message buffer, the polling time and the symbolic name of the variable address.

Additional parameters for the serial and parallel message system are the sorting criteria and how the messages are to be displayed.

3.12.6 System Parameters: Variant Buffer

System parameters available for the variant buffer are the symbolic name of the variable and the size of the variant buffer.

3.12.7 System Parameters: Password Management

System parameters that can be defined for the password management are the passwords themselves, the corresponding authorization levels and the initialization values for the authorization levels.

3.12.8 System Parameters: Printer Interface

Parameters that are specified for the printer interface are the baud rate, parity, data bits, stop bits and the handshake.

3.12.9 System Parameters: Gateway

Setting of the gateway parameters applies only to operating terminals that have been fitted with the corresponding firmware.
Parameters that can be defined for these operating terminals are:
- Smallest possible slave number
- Largest possible slave number
- Polling time for text list
- Cache size
- Polling time for cache
- Variable for cache address
- Variable for network status address

3.12.10 System Parameters: Data Set Transfer

The following parameters are required to transfer data sets from the terminal to the controller:
- Variable for recipe number
- Variable for data set number

The following parameters are required to transfer data sets from the controller (on request):
- Variable for recipe number
- Variable for data set number

3.12.11 System Parameters: Parallel Outputs

System parameters for the parallel outputs are the symbolic name of the variable and the polling time of this variable. Every output can be enabled separately.

3.13 Version Number

The version number is reserved for the user. Valid values range from 0 to 255. The value is stored in a system variable. This system variable can be displayed by the user in any I/O mask.
This variable has no further function in the operating terminal. Online editing of the version number is not possible.

3.14 Running Time Meter

TesiMod operating terminals provide 8 running time meters to the user. In the programming software, enter a variable name for the control byte which constitutes the address at which the controller can influence the running time meters.
Every bit of this control byte represents one running time meter. If a bit is set to logical 1, the corresponding running time meter is incremented in accordance with the selected polling time cycle.

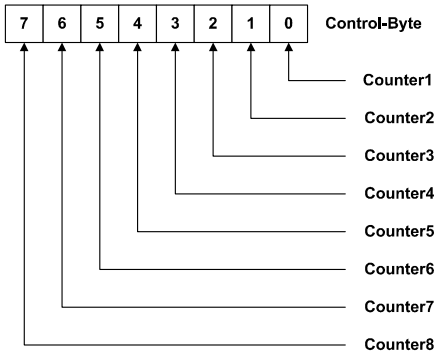


Fig. 29: Structure of the control byte

Example: A running time meter is to be set up for a maintenance interval of 50 hours.

Polling time for the counter: 60 seconds (the counter is increased by 1 every minute)

Setting: System variable **Counter1**
Fixed point number (TSdos)
Decimal number (TSwin)
4 digits; 1 post-decimal place (fractional digit)
Only positive
Factor 1; divisor 6, summand 0

The following is displayed on the terminal after 150 polling cycles:
Format: 150 : 6 = 25
Formatted display: 2.5 Hours

The precision in this example is +/- 6 minutes.

The variable "Reset Byte" can be used to reset each running time meter from the controller. To do this, the bit for the running time meter in question must be set to logical 1.

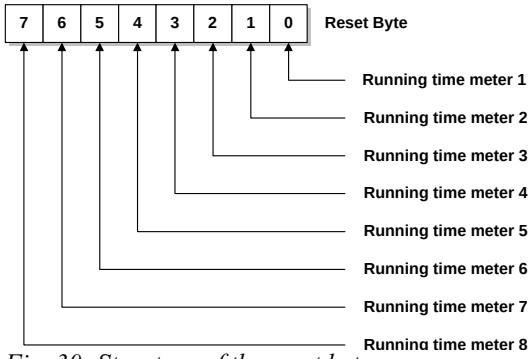


Fig. 30: Structure of the reset byte

3.15 Parallel Outputs

The parallel outputs of the operating terminal can be addressed directly with the function keys and by the controller. To be able to operate the parallel outputs from the controller, a variable must be defined for the control word. The word must be divided into two single bytes if the controller is only capable of accessing in byte mode. The byte order depends on the type of controller.

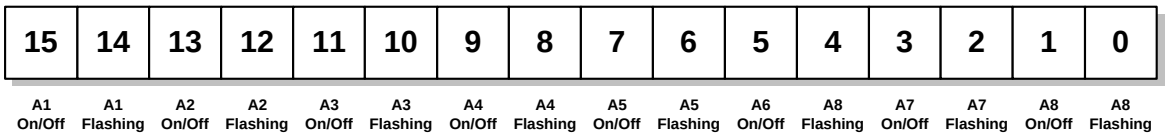


Fig. 31: Structure of the Control Word

The structure of the control word illustrates that one bit pair always controls one output. The following truth table applies to every bit pair:

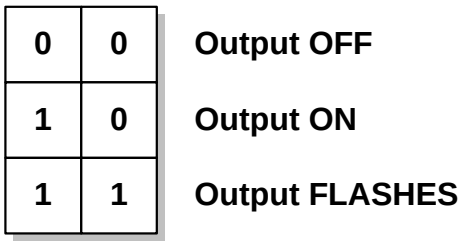


Fig. 32: Truth table for Parallel Outputs

In addition to the variable name, the polling time must be specified in the programming software to determine the cyclic intervals at which the control word is to be polled.
Outputs can also be addressed by means of function keys, provided this has been defined in the programming software. An output is being addressed (ON) as long as the function key is pressed. The programming software checks if the function keys and the controller want to access the same outputs.

3.16 Screen Saver

Some terminals are fitted with a screen saver. This function monitors all outputs to the display. If the system detects that nothing is being output to the display, a timeout begins to elapse. After the timeout has elapsed, the display is blanked and the status LED in the Help key begins to flash. The display can be reactivated by pressing any key.
Activation of the screen saver can be made dependent on the display of cyclic variables. In TSdos projects, the screen saver can not be activated at all if cyclic data are displayed.
The timeout can be defined in 0.1 second steps.
If the timeout is set to 0, the screen saver remains deactivated.

3.17 Image of the Mask Number

The mask number is a mask-specific code which the terminal writes to the controller variable entered here.
The terminal writes the mask number of the currently displayed mask into this variable whenever a new mask is activated. The variable is assigned in the variable list. In this list, a word address of the controller must be assigned to the symbolic name. The respective mask number can then be read from this address and be processed further as required. A 16-bit variable must be reserved for the mask number.
Also see the evaluation of the function keys and soft keys, etc.

3.18 Image of the Mode Selector Switch

In standard mode, the image of the mode selector switch (user mode switch) is transmitted to the controller after the initialization phase is complete. The user has the option of evaluating any unassigned DIP-switches in the controller. This allows the user to call up specific programs in the controller or to create queries in a service routine.

3.19 Terminal Clock

Every operating terminal is fitted with a real-time clock. The parameters of this real-time clock can be set in the system parameters.
Once per poll cycle, the real-time clock stores the current time, date and day of the week to the defined variable in the connected controller.
The connected controller itself can also transfer its current time, date and day of the week to the terminal.
This allows the time and clock in the controller, if any, to be synchronized or provides the real-time if controllers are not fitted with a clock.
Formats and contents of the variables must be identical with those in chapter "Image of Date and Time". To cause the terminal to read this variable, the controller needs to transfer the control code **7FF9H** through the serial message channel.

3.19.1 Image of Date and Time

The time of day, the date and the day of week are stored in BCD-format.
A maximum of 7 bytes of memory space is required. The address of the first byte is specified in the variable list. The data contents are defined as illustrated below:

Starting address +0 [Byte 0]	Y	Y	Year [0..99]
Starting address +1 [Byte 1]	M	M	Month [1..12]
Starting address +2 [Byte 2]	D	D	Day [1..31]
Starting address +3 [Byte 3]	h	h	Hour [0..23]
Starting address +4 [Byte 4]	m	m	Minute [0..59]
Starting address +5 [Byte 5]	s	s	Second [0..59]
Starting address +6 [Byte 6]	W	W	Weekday [0..6]

Fig. 33: Structure of the control byte for the time and date

The variable for the day of the week is calendar-independent and always runs modulo 6. The assignment of a number to a particular day of the week can be specified by the user when he sets the date and the text for the day.

Possible assignments in the text lists are:
0 Sunday
1 Monday
2 Tuesday
3 Wednesday
4 Thursday
5 Friday
6 Saturday

When entering the date, the user is then responsible for setting the variable for the day of the week to the correct value.
Example:
21.02.1997 (Friday) Day of the week in accordance with table above: 5

3.20 Read Coordination Byte

The term **Read Coordination Byte** indicates that the controller only reads this byte. It is only written to by the operating terminal.
This byte is used for the handshake and data coordination with the controller. For this purpose, the terminal reports its current status to the controller in the symbolic name entered here. Each bit has its own specific function. The structure is as illustrated below:

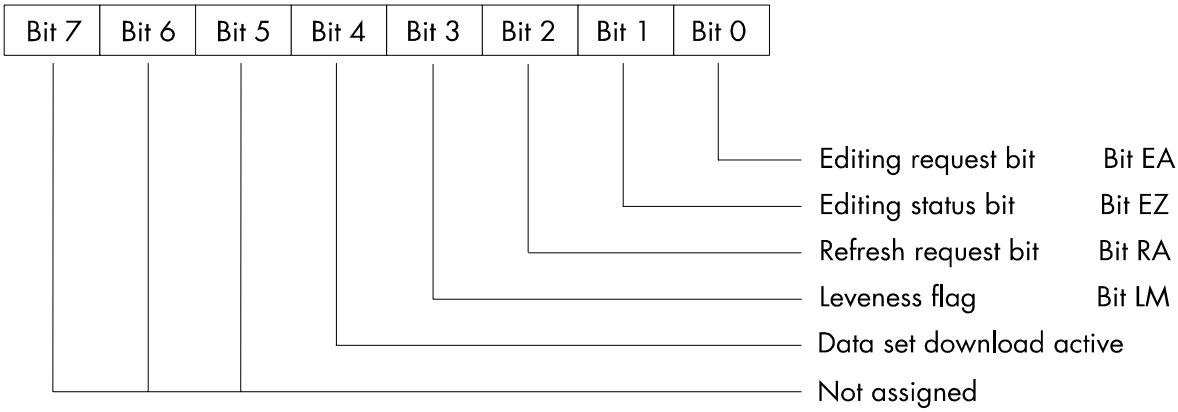


Fig. 34: Structure of the Read coordination byte

The Read coordination byte is used to pass the status of the operating terminal to the controller.

For the functionality of the individual bits refer to the respective chapters.
The terminal supports the <Read Coordination Byte> function only if the **Write Coordination Byte** has also been activated in the cyclic poll area. To activate both coordination bytes, the size of the poll area and the polling time must be specified during programming time. In addition, the addresses of the Read coordination byte and poll area must be specified in the variable list.

3.20.1 Editing Request Bit (Bit "EA")

The editing request bit is used to signal to the controller that the value of a variable is to be modified on the terminal. To do so, the operator presses the Data Release key. The status LED in the Data Release key is flashing as long as the editing release has not been set by the controller. Once the controller has set the **External Data Release** bit in the **Write Coordination Byte** to logical 1, the status LED in the Data Release key remains lit and the value can be modified by the operator.

3.20.2 Editing Status Bit (Bit "EZ")

Once the controller has set the editing release, the editing status bit is automatically set to logical 1 by the terminal. The bit is reset to logical 0 after the Enter key is pressed by the operator.

3.20.3 Refresh Request Bit (Bit "RA")

The refresh request function is set up by using an input variable with the PLC-Handshake attribute. With this function, a data release for the next input variable in the same mask will not be set until a refresh acknowledgement has been received from the controller.
Upon request, the controller can then refresh the next variable (bit "RA" = 1) before sending a refresh acknowledgement (bit "RQ" in the Write coordination byte = 1), thereby ensuring that the value the operator works with is current.

3.20.4 Liveness Flag (Bit "LM")

With some communication protocols it is not possible to check, from the controller end, whether or not the interface is in operating condition. This is why the liveness flag function has been introduced - a simple functionality that has proven to be very effective in practice.
Whenever the PLC wants to know whether a connection is still established, it writes a logical 1, and later a logical 0, to bit 3 of the **Write Coordination Byte**.
The operating terminal 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 a discrepancy occurs, the operating terminal copies the **Liveness Flag Bit** from the **Write Coordination Byte** to the **Read Coordination Byte**.

The controller is now responsible for checking within a timeout whether both states are in agreement. The transfer times and polling times must be taken into account when defining the timeout.

3.20.5 Data Set Download Active (Bit "DDA")

The **Data Set Download Active** bit remains set to logical 1 for as long as a data set is being transferred. The bit is reset after all of the data have been transferred. The controller can then work with the new values in the recipe.

3.21 Write Coordination Byte

The term **Write Coordination Byte** indicates that the controller writes to this byte.
The operating terminal only reads this byte.
Together with the **Read Coordination Byte**, this byte is used for the handshake and for data coordination with the controller. Here, the controller reports its current status to the terminal. The individual bits are independent of one another. The **Write Coordination Byte** is the first byte of the cyclic poll area.

The structure is as illustrated below:

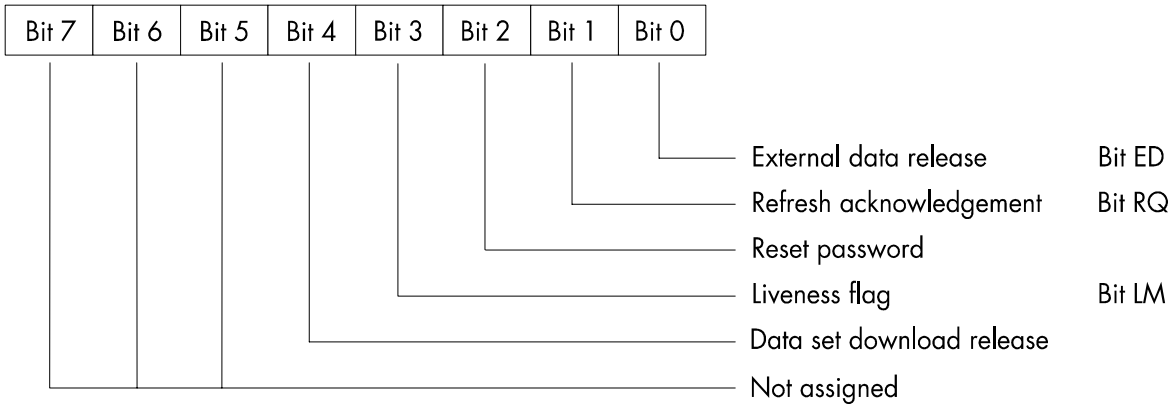


Fig. 35: Structure of the Write coordination byte

3.21.1 External Data Release (Bit "ED")

The external data release can be used to determine, from the controller end, the point of time from which editing of the value of a variable is allowed on the terminal. A flashing Data Release key LED on the operating terminal indicates that an external data release has not yet been set. Editing of the value is therefore still inhibited. Once the PLC has set the data release bit to logical 1, the Data Release LED remains lit and a new value can be entered by the operator. After all modifications have been made, the operating terminal sets the editing status bit back to logical 0, thereby signaling that the editing process has been completed.

3.21.2 Refresh Acknowledgement (Bit "RQ")

The refresh request bit is used to indicate to the controller that the values of the variables in the current mask are to be refreshed. After the values have successfully been transferred to the terminal, the controller sets the refresh acknowledgement bit to logical 1 to indicate that the process has been completed. See chapter 3.20.3 *Refresh Request Bit (Bit "RA"*).

3.21.3 Resetting the Password

The password protection should be reactivated after the operator exits a password-protected access level. This can be forced by the controller by setting the **Reset Password** bit to logical 1.

3.21.4 Liveness Flag (Bit "LM")

See chapter Liveness Flag of the **Read Coordination Byte** (3.21.4).

3.21.5 Data Set Download Release (Bit "DDF")

The Data Set Download Release (DDF) bit allows the controller to determine the point of time from which transfer of a data set to the controller can begin.

3.22 Cyclic Poll Area

In addition to a random read and write access to controller variables, a 23-byte memory area is defined in the user description as the cyclic poll area.

The cyclic poll area consists of:
- 1 byte **Write coordination byte**
- 2 bytes **serial message channel** (low and high byte)
- a terminal-specific number of **control bytes for** the **status LEDs** in the function keys

The poll area is written to by the controller and polled by the operating terminal at cyclic intervals. During this process, the controller can trigger, control, enable or inhibit actions that are connected with the terminal. In addition, the liveness flag of the Write coordination byte can be used to verify whether or not the connection with the terminal is still established. Actions are triggered and serial messages are passed to the operating terminal via the serial message channel (for example, transfer of data sets, change of masks). The bits of the control bytes can be used to activate or deactivate the status LEDs in the function keys or to place them to the flashing mode.

The marginal conditions regarding the memory area for the poll area are:
- the PLC accesses the Write coordination byte and status LEDs in bit mode and the serial message channel in byte mode or word mode
- the terminal accesses in byte or word mode
- the memory area must be contiguous.

The symbolic name for the cyclic poll area is specified in the parameters for the poll area using the programming system. The assignment of the starting address of this area is defined in the variable list.
Note that access to byte and word structures is not identical. Once an access method is selected, it should be applied throughout.

3.22.1 Byte-Oriented

If the byte-oriented assignment method has been selected in the variable list for the cyclic data area, the data area will be allocated as shown below:

Example: The cyclic poll area is set to flag byte MB 12 in the programming system.

Access in the PLC is via:

Byte Address	MB	Description
Byte address +0	MB12	Write coordination byte
Byte address +1	MB13	Message channel low-byte
Byte address +2	MB14	Message channel high-byte
Byte address +3	MB15	Status LEDs 1 to 4
Byte address +4	MB16	Status LEDs 5 to 8
Byte address +5	MB17	Status LEDs 9 to 12

Byte address +6	MB18	Status LEDs 13 to 16
Byte address +7	MB19	Status LEDs 17 to 20
Byte address +8	MB20	Status LEDs 21 to 24
Byte address +9	MB21	Status LEDs 25 to 28
Byte address +10	MB22	Status LEDs 29 to 32

The size of the data area is 11 bytes.

The structure of the byte-oriented cyclic poll area is as shown below:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte address + 0	not used	not used	not used	DDF	LM	PL	RQ	ED
Byte address + 1	serial message channel low byte							
Byte address- + 2	serial message channel high byte							
Byte address + 3	LED 1 on/off	LED 1 flashing	LED 2 on/off	LED 2 flashing	LED 3 on/off	LED 3 flashing	LED 4 on/off	LED 4 flashing
Byte address + 4	LED 5 on/off	LED 5 flashing	LED 6 on/off	LED 6 flashing	LED 7 on/off	LED 7 flashing	LED 8 on/off	LED 8 flashing
Byte address + 5	LED 9 on/off	LED 9 flashing	LED 10 on/off	LED 10 flashing	LED 11 on/off	LED 11 flashing	LED 12 on/off	LED 12 flashing
Byte address + 6	LED 13 on/off	LED 13 flashing	LED 14 on/off	LED 14 flashing	LED 15 on/off	LED 15 flashing	LED 16 on/off	LED 16 flashing
Byte address + 7	LED 17 on/off	LED 17 flashing	LED 18 on/off	LED 18 flashing	LED 19 on/off	LED 19 flashing	LED 20 on/off	LED 20 flashing
Byte address + 8	LED 21 on/off	LED 21 flashing	LED 22 on/off	LED 22 flashing	LED 23 on/off	LED 23 flashing	LED 24 on/off	LED 24 flashing
Byte address + 9	LED 25 on/off	LED 25 flashing	LED 26 on/off	LED 26 flashing	LED 27 on/off	LED 27 flashing	LED 28 on/off	LED 28 flashing
Byte address + 10	LED 29 on/off	LED 29 flashing	LED 30 on/off	LED 30 flashing	LED 31 on/off	LED 31 flashing	LED 32 on/off	LED 32 flashing
Byte address + 11	LED 33 on/off	LED 33 flashing	LED 34 on/off	LED 34 flashing	LED 35 on/off	LED 35 flashing	LED 36 on/off	LED 36 flashing
Byte address + 12	LED 37 on/off	LED 37 flashing	LED 38 on/off	LED 38 flashing	LED 39 on/off	LED 39 flashing	LED 40 on/off	LED 40 flashing
Byte address + 13	LED 41 on/off	LED 41 flashing	LED 42 on/off	LED 42 flashing	LED 43 on/off	LED 43 flashing	LED 44 on/off	LED 44 flashing
Byte address + 14	LED 45 on/off	LED 45 flashing	LED 46 on/off	LED 46 flashing	LED 47 on/off	LED 47 flashing	LED 48 on/off	LED 48 flashing

Fig. 36: Byte-oriented poll area

3.22.2 Word-Oriented

The structure of the word-oriented cyclic poll area is as shown below:

	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	not used	not used	not used	DDF	LM	PL	RQ	ED	not used	not used	not used	not used	not used	not used	not used	not used
Word address + 1	serial message channel low byte								serial message channel high byte							
Word address- + 2	LED 1 on/off	LED 1 flashing	LED 2 on/off	LED 2 flashing	LED 3 on/off	LED 3 flashing	LED 4 on/off	LED 4 flashing	LED 5 on/off	LED 5 flashing	LED 6 on/off	LED 6 flashing	LED 7 on/off	LED 7 flashing	LED 8 on/off	LED 8 flashing
Word address + 3	LED 9 on/off	LED 9 flashing	LED 10 on/off	LED 10 flashing	LED 11 on/off	LED 11 flashing	LED 12 on/off	LED 12 flashing	LED 13 on/off	LED 13 flashing	LED 14 on/off	LED 14 flashing	LED 15 on/off	LED 15 flashing	LED 16 on/off	LED 16 flashing
Word address + 4	LED 17 on/off	LED 17 flashing	LED 18 on/off	LED 18 flashing	LED 19 on/off	LED 19 flashing	LED 20 on/off	LED 20 flashing	LED 21 on/off	LED 21 flashing	LED 22 on/off	LED 22 flashing	LED 23 on/off	LED 23 flashing	LED 24 on/off	LED 24 flashing
Word address + 5	LED 25 on/off	LED 25 flashing	LED 26 on/off	LED 26 flashing	LED 27 on/off	LED 27 flashing	LED 28 on/off	LED 28 flashing	LED 29 on/off	LED 29 flashing	LED 30 on/off	LED 30 flashing	LED 31 on/off	LED 31 flashing	LED 32 on/off	LED 32 flashing
Word address + 6	LED 33 on/off	LED 33 flashing	LED 34 on/off	LED 34 flashing	LED 35 on/off	LED 35 flashing	LED 36 on/off	LED 36 flashing	LED 37 on/off	LED 37 flashing	LED 38 on/off	LED 38 flashing	LED 39 on/off	LED 39 flashing	LED 40 on/off	LED 40 flashing
Word address + 7	LED 41 on/off	LED 41 flashing	LED 42 on/off	LED 42 flashing	LED 43 on/off	LED 43 flashing	LED 44 on/off	LED 44 flashing	LED 45 on/off	LED 45 flashing	LED 46 on/off	LED 46 flashing	LED 47 on/off	LED 47 flashing	LED 48 on/off	LED 48 flashing

Fig. 37: Word-oriented poll area

Example: The cyclic poll area is set to DW 21 in the programming system:

Word Address	DW	High Byte	Low Byte
Word address +0	DW21	Write coordination byte	Reserved
Word address +1	DW22	Message channel high-byte	Message channel low-byte
Word address +2	DW23	Status LEDs 1 to 4	Status LEDS 5 to 8
Word address +3	DW24	Status LEDs 9 to 12	Status LEDS 13 to 16
Word address +4	DW25	Status LEDs 17 to 20	Status LEDS 21 to 24
Word address +5	DW26	Status LEDs 25 to 28	Status LEDS 29 to 32

3.22.3 Image of the LEDs

The image of the LEDs enables the PLC to control the status LEDs in the function keys. Functions that can be set for each status LED are the functions ON, OFF and FLASHING. Whenever the controller sets a bit, the associated LED on the operating terminal is influenced accordingly. It is important here that the size of the poll area and the polling time were set appropriately. Not defining these additional parameters may cause problems when the LED is addressed.

In the case of a function key that is to directly select a message mask, the status LED is influenced by the message system. The message system uses this status LED to indicate that a new message has been received but not yet acknowledged. To be able to influence the status LED in this

function key from the controller, the message priority must be set to 0 (zero). See chapter 3.11.5 *Status LEDs in the Function Keys*.

0	0	**Status LED OFF**
1	0	**Status LED ON**
1	1	**Status LED Flashes**

Fig. 38: Truth table for status LEDs

3.22.4 Serial Message Channel

The serial message channel is a part of the cyclic poll area and is used to transfer 16-bit information. The numbers of serial messages, selection of message masks, external selection of masks 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 terminal 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
- Mask numbers (mask number + 8000_H)
- Control codes

3.22.5 Polling Time

The polling time is the frequency with which the variables for the cyclic poll area are to be read by the operating terminal. This setting is specified in the system parameters for the poll area. Polling of this variable includes the **Write Coordination Byte**, the **Serial Message Channel** and the **Image of the status LED**. With most protocols, settings of around half a second have worked well. If the setting for the cycle time is too short, the interface protocol can no longer meet the requirements resulting in poorer reaction times. There is no general recommendation since the options depend on the user description involved. Time settings should, however, be at least greater than 100 ms. If you require additional assistance in this matter please contact our hotline.

3.22.6 Size of the Poll Area

The maximum size is 23 bytes depending on the data type and type of terminal. The poll area size can be adapted to the area that is actually used if the image of the LED or part of this function is not to be used. The default size for all operating terminals is 12 bytes.

3.23 Control Codes

Control codes have already been mentioned in the previous sections which can be used to trigger certain actions or functions in the operating terminal. All of these actions are initiated by the controller by writing the desired control code to the serial message channel within the poll area. An explanation of these control codes is provided below.

3.23.1 Triggering Data Set Printouts

The following hex code can be used from the controller to cause the current data set to be output to the printer connected to the operating terminal.

Hexadecimal code: **$7FF8_H$**

The operating terminal can indicate the status of the print process by writing either of two values to the address for the recipe number variable.

0 Data set printout ok.
255 Data set with the desired data set number can not be printed.

3.23.2 Setting the Clock in the Operating Terminal

The following hex code can be used from the controller to cause the real-time clock in the operating terminal to be set in accordance with the specifications in the defined control word.

Hexadecimal code: **$7FF9_H$**

3.23.3 Transferring Data Sets from the Controller to the Terminal

The following hex code can be used from the controller to initiate the transfer of a data set from the controller to the operating terminal. This requires that the recipe number and data set number are specified by the controller first.
The data sets are transferred as a block.

The following variables must be defined:
- Recipe number for the Request from Controller (System Parameters: Data Set Transfer)
- Data set number for the Request from Controller (System Parameters: Data Set Transfer)

Hexadecimal code: **$7FFA_H$**

3.23.4 Transferring Data Sets from the Terminal to the Controller

The following hex code can be used from the controller to initiate the transfer of a data set from the operating terminal to the controller. This requires that the recipe number and data set number are specified by the controller first.

The following variables must be defined:
- Recipe number for the Request from Controller (System Parameters: Data Set Transfer)
- Data set number for the Request from Controller (System Parameters: Data Set Transfer)

Hexadecimal code: **$7FFB_H$**

3.23.5 Transferring Data Sets from the Controller to the Terminal (Individually)

The following hex code can be used from the controller to initiate the transfer of a data set from the controller to the operating terminal. This requires that the recipe number and data set number are specified by the controller first.
The data sets are transferred individually.

The following variables must be defined:
- Recipe number for the Transfer from the terminal (System Parameters: Data Set Transfer)
- Data set number for the Transfer from the terminal (System Parameters: Data Set Transfer)

Hexadecimal code: **$7FFD_H$**

3.23.6 Refreshing the Message System

The following hex code can be used from the controller to cause the operating terminal to read in the most recent parallel messages.
This procedure can be used to achieve an event-controlled message system.

Hexadecimal code: **$7FFF_H$**

3.24 Cyclic Variables

The term Cyclic Variables refers to data which may change continuously while a mask is displayed, i.e. all types of ACTUAL VALUES. The terminal must therefore poll the controller for the current values at cyclic intervals. The time selected here specifies the intervals at which these mask values are refreshed.
In addition to the cyclic variables, the terminal also polls the controller for the current status of the Write coordination byte, the serial message channel and the LEDs (update). This process is either time or event controlled depending on the protocol. No polling will take place if the value = 0 is entered for the time period. If this variable is not polled, then an external control of the data release is not possible, for example.

3.25 Interface Parameters X2, X3

The interface parameters selected in the programming system are stored in the mask memory. It is, however, possible to modify the data in the operating terminal at a later date. Values modified in the operating terminal are retained in the battery-backed RAM. When necessary, the original settings can be restored at any time.

The parameters of the interface at connector X2 are determined by the protocol. Modifications to any of these parameters could result in communication failures. Values that are not typical are detected when the terminal is initialized and a message to this effect is generated.
The parameters of the interface X3 are reserved for control of a printer and, when transferring hard copies, of a PC (hard copy function is not possible with operating terminals with 386 processors). This process requires that the settings for both devices must match. The selected parameters do not apply to the download. For the download, the system will automatically switch to the highest possible transmission rate.

3.26 Variable Definition

The variable definition contained in the user description defines the format, type, range of values, scaling and attribute of the representation. All controller-independent parameters are stored here. For editable variables, it is additionally possible to influence the Editor and the transfer mode to the controller. The variable definition contains the symbolic name of the variables, the format definition and, if necessary, the text lists in the case of selection text variables (coded text variables).

3.26.1 Variable Formats (TSdos)

The formats, once defined in a variable definition, are stored in the programming system with a format name. This allows different variables to be easily output with the same format without having to reenter all of the parameters each time.

3.26.2 Variable List

The variable list contains the assignment of symbolic variable names to the destination hardware. Due to its design and proximity to the PLC, this file is always manufacturer-specific. In TSdos, this list contains the variable addressing and protocol-specific parameters, interface parameters, timeouts, etc. The file extension for the variable list in TSdos is .**TSV**.
In TSwin, the variable list is a part of the database. Any number of variable lists can be created. The variable list contains only the required manufacturer-specific user description parts. Its size is therefore small in relation to the entire project. A project can be quickly adapted to other PLC specifications by simply adapting the variable list to the PLC to be used.

A variable list can be created in the programming system in two ways.
First method:
TSdos:
First, the entire operator guidance is entered, upon which the programming system will, when prompted to do so, create a protocol-specific variable list that will contain all of the variable names that have been used. After the list has been created, it will also be used as a name list when the masks are programmed. The list will not yet contain information on the controller's hardware references. This information will be supplied by the user when he enters this information into the variable list in the manufacturer-specific notation. This method ensures that no variables from the variable list are omitted.

TSwin:
As a variable is defined in a mask (with or without address information) it is entered into the variable list at the same time.

Second Method:
TSdos:
The user begins with generating the variable list with the symbolic names and all of the references on the basis of the PLC program. The same names can be used here as are used in the PLC. When creating masks, the programmer can simply refer to the name list and use the respective variable names.
TSwin:
The first step can be to define the entries in the variable list or alternatively to insert them from any suitable program via the clipboard. The entries will immediately be available in TSwin on a global basis.

A combination of the two methods can also be used.
TSdos users who have the corresponding documentation or databases on their PLC programs at their disposal can alternatively generate the variable list from this PLC documentation (print file) themselves. The structure of the list (TSV-file) corresponds to a simple text file. The demos contain the definition in the form of comments. Note that the definition format varies with the protocol used.

Example for a Siemens PLC with programming unit interfacing:

```
Mit "Co" beginnen alle Kommentarzeilen.
Auszug aus einer Variablenliste:
Co Für Siemens PG Schnittstelle
Co Vp Siemens PG Variable
Co Vp \Variablenname\,\Datentyp\,\Parameter 1\,\Parameter 2\
Co oder für Siemens L1 Schnittstelle
Co Vl Siemens L1 Variable
Co Vl \Variablenname\,\Datentyp\,\Parameter 1\,\Parameter 2\,\Slave\Co

Co Datentyp      : E,A,M,EB,EW,AB,AW,MB,MW,DW,DL,DR,t,z
Co Parameter 1   : Bei Datentyp DW, DL, DR Bausteinnummer, sonst
Co                 Byte oder Wortoffset
Co Parameter 2   : Bei Datentyp E,A,M Bitnummer 0 bis .7
Co                 Bei Datentyp DW,DR,DL Datenwortnummer
Co                 sonst keine Bedeutung
Co Slave Adresse : Nur bei L1 Protokoll

Vp \Maskennummer\,\MW\,\76\,\0\
Vp \ZyklischeDaten\,\MB\,\0\,\0\
Vp \Dipschalter\,\MB\,\120\,\0\
```

3.27 Application Programming

The TesiMod operating concept is based on a mask structure that can be created by the user in accordance with the requirements and on system variables that can influence important functions in the operating terminal. Various types of masks are available to facilitate the creation of solutions for standard tasks.
Every mask structure initially consists of 4 types of masks whose contents are user-configurable.

In TSwin, any I/O mask can be programmed as a system mask. In TSdos, the following are fixed masks:
- Setup mask (1)
- Startup mask (2)
- Password mask (3)
- Main mask (4)

After being switched on, the operating terminal is initialized and displays the startup mask during this process. The setup mask can be called up by pressing the Enter key while the startup mask is displayed. The setup mask can be exited by pressing the Enter key again and the startup mask will reappear.
The next mask of the operating guidance that is displayed is the Main Mask (mask 4) which represents the first mask of the user-programmed operator guidance. A menu with menu items (node mask) is normally set up in the main mask to allow selection of lower-level masks. The user has the option of specifying a full-page help text for every mask which can be displayed by the operator with the Help key.
The functions of the system variables are briefly described in chapter 3.6.3. Some of these variables are mentioned in the following chapters to be able to explain the application programming.

3.27.1 Configuring the System

To run the programming software, you will need a PC that complies with the requirements described in the software manual. One of the computer's serial interfaces (COM1 or COM2) is connected with the interface X3 of the operating terminal using the download cable (available as an accessory). The connection of the operating terminal to the PLC (using interface X2) is established with an additional, controller-specific interface cable. Care must be taken to ensure that the correct type of interface cable is used and that the operating terminal is connected to a power source which complies with the technical specifications for the operating terminal.

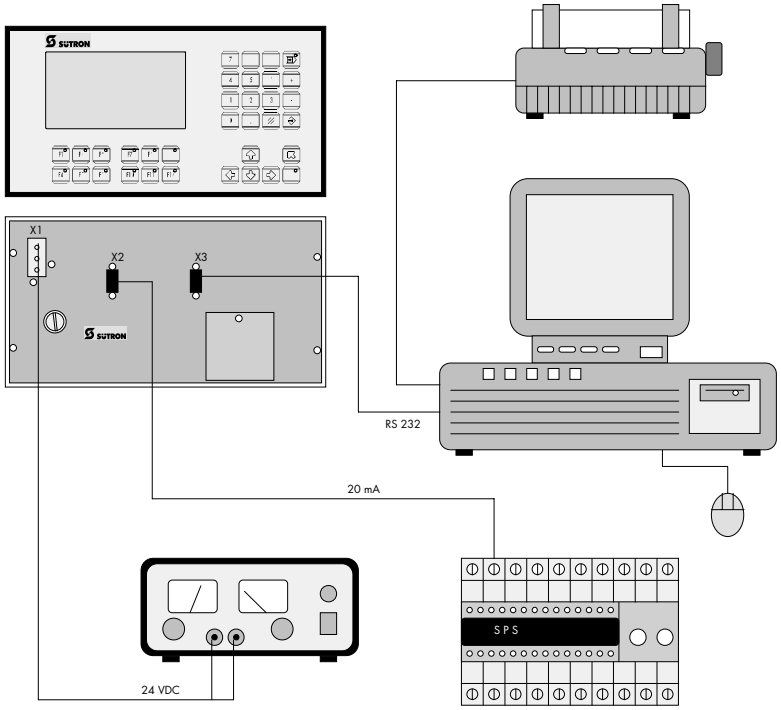


Fig. 39: System configuration

Project execution can begin once all preparatory measures have been taken. All functions provided by the operating terminal can be executed and tested with the configuration shown above. If operated without a PLC (see simulation without PLC), all operating functions with the exception of the PLC responses can be tested.

3.27.2 TSdos and TSwin Programming Systems

The programming systems TSdos and TSwin provide all of the capabilities required to conveniently and quickly program the entire range of functions offered by an TesiMod operating terminal. The programming system TSdos is an MSDOS based PC program. The program corresponds to a SAA-standardized operator interface in the text mode which can optionally be operated with hot keys, Alt-key combinations or with the mouse.
TSwin on the other hand requires either Windows95 or Windows NT 4.0. It offers Windows-conforming operability in conjunction with all of the means provided by Windows (OLE, COPY/CUT/PASTE, WYSIWYG). TSwin provides a help system that can be reached with the Help key or the function key F1 from anywhere within the program (online help).

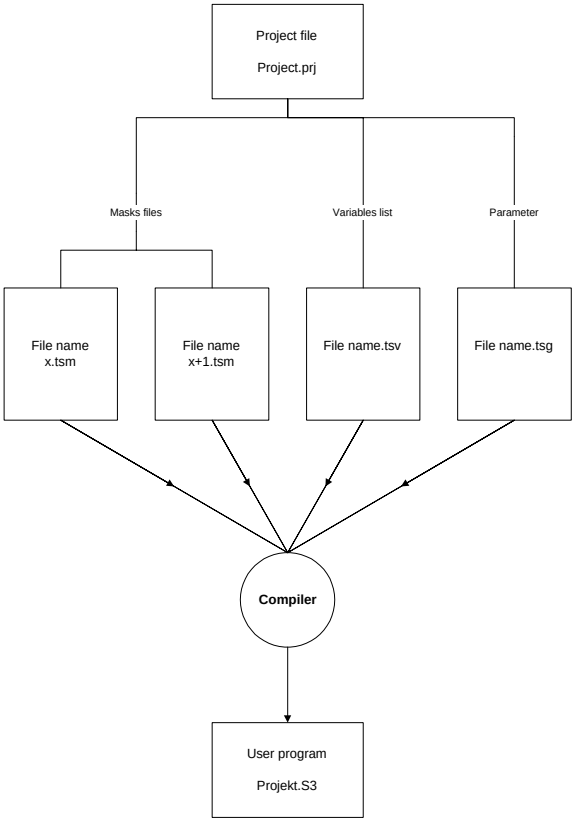


Fig. 40: Project files of the TSdos programming system

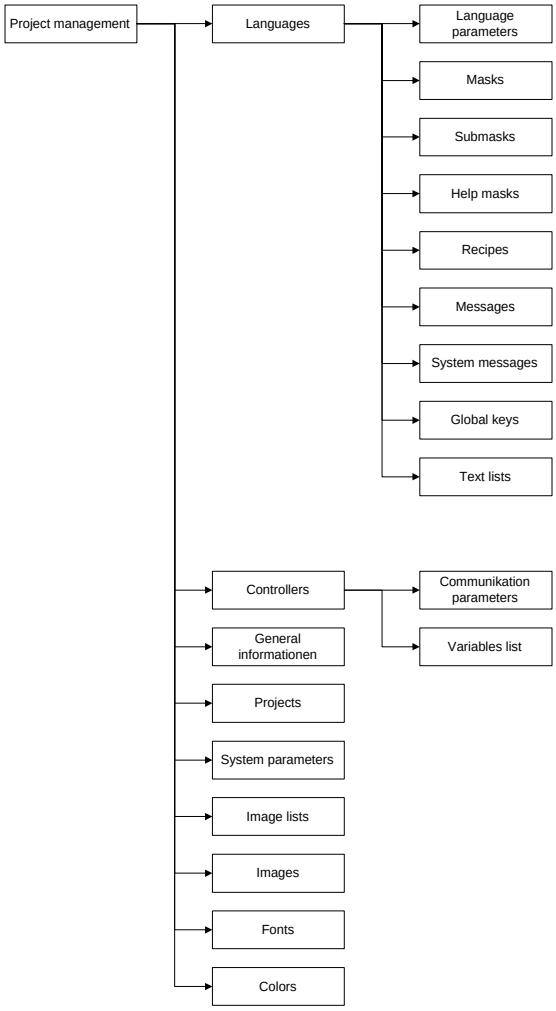


Fig. 41: Structure of the TSwin programming system

Information on how the TSdos programming system works and how it is operated is available in a separate, extensive manual. An online help system that can be called up with the F1 key provides additional support on how to operate the software. The programming systems provide all of the functions required to perform the steps necessary to create a project. TSdos can be used to create various mask, parameter and variable files and to organize them into a project.

The TSwin programming system operates similarly to a database. All project-related data are collected in this database and are organized into a project.

This project can then be compiled into a S3 file and transferred to the terminal using the download function.

3.27.3 Getting Started with Programming

Demo projects are available to provide the user with an easy way of getting started with project creation. The demo projects are complete applications that illustrate all important programming options.
Contact out sales department if an application for your type of operating terminal is not available.

3.27.3.1 Project Description

The project description consists of:

at least the user description (one per language)
a variable list (controller-specific)
the global data (parameters)
and, if used, parts of the graphical library.

The user description contains all of the controller-independent information. Symbolic names are inserted for all controller-specific variables.
The assignment of these symbolic names to the hardware is specified in the variable list. This makes it possible to easily change to another controller manufacturer without having to recreate the masks, variable definition or operating functionality. The adaptation to a different controller type can be accomplished by simply replacing the variable list.

3.27.3.2 Multilingual Projects

The mask contents of a user description determines the language in which the texts are displayed. To make an operator interface available in multiple languages, the user description of each language is stored with a language number. The desired language can then be selected online via a system variable using the assigned language number.
The number of languages that can be stored is limited only by the size of the mask memory. Please note that it is not necessary that the length of the texts, the variables and the number of masks are identical in the various languages.

The number of the currently active language is stored in the operating terminal and is retained when the terminal is switched off. When booted again, the terminal defaults to the language which was active when the terminal was turned off. In the case of a loss of data, the language that corresponds to the number 0 is automatically displayed.

The language can easily be selected on the operating terminal using a selection text. The language can then be selected with the Cursor up and Cursor down keys.

Overview of the required steps:
1. Create a text list with the language name (Example: Deutsch, English, Française)
2. Create the mask
3. Create the variable in the mask:
   OSLanguage, representation method, selection text, text list name as created in step 1.

Attention: Do not enter system variables in the variable list. This would result in the controller addresses being accessed instead of the internal operating terminal addresses.

3.27.3.3 Variants of a Project

The system variable **OsLanguage** also permits the management of variants. For example, if you design a machine with various options but do not want the operator of the machine to be distracted by these options, it is possible to select the appropriate variant online on the terminal.
Example:

VALUE	FUNCTION
0	Variant 1
1	Variant 2
2	Variant 3

The variants may differ in operating structure, the number of masks as well as the variables used. The advantage is that the user description must be maintained only once for the entire system.
Example:
A possible application is when a system is to be operated with the metric and Anglo-Saxon dimensions. In this case, the format and scaling of the variable inputs and outputs could be switched from millimeter to inches along with the language.

3.27.4 Project Documentation

The programming software also allows the generation of extensive documentation on the project. Detailed information on how to use variables, the scaling parameters, representation methods, mask contents as well as the operator guidance facilitate verification of the user description.
The documentation should cover any information necessary to recreate the user description in the event of a loss of data.

3.27.4.1 TSdos Print Files

To obtain documentation on a project, print files (lists) with the following contents can be created in TSdos:
- User description
- Documentation of individual masks
- Parameter list
- Variable list
- MAP list

The lists are generated in either the standard ASCII format or extended ASCII format (IBM mode) and can then be output to a printer using the commands provided by the operating system.

3.27.4.2 Hard Copies of the Masks in TSdos

The primary purpose of hard copies of the display is to provide the user with an easy way to create documentation. The output of hard copies when using graphics displays has therefore been designed in such a way that they can be integrated into a documentation. The upload to the PC is carried out via the X3 interface.
The file should then read in by a receiving program such as for example "Terminal" which is available under Windows 3.1x (1). The file will be in PCX format which can be imported into and be processed by various graphics and desktop publishing programs.
With the aid of the X3 interface it is possible to obtain a hard copy of each mask in the terminal. During such a process, graphics displays will transfer a PCX format while alphanumerical displays will transfer an ASCII format to the PC.
The hard copy is activated with the system variable **HardCopy**. During the programming phase you may want to assign a function key to activate this system variable. This will allow the selected masks to be transferred to the PC by simply pressing the function key.
The parameters defined for the X3 interface are valid and are not modified by the hard copy function.

3.27.4.3 Creating TSwin Documentation

TSwin offers extensive dialogs that allow those elements to be selected that are to be documented.
The TSwin documentation can then be easily processed further using word processing programs or desktop publishing programs.

3.27.5 Project Back-up

A project is backed up by storing the source files onto various data mediums, as recommended by the PC manufacturers. For information on the available options refer to your PC's operator system manual.

In addition, we recommend that the projects are printed using the documentation functions.

3.27.6 Optimizing the Transmission Rate

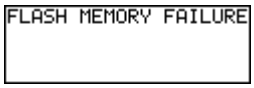
The TesiMod operating concept supports free access to various data types and data lengths. Although, the extent to which the various protocols actually support this free access varies greatly. The firmware of the terminal optimizes the access method mask-specifically to achieve extremely short access times.
This method means that an attempt is made to transmit several variables to the display simultaneously. The user can actively contribute to this process by keeping the number of different variable types used in one mask to a minimum and by addressing variables of one type contiguously (small gaps are no problem!). This allows shorter refresh intervals and reduces the load on the interface.

3.28 Downloading the User Description

The download function describes how a new project is loaded into the terminal's Flash memory. Before a new project can be downloaded, the terminal must be placed into the download mode. The download mode is activated by:
- Writing a "1" to the system variable **IntEraseEPROM** in an I/O mask.
- Switching off the supply voltage, setting the DIP switch S4 to ON, turning the device back on and once the following message is displayed

switching the DIP switch S4 to OFF again with the power on.

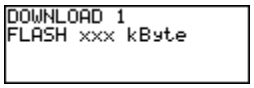
If the terminal is fitted with a normal UV-erasable Eprom instead of the Flash memory, this will be detected and a deletion or programming operation will be prevented.
The following error message will be displayed:



to indicate that the download process has not been completed successfully.

While programming with TSwin, you may want to activate the automatic download function. This will automatically switch the operating terminal into the download mode whenever a download is started on the PC.

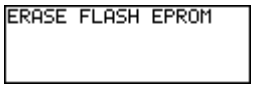
When the download function is activated, the user description currently stored in the terminal will be erased and the following information will appear on the display at the cursor position (1.1):



A new user description can now be transferred to the terminal using the Project Management dialog in the programming system. For this process, the PC must be connected to interface X3 of the terminal using the download cable.
To prevent the user description from accidentally being erased during operation, it is possible to assign an appropriately high access level to the mask with the download function, so a password is required to access the mask. If the mask has been assigned to a control key, the operator will not even know that such a mask exists.
Another method would be to prevent input by using the connected controller. The password protection function is not required in this case. With this method, the controller will provide data release for the mask number with the download function only, if specific conditions set out by the user are fulfilled.

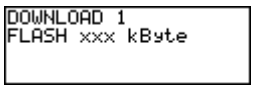
If the user does not want to make use of these options, another alternative to the software solution via the system variable **IntEraseEPROM** would be to erase the old variable definition with the aid of the DIP switch S4. This will require strict adherence to the routine to be executed. Subsequently the terminal will automatically activate the download operating mode.

To place the operating terminal to the download operating mode:
The Flash-Eprom is erased once the system variable has been activated. The following message is displayed during this process:



After the Flash-Eprom has been erased, this is indicated by the following message:

The terminal now automatically activates the download mode and the following message appears:



The terminal is now ready to receive a new user description via interface X3 and to store it in the Flash-Eprom. The progress of the data transfer will be indicated on the display by the characters ">>>>>", the number of characters will therefore change continuously!
After the transfer is complete, the following message is displayed:

After the transfer, the terminal immediately starts the initialization phase and displays the startup mask of the user description.
Various error messages may appear on the display during the download or at the beginning of the initialization phase:

ADDRESS ERROR	Error in the address calculations of the compiler
FLASH MEMORY FAILURE	The loaded user description contains errors or is incomplete.
CHECKSUM ERROR	The user description contains errors, please recompile.
BYTECOUNT OVERFLOW	The user description contains errors, please recompile.
FORMAT ERROR (S0, S3, S7)	Sequences in the S3 file that was transferred contains errors.

The display of an error message indicates that the transferred user description is incomplete or contains errors. Switching the terminal off and on at this point will automatically reactivate the download operating mode.

3.28.1 Downloading with Windows

The programming software does not necessarily have to be used for downloading. Another option is to use the program Terminal that is available under MS Windows (group "Accessories"). The following steps are required:

1. Connect COM1 or COM2 of the PC with BTxx-X3 using the download cable
2. Double-click the Terminal icon under Windows
3. Set the parameters:

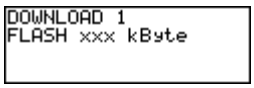
Menu:
Settings:
Communication: 19200Bd
7 bits
odd parity
xon/xoff
1 stop bit
Transfers:
Send text file:
(Enter file name) xxxxx.S3

3.28.2 Application Memory

A Flash memory or UV-Eprom can be used as an application memory. The terminal comes fitted with a 128 kByte Flash memory. This Flash memory is installed in a 32-pin DIL-precision socket and can be replaced with the aid of an extraction tool. A standard UV-Eprom of appropriate size and access time can be used instead.
The S3 file generated by the programming system can be processed by most Eprom programming units. UV-Eproms can neither be erased nor programmed in the terminal. Only Flash memories are designed to be programmed in the terminal.
Flash memories can also be programmed with the aid of modern Eprom programming units, this may eliminate the need for a download on site during servicing works.

3.28.3 Loading a User Description

To load the user description, a PC must be connected to interface X3 using the download cable. The terminal must then be placed to the download mode. The following message must appear on the display:



The terminal is now ready to receive a new user description via interface X3 and store it in the Flash memory.
Now the download of an error-free user description "name S.3" must be activated using the Project Management dialog in the programming software.
The progress of the data transfer will be indicated on the display by the characters ">>>>>", the number of characters will therefore change continuously.

The character "*" will be displayed in the programming system for every transmitted block. After the transmission is complete, the following is displayed

```
READY
```

to indicate that the transfer has completed. After the transfer, the terminal immediately starts the initialization phase and displays the startup mask of the user description.

3.28.4 Activating the Download Function using the Software

This requires that the terminal is placed to the download mode by activating an I/O mask and writing a "1" to the system variable **IntEraseEprom**.
In the event that you have decided not to use the system variable **IntEraseEprom**, the old variable definition must be erased with the DIP switch S4. This will require strict adherence to the routine to be executed. Subsequently the terminal will automatically activate the download operating mode.

3.28.5 Activating the Download Function using the Hardware

If the system variable responsible for erasing the Flash memory can not be activated in the masks or is not available at all, the Flash memory can still be erased by switching the mode selector switch to a defined position. This will cause the terminal to activate the download mode.
Switch position to completely erase the Flash memory:

S1	**ON**	S5	not used
S2	not used	S6	not used
S3	OFF	S7	not used
S4	**ON**	S8	not used

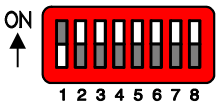


Fig. 42: Mode selector switch

The following message is displayed once power has been applied the terminal:

```
1. TURN POWER OFF
2. RESET DIP-SW 4
OTHERWISE ALL FLASH-
DATA WILL BE LOST !!
```

This message is designed to prevent accidental erasure of the user description. The user description will be preserved if this message is complied with.
Thus, turn off the power, reset the mode selector switch S4 and switch the terminal on again. The terminal will continue to work with the current user description.
Note that the contents of the Flash memory will be erased if the power is not turned off before the DIP switch S4 is reset.

3.28.6 Automatic Download Function

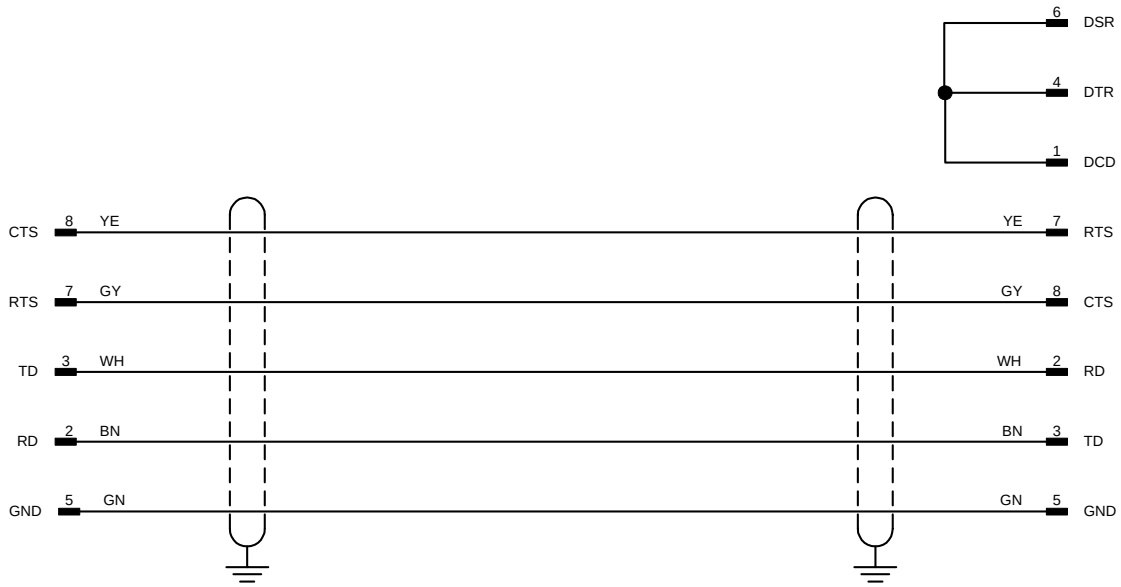
The Automatic Download option can only be selected when TSwin is used for programming. To enable this function, the appropriate check box must be selected under "System Parameters", "General Parameters".
The user description that contains this function must initially be loaded into the terminal via a forced download (DIP switch S4 at the ON position). Every download thereafter is automatically initiated by the PC wanting to load the S3 file of the new user description into the terminal. A download can be forced when the operating terminal is in full operation. During this process, all
- message data in the message memory
- all RAM-data sets that have not been backed-up
- all system parameters that have been edited online (values of the system variable, interfaces, passwords, etc.) will be lost!

3.28.7 Download Cable

Pinning of the cable connecting interface X3 of the operating terminal to the PC (except BT2, BT5N and BT20N).

TesiMod
Operating Terminal

Personal
Computer



D-Subminiature
Male Connector
9-pin

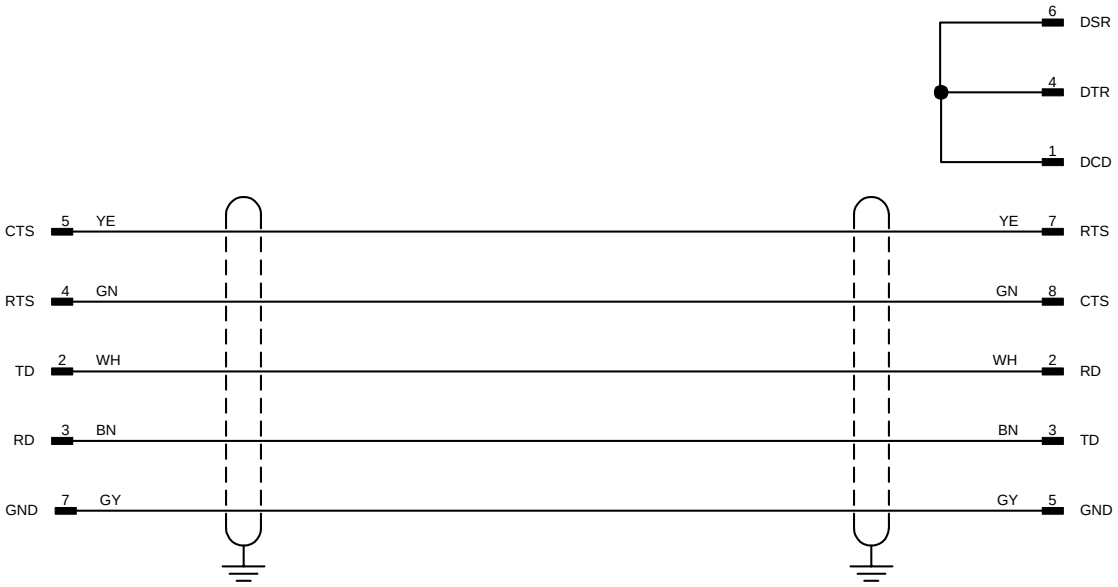
D-Subminiature
Female Connector
9-pin

The shield is connected to the metal casing on both ends.

Pinning of the cable connecting interface X3-SER2 of the operating terminal to the PC (BT2, BT5N and BT20N only).

TesiMod
Operating Terminal BT2, BT5N, BT20N

Personal
Computer



D-Subminiature
Male Connector
25-pin

D-Subminiature
Female Connector
9-pin

The shield is connected to the metal casing on both ends.

3.29 Simulation without the Controller

The terminal can be operated without a controller by setting the DIP switch S3 to ON. The DIP switch S3 will automatically set the external data release. It is possible to simulate the entire operator guidance including help texts. The variables are automatically assigned to a common variable address in the terminal. This allows the Editors and their limits to be tested. The values are, however, not retained. There is no communication with the controller.

Table of Contents

4 TesiMod Transparent Mode

In the transparent mode the terminal works like a text display with printer port. The PLC has to provide the terminal with all the symbols and control sequences which used to display the data and values. Only important errors are displayed with a message (system message). All PLC communication parameters are adjustable in the setup mask.

4.1 Setting the Operating Mode

The user-mode switch must be used to select the mode of operation "transparent mode". Turn off the terminal before selecting the mode of operation.
The position of the mode selector switch is described in the manual for the particular operating terminal.
All terminals are factory-set to the standard mode.
The ON/OFF positions on the mode selector switch are marked.

Position of the switch for transparent mode:

S1	OFF	S5	not used
S2	not used	S6	not used
S3	OFF	S7	not used
S4	OFF	S8	not used

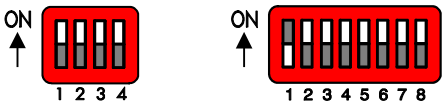


Fig. 27: User-mode switch

The selected mode of operation is activated once power is applied to the terminal.

4.2 Start-up Processes

The start-up processes are identical for all TesiMod operating terminals.
After the voltage has been applied, the following message will be displayed:

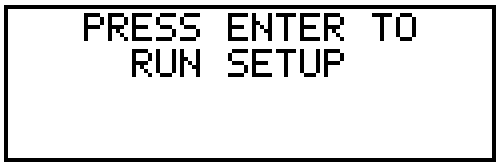


Fig. 28: Start-up mask example

The message will be displayed in an one-line, centred format or in case of smaller-sized displays in a two-line format. All LEDs will be activated while this message is being displayed, allowing the operator time to perform a visual inspection. The message will be displayed for 2 sec. After this time period has elapsed, the message will be erased, the cursor will be positioned at the top left of the display (position 1, and the characters received at interface X2 will be interpreted and displayed.

Pressing the enter key immediately after the terminal has been switched on, while the prompt is being displayed, will activate the setup mask, containing the terminal-specific parameters, on the display. All editable parameters will be displayed within one screen. Editing of the interface parameters is possible after the data release key has been pressed, the integrated LED will be activated. The input via coded text (scroll bar) prevents incorrect input. The data input is confirmed by pressing the data release key once more. The LED in the key will be deactivated. Pressing of the enter key will reboot the terminal with the current values.

Pressing the enter key before the prompt "Press Enter to run Setup" is displayed, will result in the generation of the following error message during the keyboard test.

Fig. 29: keyboard error mask

The self-test performed on the keyboard after having switched on the terminal has detected that a key is being pressed. Comply with the request and release the key. If the message is generated without any key being pressed, this indicates a defective keyboard!

4.3 Communication in the Transparent Mode

In transparent mode, the communication is carried out via interface SER1 exclusively. This is where the controller or the higher-level computer must be connected, taking into consideration the hardware configuration. Adaptation of the transmission parameters to the requirements must be carried out in the setup mask.

4.3.1 Interface Parameters

Unless otherwise specified, the following interface default parameters are used:

Interface	SER1	SER2
Baud rate	19200Bd	9600Bd
Data bits	8-bit	7-bit
Stop bits	1-bit	1-bit
Parity	none	odd
Handshake	Xon/Xoff	Xon/Xoff

The default parameters are stored in the Flash memory and are therefore not lost.

4.3.2 Receive Buffer for the Communication Interface

The receive buffer of the communication interface comprises a size of 256 bytes.

4.4 Function Setup Menu

The setup menu allows selection of the following parameters:
- Parameters communication SER1
- Parameters communication SER2
- Date
- Time
- Default contrast

In addition, the version number of the operating system is displayed in the header line.

4.4.1 Adaptation of the Parameters in the Setup Menu

Within the setup mask, the parameters can be modified by means of an editor. Pressing of the data release key allows changing into the editor. Modifying of the data is possible once the LED of the data release key has been activated. Modifying of the variable values is possible through the + key and - key or the numerical keys. The input must be confirmed by pressing the enter key. If supported by the display, the modified data will be displayed in an inverse format. To prevent incorrect or unintentional modifications, the input must be carried out in compliance with a predefined sequence which must be strictly adhered to.

4.4.2 Example of a Properly Performed Modification!

1. After having switched on the device, the enter key must be pressed immediately once the prompt is being displayed. The setup mask will appear on the display. This procedure must be carried out even if the parameters are to be viewed only.

2. It is possible to exit the menu by means of the enter key without modifying the menu. Upon pressing the data release key, the LED integrated in the key will be activated, thus signalling the editing mode within the transparent mode of operation.

3. The cursor will be positioned on the first editable variable and will be flashing. The cursor keys allow selection of the variable to be modified.

4. Numerical values may now be entered through the numerical keypad, texts are selected using the +,- keys.

5. The enter key must be pressed following each input. After a valid input, the cursor will appear on the next editable input field.

6. In case of interface parameters, the variable to be modified must be set to "SETUP DATA", thereby preventing an accidental input. The selection "SETUP DATA" must be made separately for the communication interface SER1 and the download-/printer interface SER2.

7. Exiting of the editing mode is possible by pressing the data release key once more, the LED will be deactivated; any modified values will still be displayed in an inverse format. Repeated corrections are possible using the data release key.

8. The enter key allows exiting of the setup menu. The terminal will be rebooted and the selected parameters will become effective. The entire process may be repeated, if necessary, it will not be necessary to switch the terminal off.

The data are stored in a battery-backed RAM memory. The battery is monitored such that a loss of data is prevented and a message will be displayed whenever the capacity of the battery is below the required level.

4.4.3 Setup Menu

The setup masks of the BT2 and BT5N contain exactly the same information. This information is packed, thus permitting all parameters to be included into one mask.

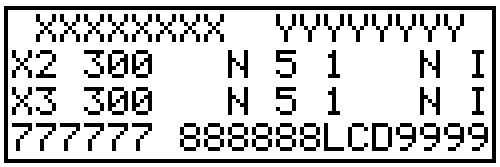


Fig. 30: setup mask

1st line	**XXXXXXXX**	Program Version	
	YYYYYYYY	protocol driver	
2nd line	X2	Communication SER1	
	300	Baud rate	(19200)
			(9600)
			(4800)
			(2400)
			(1200)
			(600)
			(300)
	N	Parity	(O)dd
			(E)ven
			(N)one
	5	Data bits	(8)bit
			(7)bit
			(6)bit
			(5)bit
	1	Stop bits	(1)bit
			(1.5)bit
			(2)bit
	N	Handshake	(N)one
			(R)TS/CTS
			(X)on/Xoff
	I	Acceptance	(I)nactive
			(S)etup Data
			(D)efault Values
3rd line	X3	Communication SER2	
	(Structure same as line 2)		

4th line	777777	Date	010100 to 311299
	888888	Time	000000 to 235959
	9999	Contrast	-125 to +125

4.5 Display

The first cursor position on the display is on the top left (X=1, Y=1). The number of permissible rows and columns varies with the terminal type.

	Lines:	Columns:
BT2 normal	4	20
BT5N normal	4	20
BT20N normal	16	40
BT20N zoom	8	20

4.5.1 Character Set, Character Attributes

The displayable character set is listed in the hardware reference description of the respective terminal. Please note that it is <u>not</u> possible to address the extended ASCII character set in the range of $128_{D}..255_{D,}$ when selecting a character length of 7 bits.

4.6 Keys

A start and stop code is transmitted to the controller for each key that is being pressed. Please consult the table of the terminal for the respective code being generated. The transmission of the characters is effected in compliance with the conditions of the selected interface parameters.

4.6.1 Key Codes for the Terminals

	Key	Start-code	Stop-code	BT2	BT5N	BT20N
0	48	33	-	x	x	
1	49	34	-	x	x	
2	50	35	-	x	x	
3	51	36	-	x	x	
4	52	37	-	x	x	
5	53	38	-	x	x	
6	54	39	-	x	x	
7	55	40	-	x	x	
8	56	41	-	x	x	
9	57	42	-	x	x	
MINUS	111	125	x	x	x	
DEC PT.	112	126	-	x	x	
PLUS	113	124	x	x	x	
DATA RELEASE	108	122	x	x	x	
ENTER	109	121	x	x	x	
DELETE	110	123	-	x	x	
?-HELP	106	107	x	x	x	
CURSOR UP	101	118	-	x	x	
CURSOR DOWN	102	119	x	x	x	
CURSOR LEFT	103	116	x	x	x	
CURSOR RIGHT	104	117	-	x	x	
HOME	105	120	-	x	x	

Key	Start-code	Stop-code	BT2	BT5N	BT20N
F1	65	66	x	x	x
F2	67	68	x	x	x
F3	69	70	x	x	x
F4	71	72	x	x	x
F5	73	74	-	x	x
F6	75	76	-	x	x
F7	77	78	-	-	x
F8	79	80	-	-	x
F9	81	82	-	-	x
F10	83	84	-	-	x
F11	85	86	-	-	x
F12	87	88	-	-	x
print	77	78	-	x	-
paging	79	80	-	x	-

4.7 Interface Control Characters

Control Characters: Control characters are interpreted by the terminal and can not be displayed on the display!

Character		Code	Function
Backspace	BS	8	Cursor is moved to the left by one column.
Linefeed	LF	10	Cursor is moved downwards by one line.
Return	CR	13	Cursor is moved to the beginning of the line.
Xon	XON	17	Ready-signal during software handshake
Xoff	XOFF	19	Not ready until next XON
Escape	ESC	27	Start code for all control sequences
Delete	DEL	127	Deletes character beneath cursor position

4.7.1 LED Codes for the Operating Terminals

Key	dec. Code	BT2	BT5N	BT20N
DATA RELEASE	20	x	x	x
?-HELP	19	x	x	x
F1	1	x	x	x
F2	2	x	x	x
F3	3	x	x	x
F4	4	x	x	x
F5	5	-	x	x
F6	6	-	x	x
F7	7	-	-	x
F8	8	-	-	x
F9	9	-	-	x
F10	10	-	-	x
F11	11	-	-	x
F12	12	-	-	x

4.7.2 Control Sequences for the Operating Terminals

Sequence	Description
<ESC>	Represents the ESC character.
<A>	Represents the line number.
<B>	Represents the column number.
	<A> and <B> correspond to a decimal number, specified through one or several ASCII characters. The limits are governed by the size of the display.
<An>	Represents a decimal number which selects a sub-function. Multiple sub-functions must be separated by semicolons.
<Pn>	Represents a numerical parameter. Denotes a decimal number.

BT2	BT5N	BT20N		
x	x	x	<ESC>[2J	Erase screen, and cursor in line 1 / column 1.
x	x	x	<ESC>[K	Erasing of the line: Deletes all characters to the right of the cursor position up to the end of the line, including the cursor position.
x	x	x	<ESC>[<A>;<B>H	Positioning of the cursor in line A of column B. A, B limited by the display.
x	x	x	<ESC>[<Pn>A	Cursor upwards: Moves the cursor up by the number of lines specified without changing the column. Within the uppermost line, the command will be ignored.
x	x	x	<ESC>[<Pn>B	Cursor downwards: Moves the cursor down by the number of lines specified without changing the column. Within the last line on the bottom, the command will be ignored.
x	x	x	<ESC>[<Pn>C	Cursor to the right: Moves the cursor to the right by the number of columns specified without changing the line. Within the column on the extreme right, the command will be ignored.
x	x	x	<ESC>[<Pn>D	Cursor to the left: Moves the cursor to the left by the number of columns specified without changing the line. Within the column to the extreme left, the command will be ignored.
x	x	x	<ESC>&C	Activate cursor.
x	x	x	<ESC>&D	Deactivate cursor.
x	x	x	<ESC>[s	Stores the cursor position: An additional ESC-sequence can be used to move the cursor back to the stored position.
x	x	x	<ESC>[u	Restores the cursor position: Moves the cursor back to the stored position.
-	-	x	<ESC>&Z	Characters are displayed in capitals (Zoom 2).
-	-	x	<ESC>&A	Characters are displayed in normal size.
x	x	x	<ESC>[<A1>;<An>m	Setting the character attributes. A1 to An corresponds to the attribute number
x	x	x		0 = deselects all attributes
-	-	x		4 = underline
x	x	x		5 = flashing
-	-	x		7 = inverse
x	x	x	<ESC>[<A1>;<An>q	Activate LED. A1 to An corresponds to the number of the LED
				A = 0 activate all LEDs
x	x	x	<ESC>&q<A1>;<An>:	LED flashing. A1 to An corresponds to the number of the LED
				A = 0 all LEDs flashing
x	x	x	<ESC>&p<A1>;<An>:	Deactivate LED. A1 to An corresponds to the number of the LED
				A = 0 deactivate all LEDs
x	x	x	<ESC>[6n	Inquiry by the controller as to which terminal is connected. Response provided by the terminal:
x	-	-		Response <ESC>[02
-	x	-		Response <ESC>[05
-	-	x		Response <ESC>[20

BT2	BT5N	BT20N		
x	x	x	<ESC>&t	Display time at cursor position. Format hh:mm:ss
x	x	x	ESC>&d	Display date at cursor position. Format TT.MM.JJ
x	x	x	<ESC>&JJMMTThhmmssi	Set Date / Time. JJ = Year MM = Month TT = Day hh = Hours mm = Minutes ss = Seconds
x	x	x	<ESC>&u	Transmit clock to interface X2. The response from the terminal to this request has the same format as the setting for the date/time above. Response: <ESC>&JJMMTThhmmssi
x	x	x	<ESC>[<A>;<B>U	Cyclic output of the time in line A and column B is started. The output occurs in accordance with the current character attributes.
x	x	x	<ESC>[0;0U	Cyclic output of the time is stopped.
x	x	x	<ESC>&K	Default value for the display's contrast setting.
x	x	x	<ESC>&+	Increments the selected value by +1. The contrast becomes "brighter". The maximum total range of values is ±125. This range of values is being limited to values appropriate for the respective display.
x	x	x	<ESC>&-	Decrements the selected contrast value by -1. The contrast is "dimmed".

4.8 Error Messages

Any error messages generated by the terminal are principally displayed on the top left of the display. The error messages are generated by the terminal software in plain text. The error message is displayed once at the cursor position 1,1. The previous text will be overwritten. The message will not be deleted! - but will rather need to be overwritten by the user. This must be taken into consideration when creating the program. The following internal error text might be displayed:

LOW BAT	Indicates that the capacity of the battery to back data is falling below the required level. Upon initial display of this message, the battery should be replaced within 3 days! A battery test is performed after the terminal has been switched on and every 60 minutes thereafter.
FRAMING ERROR	The data format of the characters received at the interface does not conform with the selected interface parameters. Check number of data bits, stop bits and parity bit.
PARITY ERROR	A parity error has been detected regarding the characters received at the interface. This indicates a possible problem with the transmission line (character length, baud rate, interference, invalid interface parameter etc.)
OVERRUN ERROR	The terminal was unable to interpret and evaluate the characters received at the interface. This indicates a possible error regarding the handshake procedure between terminal and controller. Check hardware or software handshake.
TRANSMIT ERROR	On account of a blocked transmitter (no handshake enable signal), the terminal was unable to transmit the characters within the specified timeout period. This indicates an error regarding the handshake procedure between terminal and controller. Check hardware or software handshake.
KEYBOARD ERROR PLEASE RELEASE KEY	The self-test performed on the keyboard after having switched on the terminal has detected that a key is being pressed. Please comply with the message. Display of this message without any key being pressed indicates a defect in the keyboard!

5 Controller and Bus Interfacing

5.1 TesiMod - 3964R Interfacing

5.2 TesiMod - BUEP19 Bosch PLC

5.3 TesiMod - DIN-Meßbus Master/Slave

5.4 TesiMod - BUEP19E Bosch PLC

5.5 TesiMod - PROFIBUS-DP

Table of Contents

5.1 TesiMod - 3964/RK512 Interfacing

5.1.1 General Information

Via the serial procedure 3964, the operating terminals of the TesiMod series can be connected to a programmable controller.

The logical part of the 3964 protocol, RK512, is adapted to the communication with a Siemens PLC via a CP525, or compatible, communications processor.

The hardware and software components of the TesiMod system are fully adapted to the parameters and marginal conditions of the protocol 3964/ RK512.

This offers the user of TesiMod operating terminals the following advantages:

- Random read and write access to any data within the PLC. Data of existing PLC programs can be displayed and modified directly in the operating terminal. It is not necessary to adapt the PLC program to the operating terminal in any respect since it is not required that communications data be stored in a specified address area or data type area.

- The TesiMod operating terminal automatically polls cyclic data in a free defined memory area.

- Simultaneous connection of theTesiMod operating terminal and the programming unit (PU) is possible.

- Only a minimum of configuration is required for the installation of the data handling block required in the PLC in addition to the protocol function blocks.

- Minimal increase of the cycle time in the PLC.

- The protocol provides error control. Transmission errors are detected and, if possible, eliminated by repeating the transmission.

- A noise-immune interface hardware in accordance with the 20mA current loop interface standard permits the application even in a harsh industrial environment.

- The graphical operator guidance offers the user a maximum of assistance in defining the variable addresses. The definitions (abbreviations) used here are identical with the definitions used within the PLC program.

5.1.2 Technical Description

The terminal is always the active partner which either requests data from or sends data to the partner.

Direct read-access is possible to all PLC data.
Direct write-access is limited to data blocks only.

The installation of the supplied function block allows an indirect write-access to all PLC data types.

All byte-structured data types can also be accessed in bit-mode.
It is also possible to access all inidividual bytes of a data word within a data block.

5.1.3 Parameters Interface SER1

The following interfave parameters are available:

Baud rate:	300, 600, 1200, 2400, 4800, **9600**, 19200 Baud
Parity:	none, **even**, odd
Data length:	5, 6, 7, **8** Bits
Stop bits:	1, 1.5, **2** Stop bits
Handshake:	**no handshake**, hardware handshake, software handshake

The **fat** printed settings are standard.

The following protocol parameter settings are also available for interface **X2**:
- Communications block number
- Data word offset
- Coordination flag number
- Coordination flag bit
- Protocol with/without coordination flag
- Block check
- CPU number

5.1.3.1 Interface Parameters of the Communications Module

The interface parameters of the communications module must comply with the parameters of the TesiMod operating terminal.

5.1.3.2 PLC Configuration

To allow write-access to all PLC data, it is merely necessary to install the supplied function block and to execute it at cyclic intervals.
In addition, a coordination flag must be defined and a data block must be created as communications block with a size of 128 bytes.
This communications block and coordination flag must be specified to the supplied function block as parameter.

5.1.4 Data Type Structure

a) Alphanumerical Texts
Are stored in the memory byte for byte in ascending address order.

b) Counter
A distinction is made between variables which have been assigned a counter address and variables which have been assigned another PLC address

Counter address
When accessing counter addresses, the count value is interpreted in the binary format, the control bits of the counter are masked out. Therefore, to avoid control bits from being erased, counter addresses should be accessed in the **read-mode** only.

All other addresses
The count value is interpreted in BCD-code. This allows the transfer of this value within the PLC program to the counter by means of the accumulator. This function should be used for indirect write-operations of count values since the values are available in the Siemens conformal format.

c) Timer
Timer functions consist of a time value and a time base. The terminal operates with imaginary unsigned 4-byte variables, even though the data stored in the PLC comprise only 2 bytes.

When read-accessing the timer, the terminal converts the time value and time base into a terminal-internal unsigned 4-byte number, which represents the time value in reference to the time base of 0.01 second.

Ex.: A range of 10 (time base is 1.0 second) and a time value of 999, are represented or edited, respectively, in the terminal by the value 99900. Scaling of this value to other value ranges is possible by specifying a factor and divisor within the variable definition.

Before writing a timer variable to the PLC, the time value and the smallest possible time base are formed from the terminal-internal unsigned 4-byte value.

In addition, a distinction is made between variables which have been assigned a timer address or another PLC address.

Timer address
When accessing timer addresses, the time value is interpreted binary format. To avoid timer control bits from being erased, this access should occur in the **read-mode** only.

All other addresses
The time value is interpreted BCD-coded.This access should be used for indirect write-operations of time values since the values are available in the Siemens conformal format.

d) Floating Point Number
The data are interpreted in the Siemens floating point format.

e) Binary Variables with a Length of 1, 2 or 4 Bytes
Data with a length of 2 bytes are interpreted in the PLC-conformal byte order for words.
Data with a length of 4 bytes are interpreted in the PLC-conformal byte order for long words.

5.1.4.1 Data Types

Direct accessing of the following data types is possible:

I	input bits	(bit access)
Q	output bits	(bit access)
F	flag bits	(bit access)
IB	input bytes	(byte access)
QB	output bytes	(byte access)
FB	flag bytes	(byte access)
IW	input word	(word access)
QW	output word	(word access)
FW	flag word	(word access)
ID	input double word	(double word access)
QD	output double word	(double word access)
FD	flag double word	(double word access)
DW	data word	(word access)
DL	data word, left-hand (high)	(word access)
DR	data word, right-hand (low)	(word access)
DD	data double word	(double word access)
T	timer	(word access)
C	counter	(word access)

The size of each data area is governed by the CPU of the PLC.

5.1.4.2 Special Simatic-Data Formats

The following data formats are supported in the editors:

KB 0...255
Variable in byte-format

KF -32768...+32767
Variable in 16bit fixed point number-format

KH 0000...FFFF
Variable in 4-digit hexadecimal number-format

DH 00000000...FFFFFFFF
Variable in 8-digit hexadecimal number-format
KC !!...zz (2 ASCII-characters each)
Variable represented by 2 characters in ASCII-format
KT 000.0...999.3
Variable represented as time value
KZ 000...999
Variable represented as count value
KG ±1.2*10-38...±3.4*10+38
Variable in 32bit floating point number-format
KM 00000000 00000000...11111111 11111111
Variable in bit pattern-format

5.1.5 Additional Functions

In addition to the random write and read access to PLC variables, a memory area comprising 11 (12) bytes is specified in the mask definition as poll area. The location of this memory area can also be specified in the mask definition.

Only marginal conditions regarding this memory area:
- the PLC must be able to access in bit-mode and the terminal in byte-mode
- the memory area must be contiguous.

Byte-structured Memory Mapping

The data area comprises a maximum of 11 bytes

Example: The cyclic poll area is set to flag byte (FB) 12 in the TesiMod programming system.

Access to the PLC occurs via:

F12.0	ED	External data release
F12.1	RR	Bit for refresh request
BYTE address +0	FB12	Write coordination byte
BYTE address +1	FB13	Message channel low-byte
BYTE address +2	FB14	Message channel high-byte
BYTE address +3	FB15	Function keys LED1...4
BYTE address +4	FB16	Function keys LED5...8
BYTE address +5	FB17	Function keys LED9...12
BYTE address +6	FB18	Function keys LED13...16
BYTE address +7	FB19	Function keys LED17...20
BYTE address +8	FB20	Function keys LED21...24
BYTE address +9	FB21	Function keys LED25...28
BYTE address +10	FB22	Function keys LED29...32

Word-structured Memory Mapping

The data area comprises a maximum of 6 words or 12 bytes.

Example: The cyclic data area is set to DW21 in the TesiMod programming system

Word address	DW	Highbyte	Lowbyte
Word address +0	DW21	Write coordination byte	Reserved
Word address +1	DW22	Message channel high byte,	Message channel low-byte
Word address +2	DW23	Function keys LED1...4	LED5...8
Word address +3	DW24	Function keys LED9...12	LED13...16
Word address +4	DW25	Function keys LED17...20	LED21...24
Word address +5	DW26	Function keys LED25...28	LED29...32

5.1.6 Physical Interfacing

Pinning of the TesiMod 20mA current loop

Pin	Designation	Function
1	Shield	Shield
2	T+	Transmit Data, Positive Polarity
3	S1+	Power Source 2, Positive Polarity
4	R+	Receive Data, Positive Polarity
5	R-	Receive Data, Negative Polarity
6	S2+	Power Source 1, Positive Polarity
7	T-	Transmit Data, Negative Polarity
8	S1-	Power Sink 1, Negative Polarity
9	S2-	Power Sink 2, Negative Polarity

Pinning of the TesiMod RS232c interface

Pin	Designation	Function
1	*DCD*	*Data Carrier Detect*
2	RD	Receive Data
3	TD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	Signal Ground
6	*DSR*	*Data Set Ready*
7	RTS	Request to Send
8	CTS	Clear to Send
9	*RI*	*Ring Indicator*

Italic printed pins are not available.

Pinning of the TesiMod RS485 interface

Pin	Designation	Function
1	Shield	Shield
2	T(A)	Transmit Data Channel A
3	R(A)	Receive Data Channel A
4	RTS(A)	Request to Send Channel A
5	CTS(A)	Clear to Send Channel A
6	*TXCK(B)*	*Transmitter Clock Channel B*
7	*RXCK(B)*	*Receiver Clock Channel B*
8	SG	Signal Ground
9	T(B)	Transmit Data Channel B
10	R(B)	Receive Data Channel B
11	RTS(B)	Request to Send Channel B
12	CTS(B)	Clear to Send Channel B
13	*TXCK(A)*	*Transmitter Clock Channel A*
14	*RXCK(A)*	*Receiver Clock Channel* A
15	nc	not connected

Italic printed pins are not available.

Pinning of the TesiMod universal interface TTY / 20mA current loop

Pin	Designation	Channel	Function
10	T+	SER1	Transmit Data, Positive Polarity
12	S1+	SER1	Power Source 2, Positive Polarity
13	R+	SER1	Receive Data, Positive Polarity
14	R-	SER1	Receive Data, Negative Polarity
16	S2+	SER1	Power Source 1, Positive Polarity
19	T-	SER1	Transmit Data, Negative Polarity
21	S1-	SER1	Power Sink 1, Negative Polarity
24	S2-	SER1	Power Sink 2, Negative Polarity

Pinning of the TesiMod universal interface RS485

Pin	Designation	Channel	Function
8	T(A)	SER1	Transmit Data Channel A
9	T(B)	SER1	Transmit Data Channel B
11	SGND	SER1	Signal Ground
22	RD(A)	SER1	Receive Data Channel A
23	RD(B)	SER1	Receive Data Channel B

Pinning of the TesiMod universal interface RS232c

Pin	Designation	Channel	Function
1	Shield	SER2	Low-noise Earth
2	TD	SER2	Transmit Data

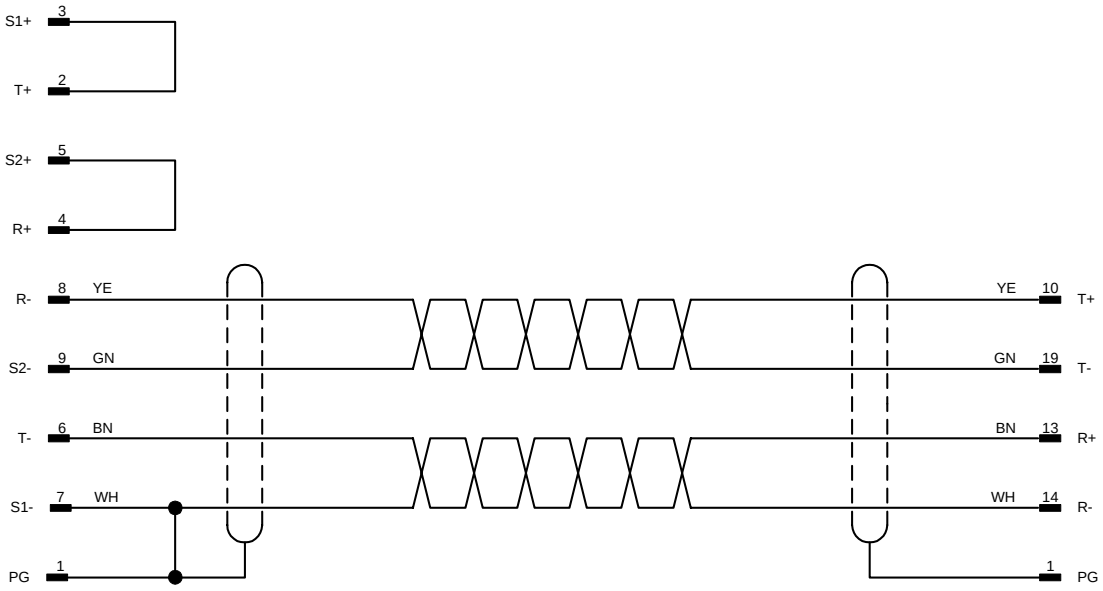
3	RD	SER2	Receive Data
4	RTS	SER2	Request to Send
5	CTS	SER2	Clear To Send
7	SGND	SER2	Signal Ground
20	DTR	SER2	Data Terminal Ready

5.1.6.1 Connecting Cable TTY / 20 mA - Siemens CP523/525

TesiMod
Operating Terminal

Sender active
Receiver active

Serial Interface Module for
Simatic-S5
CP 524/525
Helmholz SAS523/525
Sender passive
Receiver passive



D-Subminiature
Male connector
9 pin

D-Subminiature
Male connector
25 pin

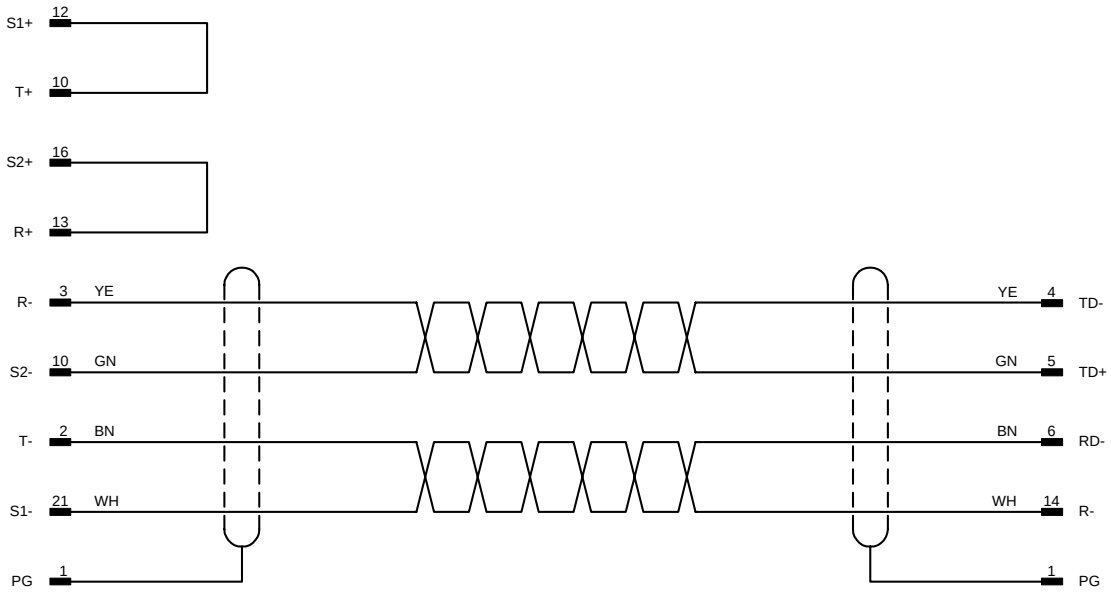
The shield is connected to the metal housing on both sides.

5.1.6.2 Connecting Cable Universal Interface TTY / 20 mA - Siemens CP523/525

TesiMod
Operating Terminal

Sender active
Receiver active

Serial Interface Module for
Simatic-S5
CP 524/525
Helmholz SAS523/525
Sender passive
Receiver passive



D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
25 pin

The shield is connected to the metal housing on both sides.

/000-0108/
Bosch_T_eng_V13_3005010QK0

5.1.6.3 Connecting Cable RS232 - Siemens CP523/525

TesiMod
Operating Terminal

Sender active
Receiver active

Serial Interface Module for
Simatic-S5
CP 523/525

Sender passive
Receiver passive

D-Subminiature
Male connector
9 pin

D-Subminiature
Male connector
25 pin

The shield is connected to the metal housing on both sides.

5.1.6.4 Connecting Cable Universal Interface RS232c - Siemens CP523/525

TesiMod
Operating Terminal
Sender active
Receiver active

Serial Interface Module for
Simatic-S5 CP525
Sender passive
Receiver passive

D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
25 pin

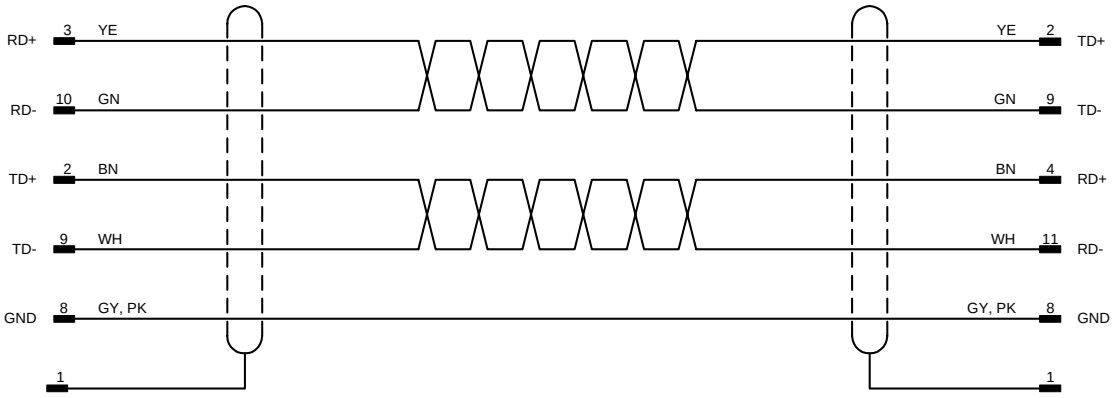
The shield is connected to the metal housing on both sides.

/000-0108/
Bosch_T_eng_V13_3005010QK0

5.1.6.5 Connecting Cable RS485 - Siemens CP524/525

TesiMod
Operating Terminal

Serial Interface Module for
Simatic-S5 System
Siemens CP524/525



D-Subminiature
Male connector
15 pin

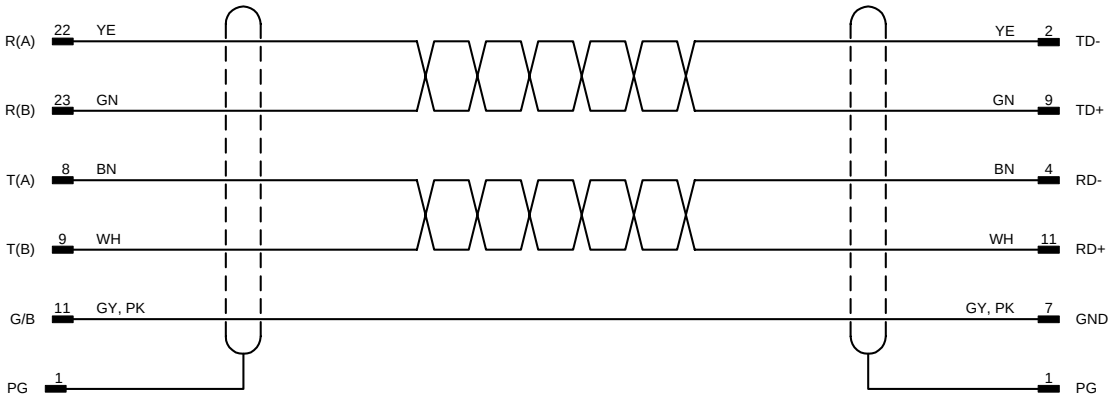
D-Subminiature
Male connector
25 pin

The shield is connected to the metal housing on both sides

5.1.6.6 Connecting Cable Universal Interface RS485 - Siemens CP524/525

TesiMod
Operating Terminal

Serial Interface Module for
Simatic-S5 System
Siemens CP524/525



D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
25 pin

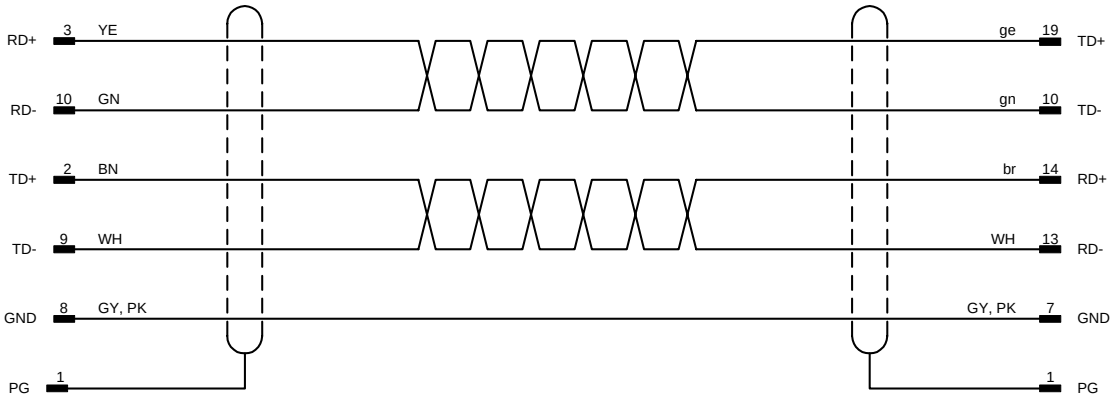
The shield is connected to the metal housing on both sides

/000-0108/
Bosch_T_eng_V13_3005010QK0

5.1.6.7 Connecting Cable RS485 - Helmholz SAS 523/525

TesiMod
Operating Terminal

Serial Interface Module for
Simatic-S5 System
SAS 523/525 Helmholz



D-Subminiature
Male connector
15 pin

D-Subminiature
Male connector
25 pin

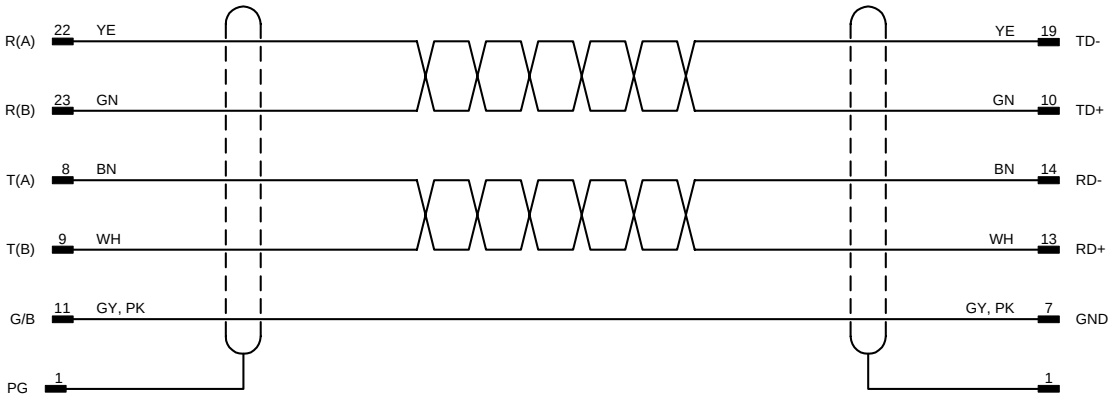
The shield is connected to the metal housing on both sides

CAUTION!
At the SAS 525 the bridges of the termination resistors for Peer-to-Peer connection must be closed as described in the manual of the PLC
At the TesiMod terminal the termination must be swiched to the ON-state.

5.1.6.8 Connecting Cable Universal Interface RS485 - Helmholz SAS 523/525

TesiMod
Operating Terminal

Serial Interface Module for
Simatic-S5 System
SAS 523/525 Helmholz



D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
25 pin

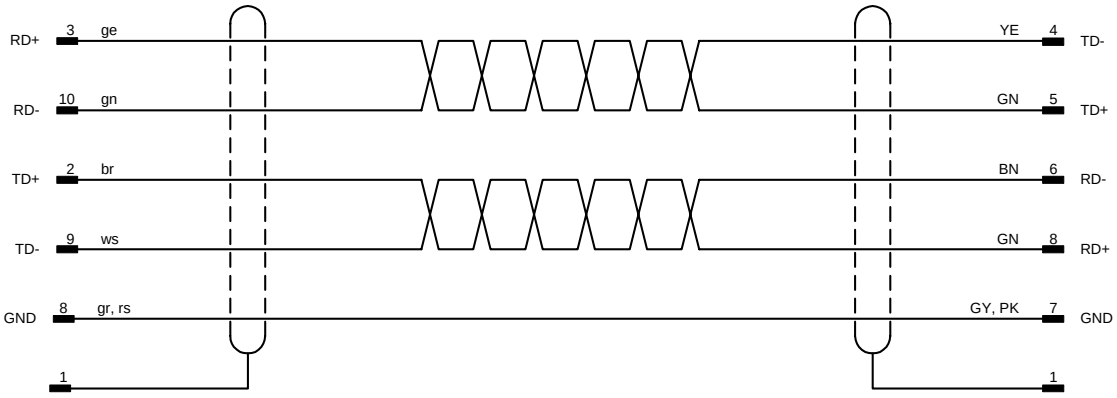
The shield is connected to the metal housing on both sides

CAUTION!
At the SAS 525 the bridges of the termination resistors for Peer-to-Peer connection must be closed as described in the manual of the PLC
At the TesiMod terminal the termination must be swiched to the ON-state.

5.1.6.9 Connecting Cable RS485 - VIPA BGM79-43

TesiMod
Operating Terminal

Serial Interface Module for
Simatic-S5 System
BGM79-43 VIPA



D-Subminiature
Male connector
15 pin

D-Subminiature
Male connector
25 pin

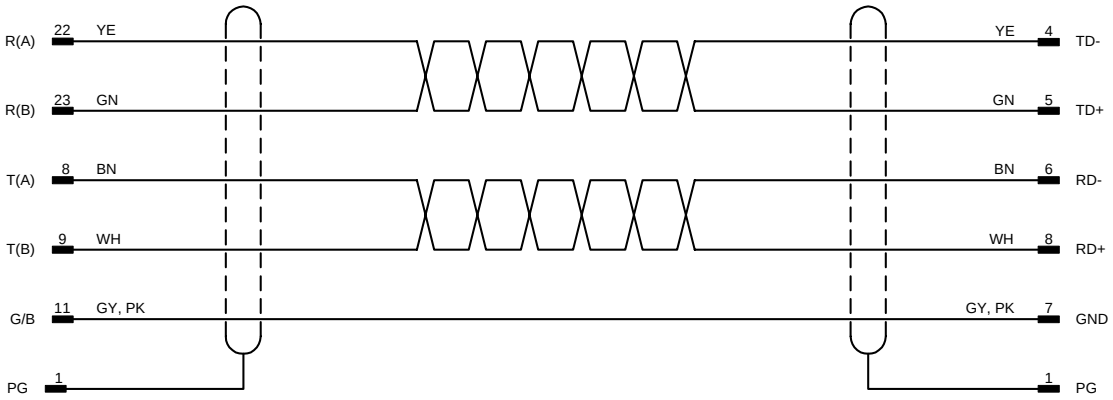
The shield is connected to the metal housing on both sides

CAUTION!
Only able to operate, when termination at the TesiMod terminal is switched to the ON-state.(Because of the TD-closedown on the VIPA-card)

5.1.6.10 Connecting Cable Universal Interface RS485 - VIPA BGM79-43

TesiMod
Operating Terminal

Serial Interface Module for
Simatic-S5 System
BGM79-43 VIPA



D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
25 pin

The shield is connected to the metal housing on both sides

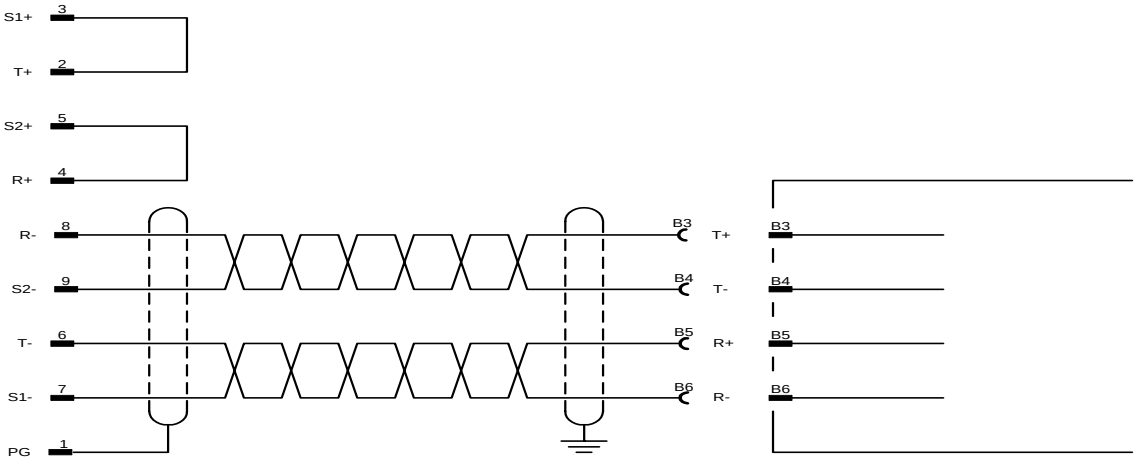
CAUTION!
Only able to operate, when termination at the TesiMod terminal is switched to the ON-state.(Because of the TD-closedown on the VIPA-card)

/000-0108/
Bosch_T_eng_V13_3005010QK0

5.1.6.11 Connecting Cable TTY / 20 mA - EBERLE PLS 514 - K43

TesiMod
Operating Terminal

Module K43
for EBERLE PLS 514



D-Subminiature
Male connector
9 pin

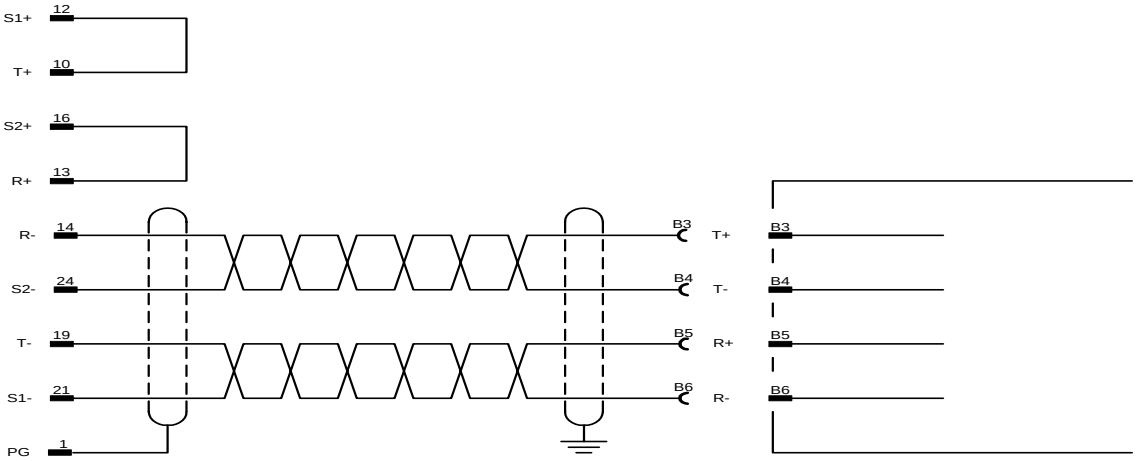
VG 95324 Plug Connector
DIN 41612 DIN 41612
42 pin 96 pin

The shield is connected to pin 1 / metal housing

5.1.6.12 Connecting Cable Universal Interface TTY / 20 mA - EBERLE PLS 514 - K43

TesiMod
Operating Terminal

Module K43
for EBERLE PLS 514



D-Subminiature
Male connector
25 pin

VG 95324 Plug Connector
DIN 41612 DIN 41612
42 pin 96 pin

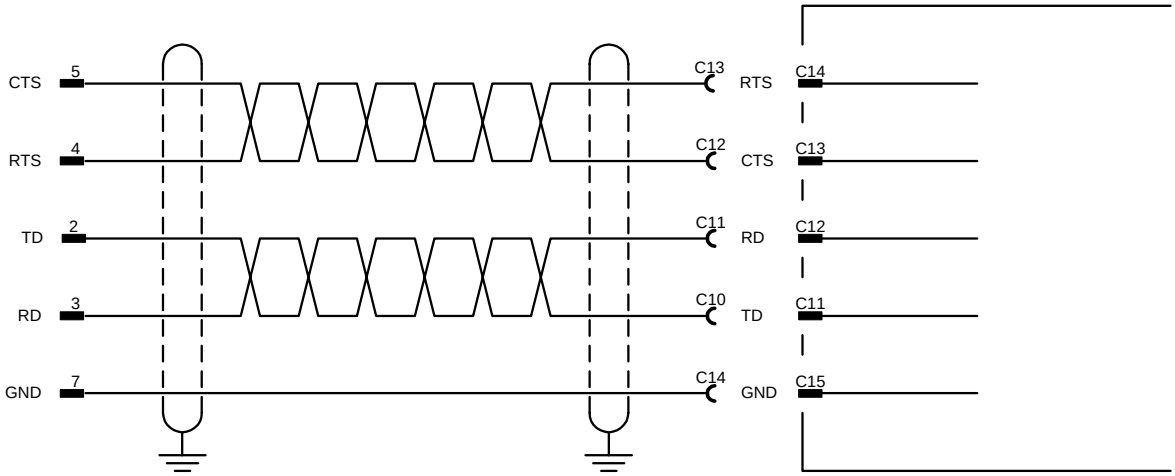
The shield is connected to pin 1 / metal housing

/000-0108/
Bosch_T_eng_V13_3005010QK0

5.1.6.13 Connecting Cable RS232 - EBERLE PLS 514 - K43

TesiMod
Operating Terminal

Module K43
for EBERLE PLS 514



D-Subminiature		VG 95324	Plug Connector
Male connector		DIN 41612	DIN 41612
9 pin		42 pin	96 pin

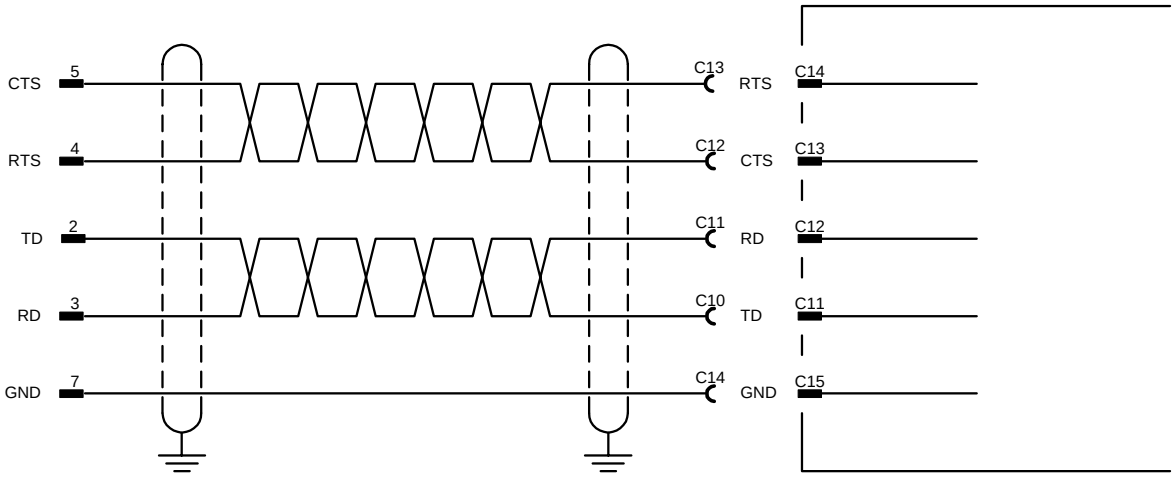
The shield is connected to the metal housing on both sides

5.1.6.14 Connecting Cable Universal Interface RS232 - EBERLE PLS 514 - K43

For the communication between the terminal and the communication module EBERLE PLS514-K43 the software handshake must be active. This setting is adjustable in the programming software of the terminal.

TesiMod
Operating Terminal

Module K43
for EBERLE PLS 514



D-Subminiature
Male connector
25 pin

VG 95324 Plug Connector
DIN 41612 DIN 41612
42 pin 96 pin

The shield is connected to the metal housing on both sides

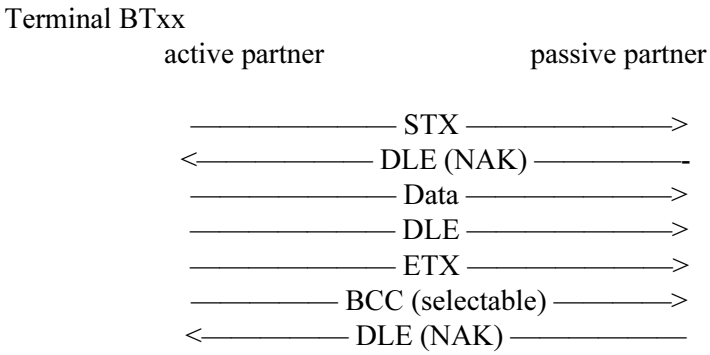
/000-0108/
Bosch_T_eng_V13_3005010QK0

5.1.7 3964 Procedure

To initiate the communication set-up, the active partner will transmit the signal STX (02h) upon which the partner will have to respond with DLE (10h) within the specified acknowledgement delay period (2s).
Subsequently, the procedure data are transmitted, the receipt of which is monitored by a character delay time (220ms).
After the data have been transmitted, the passive partner will acknowledge the receipt with DLE.

In the event of errors, the pasive partner will transmit a NAK (15h).
In the event of errors during the communication set-up, the active partner will make up to 3 attempts to establish the communication.

If the passive partner transmits a NAK after receipt of the data, the active partner will make up to 6 attempts to establish the communication and to transmit the data.



Please note:

If the character 10_{16} is to be transmitted but not to be evaluated as a DLE, the transmitting device will add another 10_{16}.
If the receiving device detects a 10_{16} (DLE) twice, it will accept 10_{16} only once and will not evaluate it as a DLE control character.

5.1.7.1 Block Check BCC

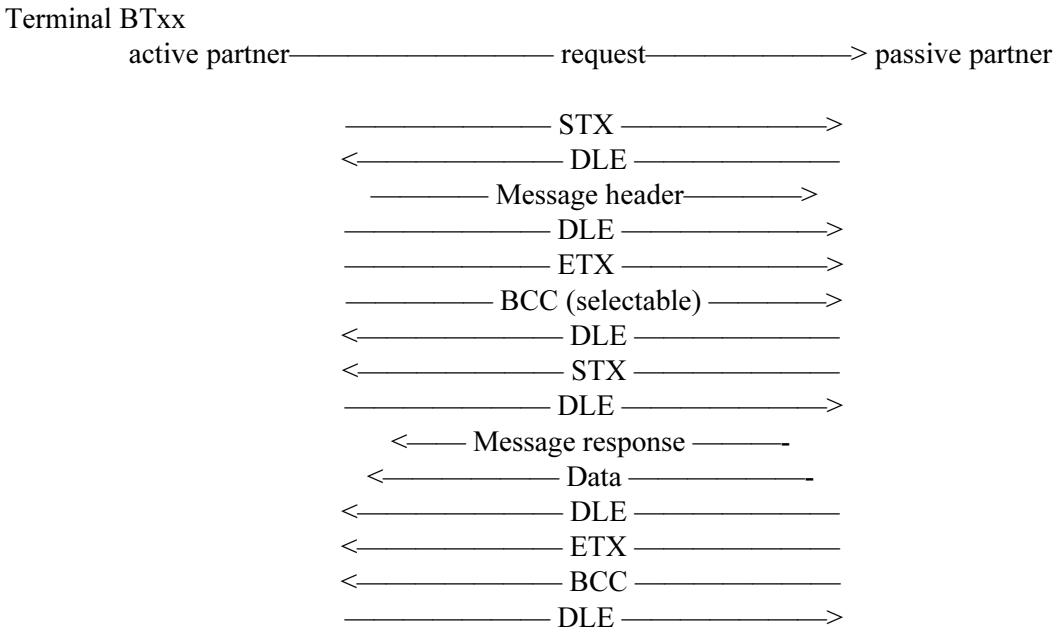
With the block check method (3964R), a block check character is created and added to the end of the block.

The BCC is formed through a logic XOR of all characters with the exception of the start character STX (02_{16}).

5.1.7.2 Logical Part of the Procedure 3964, RK512

A logical part, which complies with the Siemens protocol RK512 to a great extent, has priority over the physical part of the procedure.
This part governs the type and contents of the data part.

5.1.8 Message Request of Data



5.1.8.1 Structure Message Header (10 bytes) Request of Data

	Function	ASCII	Hex	Comment
1. Byte	Message		00	always 00
2. Byte	identifier		00	always 00
3. Byte	Data direction	E	45	E = Request
4. Byte	Command			see below
5. Byte	2 Byte			see below
6. Byte	Source			
7. Byte	2 Byte			see below
8. Byte	Number of bytes			
9. Byte	Coordination flag number FF indicates no			
10. Byte	coordination flag bit			FF Coord. flag

5.1.8.2 Data Specification in the Message Header

The data types of the PLC addressed by the terminal are implemented in the message header by the Bytes 4 - 8 as illustrated below.

Source	Access	Data type TesiMod	Command Byte 4	Word-Parameter Byte 5+6	Byte-Parameter Byte 5	Byte 6	Number in Byte 7+8
Input	Bit	E	E		Offset	Bit-Nr	1 Byte
Output	Bit	A	A		Offset	Bit-Nr	1 Byte
Flag	Bit	M	M		Offset	Bit-Nr	1 Byte
Input	Byte	EB	E	Offset			n Bytes
Output	Byte	AB	A	Offset			n Bytes
Flag	Byte	MB	M	Offset			n Bytes
Input	Word	EW	E	Offset			n Bytes
Output	Word	AW	A	Offset			n Bytes
Flag	Byte	MB	M	Offset			n Bytes
Input	D-Word	ED	E	Offset			n Bytes
Output	D-Word	AD	A	Offset			n Bytes
Flag	D-Word	MD	M	Offset			n Bytes
D-Block	Word	DW	D		DB	DW	n Words
High-Byte DB	Byte	DL	D		DB	DW	1 Word
Low-Byte DB	Byte	DR	D		DB	DW	1 Word
D-Block	D-Word	DD	D		DB	DW	n Words
Timer	Word	T	T	Offset			n Words
Counter	Word	Z	Z	Offset			n Words

5.1.8.3 Coordination Flag

A flag bit is specified in the message header which is used by the receiving device to monitor the receipt of data.
The monitoring function is deactivated if the value for the coordination flag is FF_{16}, FF_{16}.

If a coordination flag is specified, it will be set in the passive partner upon receipt of data.
Once this flag has been set, processing of the data received will be initiated.
After the data have been processed, the flag will be reset again.
If this flag is still set upon the receipt of data, the passive partner will transmit a response message to the active partner thereby indicating the respective error.

NOTE:
For connections to a Siemens communications module CP524/525, the coordination flag in DB1 must be entered into the list of inter-processor communication flags.

5.1.8.4 Structure 4-Byte-Sized Response Message

	Function	ASCII	Hex	Comment
1. Byte	Message		00	always 00
2. Byte	identifier		00	always 00
3. Byte				always 00
4. Byte	Error code		xx	

5.1.9 Message Transmission of Data

Terminal BTxx

active partner ——————— transmit ———————> passive partner

——————— STX ———————>
<——————— DLE ———————-
————Message header————>
——————— Data ———————>
——————— DLE ———————>
——————— ETX ———————>
———— BCC (selectable) ————>
<——————— DLE ———————-
<——————— STX ———————-
——————— DLE ———————>
<—— Response Message ———
<——————— DLE ———————-
<————————ETX ———————-
<——————— BCC ———————-
——————— DLE ———————>

5.1.9.1 Structure Message Header (10bytes) Transmission of Data

	Function	Ascii	Hex	Comment
1. Byte	Message		00	always 00
2. Byte	identifier		00	always 00
3. Byte	Data direction	A	41	A = Transmission
4. Byte	Command	D	44	D = Data block
5. Byte	2 Byte Destination			Data block
6. Byte				Data word
7. Byte	2 Byte number of data			Number of words
8. Byte				
9. Byte	Coordination flag number			FF indicates without
10. Byte	Coordination flag bit			FF Coord. flag

The destination for data to be transmitted is always a data block.

5.1.9.2 Special Features of the Protocol 3964R

A write-access to data is possible in data blocks only.
I.e. only the data tpye DW allows a direct write-access to the respective destination.

To allow a write-access to all data yet, the data are transmitted to a defined data block, the communications block.

In this case, the destination, comprising 4 bytes, must be added in front of the data prior to transmission.

5.1.9.3 Assignment of Bytes 1-4

Destination	Access	Data type TesiMod	Command Byte 1	Data-Number Byte 2	Byte-Parameter Byte 3	Byte 4	Word-Parameter Bytes 3+4
Input	Bit	I	10	1 Byte	Bit-No.	Offset	
Output	Bit	Q	11	1 Byte	Bit-No.	Offset	
Flag	Bit	F	12	1 Byte	Bit-No.	Offset	
Input	Byte	IB	1	n Bytes		Offset	
Output	Byte	QB	2	n Bytes		Offset	
Flag	Byte	FB	3	n Bytes		Offset	
Input	Word	IW	1	n Bytes		Offset	
Output	Word	QW	2	n Bytes			Offset
Flag	Word	FW	3	n Bytes			Offset
Input	Word	ID	1	n Bytes			Offset
Output	Word	QD	2	n Bytes			Offset
Flag	Word	FD	3	n Bytes			Offset
HIGH-Byte DB	Byte	DL	26	1 Byte	DB	DW	
LOW-Byte DB	Byte	DR	27	1 Byte	DB	DW	
Counter	Word	C	20	n Bytes			Offset
Timer	Word	T	21	n Bytes			Offset

A write-access to these data types is principally carried out with a coordination flag.
Just like the communications block, the coordination flag must be defined such that the settings in the terminal and passive device are in agreement.

Part of a program installed in the receiving device, denoted in the PLC as data handling block, will monitor the coordination flag.

If the coordination flag is set, the data will be processed by the data handling block in accordance with the first 4 bytes in the communications block.
After the data have been processed, the coordination flag will be erased thereby allowing further transmissions.

5.1.10 Protocol 3964R - Restrictions

The protocol 3964 allows data comprising a maximum of 128 bytes to be transmitted per message. No further messages are transmitted or processed.

5.1.11 Function Block for Siemens 115 U

The supplied function block FB186 can be implemented in a Siemens 115 U.

The function block processes data from the terminal which are to be transmitted to the PLC.

The first 4 bytes of the data which are written to the defined communications data block are interpreted in accordance with the table "Assignment of Bytes 1-4".

The function block FB186 supports the following commands from the table "Assignments of Bytes 1-4":

2	Output Byte
3	Flag Byte
11	Output Bit
12	Flag Bit.

When activating the function block, the communications data block and the coordination flag must be specified as parameter. These parameters must comply with the settings in the terminal.

The defined communications data block must comprise a size of 128 bytes or 64 data words, respectively.

5.1.12 Application Example for CP525 in 115U

In this example:
The communications data block is DB33 starting at DW0.
The coordination flag is M100.3.
Flag 50 is to be written to.

The coordination flag, also referred to as interprocessor communication flag by Siemens, must be activated on the CP525 via a hardware jumper (see reference material CP525).

In addition, the interprocessor communication flag must be defined in DB1 of the 115U. An illustration is shown below:

```
DB1:
DW  0:   KH=4D41 \
     1:   KH=534B  >      "MASK01"     Header identifier
     2.   KH=3130  /

DW  3:   KH=CA00                       Output-interprocessor
                                       communication flag
     4:   KF=100                       MB100

DW  5:   KH=EEEE                       End identifier
```

The following structure must be implemented in OB1. This block is executed at cyclic intervals.

```
      :     .
      :     .
      :SPA FB245                Execution of the RECEIVE-block
NAME:REC-ALL

ANZW:MW220                      The interprocessor communication flag number is
                                entered in the indication word parameter of the
            .                   RECEIVE-block
            .
      :L    MB221               If the defined  interprocessor communication flag is in
                                the LOW-Byte
      :L    KF+100
      .!=F
      :S    M 100.3             it must be set here.
      :     .
      :     .
      :SPA FB186                Evaluation block
                                Processes the data of the communcations data block
                                and resets flag 100.3.
```

5.1.13 Initialization for module K43 in EBERLE PLS514

The following settings are required to communicate with the module K43 in a EBERLE PLS 514:

For TTY connection:

Baud rate:	9600
Parity:	even
Databits:	8
Stopbits:	1
Handshake:	no handshake

Coordination flag:	no
Special PLC communication:	no
DB:	0
DW:	0
Block check:	yes
CPU number inside PLC:	0
Floating point numbers in the IEEE format:	no

For RS232 connection:

Baud rate:	9600
Parity:	even
Databits:	8
Stopbits:	1
Handshake:	hardware handshake

Coordination flag:	no
Special PLC communication:	no
DB:	0
DW:	0
Block check:	yes
CPU number inside PLC:	0
Floating point numbers in the IEEE format:	no

For timers, BCD numbers and counters always used double words (32 bit). For example DW0.

The values in the INI.IL file are used to initialize the K43 module. The values in this file are set as follows.

```
{Initialize the K43 module}

*********************************************************}

{       Communication data block:        INI.PBS}

{       Version:                         01 01}

{       No changes in the program}

{ *********************************************************}

{ }
{ *********************************************************}
{       Initialization of the K43}
{ *********************************************************}

{       This program is the initialization for the }
{       communication module K43.}
{       The initialization starts as soon as the supply voltage }
{       of the PLS 514 is switched on. }

{       The needed flags will initialized in the first cycle of the PLS 514 }
{       and the initialization program initiates. }

{       The parameters for the K43 module are transmitted }
{       in the initialization program. The initialization program }
{       is not necessary if the default setting is used. }

{       After the initialization, the operating mode data exchange }
{       with the K43 switches over automatically. }

{       The operating mode of the K43 depends on }
{       the commando digit. }
{       The commando digit is digit number 3 on layer 0. }
```

```
{       The function of the bits is shown below. }

{   Bit   3   2   1   0}
{       _________________}
{       !   !   !   !   !}
{       ! X ! X ! X ! X !}
{       !   !   !   !   !}
{       ________}
{         !   !   !   !}
{         !   }
{         !   !   !   0  = no SW-Reset }
{         !   1   !   1  = SW-Reset }
{         !   }
{         !   !   X      = no function }
{         !}
{         !   0          = K 43 operating mode passive }
{         !   1          = K 43 transmission over V.24 TTY }

{         0              = initialization }
{         1              = transmit data with K 43 active }

{       Reset of the initialization shift register and }
{       the necessary flags in the first cycle. }
{       To jumper the self-test of the interface module K43 }
{       a closing delay in the PLS 514 is necessary. }
{       The closing delay is managed with the auxiliary }
{       counter #Z0 bis #Z2.}
{       ————————————————————}

L       %ZK1
LD      %K 0
=D      INIREG
=D      Z0
=D      Z1
=D      Z2
=D      DW_ZV_0
=D      DW_ZV_1
=D      DW_ZV_2
=D      DW_ZV_3

L       %MS10
ZV      Z0
ZV      Z1
ZV      Z2

L       %K 1
LD      Z2
GL      %K 2
LD      INIREG
GL      %K 0
S       INIREG0

{       Initialize the protocol:}
{       ==============================}

{       To initialize the protocols the digits }
{       0 .. 2 of layer 0 and the digits of layer 1}
{       have the described meaning }


DAL     K43_E0

L       INIREG0
A       DACK
```

/000-0108/
Bosch_T_eng_V13_30050100QK0

```
=       %NOP

{       Definition of the commando digit 3 of layer 0 }
{       ————————————————————————}

{       Setting the data transmission to the K43 on initialize mode }

{       Digit 0xx3}

{   Bit   3   2   1   0}
{       ———————————————}
{       !   !   !   !   !}
{       ! X ! X ! X ! X !}
{       !   !   !   !   !}
{       ————————}
{         !   !   !   !}
{         !   }
{         !   !   !   0  = no SW-Reset }
{         !   1   !   1  = SW-Reset }
{         !   }
{         !   0          = K 43 operating mode passive }
{         !   1          = K 43 transmission over V.24 TTY }

{         0              = initialization }
{         1              = transmit data with K43 active }

{       Input of the constant 00 in the digit 0xx3 }
{       that means initialize of the K43 }
{               without SW-Reset}
{               Initialize protocol in accordance with layer 0, }
{               digit 0...2 and layer 1, digit 0...7}

LD      %K 0
=D      KOMMAND

{       Initialize with layer 0 digit 0,1 and 2:}
{       ————————————————————————}
{       Digit 0xx0}

{   Bit   3   2   1   0}
{       ———————————————}
{       !   !   !   !   !}
{       ! X ! X ! X ! X !}
{       !   !   !   !   !}
{       ————————}
{         !   !   !   !}
{         !   }
{         !   0   0   0  =   110 Baud }
{         !   0   0   1  =   300 Baud }
{         !   0   1   0  =   600 Baud }
{         !   0   1   1  =  1200 Baud }
{         !   1   0   0  =  2400 Baud }
{         !   1   0   1  =  4800 Baud }
{         !   1   1   0  =  9600 Baud }
{         !   1   1   1  = 19200 Baud       }

{         0              = data format 7 Bit }
{         1              = data format 8 Bit }

{       Input of the constant 14 in the digit 0xx0 }
{       that means 9600 baud, data format 8 bit}

LD      %K 14
=D      Data00
```

```
{       Digit 0xx1}
{       ─────}
{ }
{   Bit   3   2   1   0}
{       _________________}
{       !   !   !   !   !}
{       ! X ! X ! X ! X !}
{       !   !   !   !   !}
{       ─────────}
{         !   !   !   !}
{         !   !   !}
{         !   !   !   0  = no parity bit, no test  (none)}
{         !   !   !   1  = parity like bit 1.1 and 1.2 }
{         !   !   }
{         !   0   0   1  = test for odd parity   (odd)}
{         !   1   0   1  = test for even parity     (even)}
{         !   }
{         !   0   1   1  = parity bit always set „1" (mark)}
{         !                 no test                  }
{         !   1   1   1  = parity bit always set „0" (space)}
{         !                 no test                  }

{         0              =  1 stop bit}
{         1              =  2 stop bit}

{       Input of the constant 05 in das Digit 0xx1 }
{       that means even parity, 1 stop bit }
{ }
LD      %K 5
=D      Data01


{       DIGIT 0xx2}
{       ─────}

{   Bit   3   2   1   0}
{       _________________}
{       !   !   !   !   !}
{       ! X ! X ! X ! X !}
{       !   !   !   !   !}
{       ─────────}
{         !   !   !   !}
{         !   !   !}
{         !   !   !   0  = RTS/CTS switched on}
{         !   !   !   1  = RTS/CTS switched off}
{         !   !   }
{         !   !   0      = V 24}
{         !   !   1      = TTY}
{         !   }
{         !   0          = active transmission with low priority}
{         !   1          = active transmission with high priority}

{         0              = protocol 3964 active}
{         1              = protocol 3964 R active}

{       Input of the constant 09 in the digit 0xx2 with RS232 }
{       Input of the constant 11 in the digit 0xx2 with TTY }
{       that means handshake XON/XOFF, V24 interface, }
{       low priority, protocol 3964 R active.}

LD      %K 11
=D      Data02
```

```
{       Initialize layer 1}
{       ==========================}

{       The initialize of the data blocks is made in }
{       hexadecimal code. }
{       Two half bytes result in one character and }
{       have to put in two digits.}
{  }
{       Initialize the data block number 0}
{       ——————————————————}

{       Digit 1xx0  least significant half byte of the data block number 0}
{       Digit 1xx1  more significant half byte of the data block number 0}
{           }
{  ->  for example: 1st data block on 32 dec. => 20 hex. }
{ }
LD      %K 0
=D      Data10
LD      %K 2
=D      Data11

{       Initialize the 1st data block}
{       ——————————————————}
{                                                 }
{       Digit 1xx2  least significant half byte of the 1st data block}
{       Digit 1xx3  more significant half byte of the 1st data block}

{  ->  for example:  1st data block on 33 dec. => 21 hex. }

LD      %K 1
=D      Data12
LD      %K 2
=D      Data13

{       Initialize the 2nd data block}
{       ——————————————————}

{       Digit 1xx4  least significant half byte of the 2nd data block}
{       Digit 1xx5  more significant half byte of the 2nd data block}

{  ->  for example:  2nd data block on 34 dec. => 22 hex. }
{   }
LD      %K 2
=D      Data14
LD      %K 2
=D      Data15
{           }
{       Initialize the 3rd data block }
{       ——————————————————}

{       Digit 1xx6  least significant half byte of the 3rd data block }
{       Digit 1xx7  more significant half byte of the 3rd data block }
{           }
{  ->  for example:  3rd data block on 35 dec. => 23 hex. }

LD      %K 3
=D      Data16
LD      %K 2
=D      Data17

{       End the initialization }
{       ——————————————}
```

```
SL      INIREG
DAS     K43_E0
DAS     K43_E1
DAL     K43_E0
R       %NOP

{       Start data transmisssion }
{       ========================}

{       After initialization the data transmission is }
{       enabled if the commando digit is set. }

{       Meaning of the commando digit during the data transmisssion }

{   Bit   3   2   1   0}
{         _______________}
{         !   !   !   !   !}
{         ! X ! X ! X ! X !}
{         !   !   !   !   !}
{         ________}
{           !   !   !   !}
{           !   }
{           !   !   !   0  = no SW-Reset }
{           !   1   !   1  = SW-Reset }
{           !   }
{           !   0          = K 43 operating mode passive }
{           !   1          = K 43 transmisssion over V.24 TTY }

{           0              = initialization }
{           1              = data transmisssion with K43 active }


{       Definition of the commando digit 3 of layer 0 }
{       with the constant 08 }
{       that means data transmission with block transmission mode }
{       ____________________________}

L       INIREG1
A       DACK
=       %NOP
LD      %K 8

=D      KOMMAND

{       Stepping the initialization }
{       ___________________}

SL      INIREG
DAS     K43_E0
DAS     K43_E1
DAL     K43_E0
R       %NOP
```

```
{ ********************************************************}
{       Reset the K 43}
{ ********************************************************}

{       SW_Reset of the K 43 with the flag #Bed_Res}
{       by setting the commando digit }

{       Meaning of the commando digit}

{   Bit   3   2   1   0}
{         _________________}
{         !   !   !   !   !}
{         ! X ! X ! X ! X !}
{         !   !   !   !   !}
{         _________}
{           !   !   !   !}
{           !   }
{           !   !   !   0  = no SW-Reset }
{           !   1   !   1  = SW-Reset }
{           !   }
{           !   0           = K 43 operating mode passive }
{           !   1           = K 43 transmisssion over V.24 TTY }

{           0               = initialization }
{           1               = data transmisssion with K43 active }

{       Definition of the commando digit 3 of layer 0 }
{       with the constant 09 }
{       that means data transmission and SW-Reset }
{       ____________________________}

L       Bed_Res
A       Inireg2
A       DACK
=       %NOP

LD      %K 12
=D      KOMMAND

{       Initiate initialization programm new }
{       ____________________}

LD      %K 0
=D      INIREG
DAS     K43_E0
DAS     K43_E1
DAL     K43_E0
R       %NOP


Var
Bed_Res %       0007.0  { condition of reset K 43}
INIREG  %       0270    { ini. shift register}
INIREG0 %       0270.0  { ini. protocols}
INIREG1 %       0270.1  { ini. data transmission}
DW_ZV_2 %       0273    { data word counter 2}
DW_ZV_3 %       0274    { data word counter 3}
Z0      %       0275    { auxiliary counter 0}
Z1      %       0276    { auxiliary counter 1}
Z2      %       0302    { auxiliary counter 2}
End_Var
```

5.1.14 Error Messages

These error messages are displayed on the terminal.

Code	1	E_SLAVE_NOT_READY	Slave not ready, no connection
	2	E_PROTOCOL	Invalid character, no repetition
	3	E_FRAME	Byte frame error, despite repetition
	4	E_TIMEOUT	Timeout error
	5	E_CRC_BCC	CRC error, despite repetiion
	6	E_PARITY	Parity error, despite repetition
	7	E_SEND_ABORT	Abort send process
	8	R_REC_ABORT	Abort receive process
	9	E_BUF_SIZE	Insufficient cyclic buffer
	10	E_NO_DEFINE	No cyclic data defined
	11	E_DEFINE	Cyclic data already defined
	15	E_NO_PROTOCOL	Protocol is not supported
	16	E_OVERRUN	Overrun
	40	E_SYS_ADDRESS	Undefined system variable
	50	E_QUITTUNG_OPEN	Invalid acknowledge during communication set-up
	51	E_QUITTUNG_DATA	Invalid acknowledge after transmission of data
	52	E_NO_RESPONSE	No response message
	53	E_RECEIVE_COUNT	Incorrect number of data received

Errors which are transmitted by the programming controller (PU) via the response message.

61	E_NO_AG 10 from PU	No connection to PU
62	E_WRONG_ORDER 16 from PU	Invalid command in message header
63	E_INV_DEST 20 from PU	Invalid destination has been addressed
64	E_KOO_MERKER_SET 50 from PU	Coordination flag is still set
65	E_SEND_COUNT 52 from PU	Number of data transmitted does not comply with the specification in the message header

66 E_SYNCH
54 from PU ..... Awaiting response message
70 E_AG ..... The subcode contains the error which is transmitted in the response message by the PLC

Subcode

10 ..... No connection to PU
12 ..... Start address to high Using co-ordination flag as data type not allowed. CPU-no. to high
16 ..... Invalid Opcode
20 ..... DB not available DB to short
50 ..... Co-ordination flag still set
52 ..... More data received than expected Less data received then expected
54 ..... Synchonisation error (following telegram expected)

Table of Contents

5.2 TesiMod - Bosch PU-Interfacing via BUEP19

5.2.1 General Information

TesiMod operating terminals allow for a simple connection to the Bosch PLCs thus making TesiMod operating terminals the perfect man-machine-interface for your Bosch PLC.
The TesiMod operating terminal is connected to the respective PLC-module.
The data communication on the interface is handled by the PU protocol BUEP19. With the CPUs ZE300 / ZE301 / R300 / R301 / R600 communication is possible with any system implementing the BUEP19 protocol.

The hardware and software components of the TesiMod system are fully adapted to the parameters and marginal conditions of the PU interface.

This offers the user of TesiMod operating terminals the following advantages:

- Random write and read access to any data within the PLC. Data of existing PLC programs can be displayed and modified directly in the operating terminal. It is not necessary to adapt the PLC program to the operating terminal in any respect since it is not required that communications data be stored in a specified address area or data type area.

- The TesiMod operating terminal automatically polls cyclic data in the cyclic poll area. The location of this poll area are custom assignable.

- No configuration required within the PLC.

- The PU protocol is handled entirely by the firmware of the PLC. A PLC program (function blocks, etc.) in the PLC is not required for the handling of the communication.

- The protocol provides error control. Transmission errors are detected and, if possible, eliminated by repeating the transmission. An electrically isolated, noise-immune interface hardware in accordance with the 20mA current loop interface standard permits the application even in a harsh industrial environment.

- The parameters of the communications interface are assigned in the TS programming software in a protocol-specific manner and are stored in the mask definition. Modifying of the parameters is also possible in the set-up mask or any other I/O mask of the terminal.

- The graphical operator guidance offers the user a maximum of assistance in defining the variable addresses within the list of variables. The definitions (abbreviations) used here are identical with the definitions used within the PLC program.

5.2.2 Technical Description

The interfacing of the TesiMod operating terminal to the Bosch PLCs is effected by means of the BUEP19 PU-protocol.

The PU protocol BUEP19 allows random read and write access to all PLC data. Any byte-structured data types can also be accessed in bit-mode. The size of the address area depends on the respective PLC.

A read access must occur, before individual bits or bytes of a flag word can be accessed for a write operation. Subsequently, a write access is possible to the entire data structure. When accessing individual bits or bytes, special care must be taken to ensure that neither the terminal nor the PLC modify individual bits within one byte (or individual bits within one word, respectively).

The following byte order applies to the access in word-mode: **HighByte-LowByte**.

5.2.3 Parameters Interface X2 for PU-Interfacing

The TesiMod operating terminal adapts to the default parameters of the PU interface. It is therefore **not** necessary to modify the interface parameters in the PLC.
To ensure a proper communication, the parameters must **not** be modified.

Baud rate:	**9600** Baud
Parity:	**even**
Data length:	**8**
Stopbits:	**1**
Handshake:	**no handshake**

5.2.3.1 Protocol Parameters Target Module

To ensure an error-free data transmission, the terminal must be informed of which module is to be connected to interface X2.
ZE300 / ZE301
R300 / R301
R600

5.2.3.2 Protocol Parameter Block Check

As a default, the protocol BUEP19 implements the data block check method CRC16.
The programming device with a EP/AG module, however, uses a data block check method in accordance with LRC8.
To avoid difficulties during the development phase during which the PU and the terminal are alternately connected to the PLC, the terminal allows selection of the block check method LRC8.

5.2.3.3 Protocol Parameter Coordination Flag

The protocol permits defining of a coordination flag.

5.2.4 Data Type Structure

a) Alphanumerical Text
Is stored in the memory byte for byte in ascending address order.

b) Counter
The count value is interpreted in binary format. The maximum value is 8191.

c) Timer
Timer functions consist of a time value and a time base. The terminal operates with imaginary unsigned 4-byte variables, even though the data stored in the PLC comprise only 2 bytes.

When read-accessing the timer, the terminal converts the time value and time base into a terminal-internal unsigned 4-byte number, which represents the time value in reference to the time base of 0.01 second.
Ex.: A range of 10 (time base is 1.0 second) and a time value of 999 , are represented or edited, respectively, in the terminal by the value 99900. Scaling of this value to other value ranges is possible by specifying a factor and divisor within the variable definition.
Before writing a timer variable to the PLC, the time value and the smallest possible time base are formed from the terminal-internal unsigned 4-byte value.

d) Floating Point Numbers
The data are interpreted in the Siemens floating point format.

e) Binary Variables with a Length of 1, 2 or 4 Bytes
Data with a length of 2 bytes are interpreted in the PLC-conformal byte order for words.
Data with a length of 4 bytes are interpreted in the PLC-conformal byte order for long words.

5.2.4.1 Data Types

Direct accessing of the following data types is possible:
The data types listed below can be accessed in bit, byte or word-mode. The access modes are distinguished by the abbreviations BY, B and W.

I	input
O	output
M	flag

The data types listed below can be accessed in word-mode only.

D	data word	(0 ... 510)
DB	data buffer	(0 ... 510, only with ZE300)

T	timer	(R300 and R600 read-only)
C	counter	(R300 and R600 read-only)

The size of each data area is governed by the CPU of the PLC.

5.2.5 Additional Functions

In addition to the random write and read access to PLC variables, a memory area comprising 12 bytes is specified in the mask definition as poll area. The location of this memory area is specified in the mask definition.

Only marginal conditions regarding this memory area are that the PLC must be able to access in bit-mode and the terminal in **word-mode** and the memory area must be contiguous.

The addresses M, D or DB can be accessed in word-mode.

The data area comprises a maximum of 6 words or 12 bytes.

Example: The cyclic data area is set to DW21 in the TesiMod programming system

Word address	**DW**	**High byte**	**Low byte**
WORD address +0	DW21	Write coordination byte	Reserved
WORD address +1	DW22	Message channel high-byte	Message channel low-byte
WORD address +2	DW23	Function keys LED1...4	LED5...8
WORD address +3	DW24	Function keys LED9...12	LED13...16
WORD address +4	DW25	Function keys LED17...20	LED21...24
WORD address +5	DW26	Function keys LED25...28	LED29...32

5.2.6 Physical Interfacing

TesiMod 20mA Current Loop - Bosch PU-Interfacing

Pinning of the TesiMod 20mA current loop

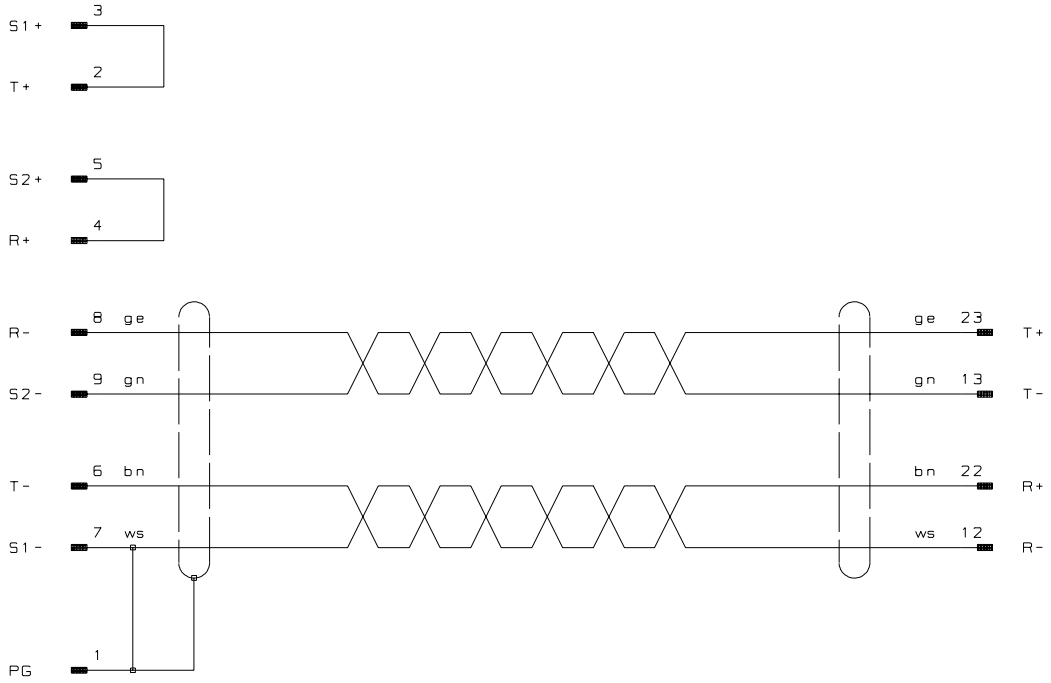
Pin	Designation	Function
1	Shield	Shield
2	T+	Transmit Data, Positive Polarity
3	S1+	Power Source 2, Positive Polarity
4	R+	Receive Data, Positive Polarity
5	R-	Receive Data, Negative Polarity
6	S2+	Power Source 1, Positive Polarity
7	T-	Transmit Data, Negative Polarity
8	S1-	Power Sink 1, Negative Polarity
9	S2-	Power Sink 2, Negative Polarity

Pinning of the TesiMod Universal Interface TTY / 20mA current loop

Pin	Designation	Channel	Function
10	T+	SER1	Transmit Data, Positive Polarity
12	S1+	SER1	Power Source 2, Positive Polarity
13	R+	SER1	Receive Data, Positive Polarity
14	R-	SER1	Receive Data, Negative Polarity
16	S2+	SER1	Power Source 1, Positive Polarity
19	T-	SER1	Transmit Data, Negative Polarity
21	S1-	SER1	Power Sink 1, Negative Polarity
24	S2-	SER1	Power Sink 2, Negative Polarity

5.2.6.1 Connecting Cable TTY / 20 mA - Bosch PU

TesiMod	Bosch PLC
Operating Terminal	e.g. ZE301
Sender aktive	Sender passive
Receiver aktive	Receiver passive

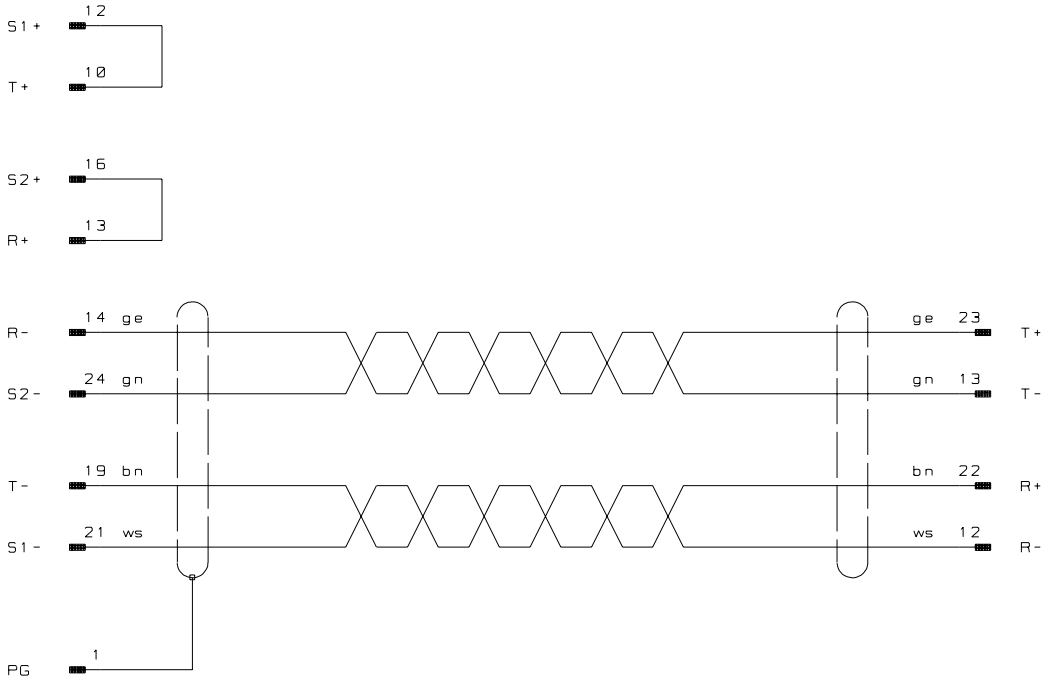


D-Subminiature	D-Subminiature
Male connector	Male connector
9 pin	25 pin

The shield is connected to the metal housing on both sides.
At the side of the operating terminal pin 1 and pin7 are connected to the shield / metal housing.

5.2.6.2 Connecting Cable Universal Interface TTY / 20 mA - Bosch PU

TesiMod	Bosch PLC
Operating Terminal	e.g. ZE301
Sender aktive	Sender passive
Receiver aktive	Receiver passive



D-Subminiature	D-Subminiature
Male connector	Male connector
25 pin	25 pin

The shield is connected to the metal housing on both sides.
At the side of the operating terminal pin 1 and pin7 are connected to the shield / metal housing.

/000-0108/
Bosch_T_eng_V13_3005020QK0

5.2.7 Error Messages

Code	1	E_SLAVE_NOT_READY	Slave not ready
	2	E_PROTOCOL	Sequence of the packets
	3	E_FRAME	Character frame error
	4	E_TIMEOUT	Timeout error
	5	E_CRC_BCC	CRC error
	6	E_PARITY	Parity error
	7	E_SEND_ABORT	Abort send process
	8	R_REC_ABORT	Abort receive process
	9	E_BUF_SIZE	Insufficient cyclic buffer
	10	E_NO_DEFINE	No cyclic data defined
	12	E_DEFINE	Cyclic data already defined
	15	E_NO_PROTOCOL	Selected protocol is not supported
	16	E_OVERRUN	Receive buffer overrun
	40	E_SYS_ADDRESS	Undefined system variable

Bosch-specific error message

50	E_QUITTUNG_START	No communication set-up
51	E_QUITTUNG_OPEN	Incorrect acknowledge signal during communication set-up
52	E_QUITTUNG_DATA	Incorrect acknowledge signal to transmitted information block
53	E_NO_RESPONSE_WRONG_CHAR	No response message
54	E_TIMEOUT_NO_RESPONSE	Timeout - no response message
55	E_TIMEOUT_BLOCKZEIT	Timeout - block time has been exceeded
56	E_TIMEOUT_QUIT_RESPONSE	Timeout - no acknowledge signal
57	E_ABBRUCH_SPS	EOT -PLC abort
58	E_RECEIVE_COUNT	Number of received data is incorrect

PLC-Error

62 E_WRONG_ORDER
No. 32 from the PLC ..... write access to T, C, to module not permitted

67 E_WRONG_PARAMETER
No. 37 from the PLC ..... incorrect parameter

68 E_CHAR_COUNT
No. 38 from the PLC ..... number of bytes received is incorrect accordance with the message header

69 E_SYSTEM
No. 39 from the PLC ..... incorrect P1 in the system message

71 E_DIRECTION
No. 41 from the PLC ..... direction not defined

72 E_DB_SHORT
No. 42 from the PLC ..... DB too small

74 E_DB_NOT_PROG
No. 44 from the PLC ..... DB not programmed

76 E_DB_NOT_DEF
No. 46 from the PLC ..... DB not defined

78 E_WRONG_TYP
No. 48 from the PLC ..... block type unknown

79 E_P2_NULL
No. 49 from the PLC ..... parameter 2 is 0

94 E_TELE_TYP
No. 64 from the PLC ..... message type incorrect

Table of Contents

5.3 TesiMod - DIN-Meßbus

TesiMod operating terminals allow for a simple interfacing to process computers or PLCs via the DIN-Meßbus (DIN 66348 - part 2).
The DIN-Meßbus has a master-slave type structure, i.e. the bus master polls the connected bus slaves. Thus, within this description a distinction is made between the master and the slave interfacing.

5.3.1 TesiMod - DIN-Meßbus-Master

5.3.1.1 Process Computer as Bus Master

Any process computer complying with the following marginal conditions is suitable as bus master:

- Communication in accordance with DIN66348 - part 2 (specifies the data transmission layers 1 and 2 of the ISO/OSI - layer model)

- Interpretation of the data contents in compliance with the SÜTRON TesiBus specification

The services (DIN-Meßbus-user data contents) are defined in an address-transparent manner. A number of different address formats are available for the interfacing. A SÜTRON-internal address format is used for indirect PLC interface connections whereas a linear, byte-structured memory addressing technique with an address area of 2^{24} bits is implemented for regular process computer interface connections.

5.3.1.2 Gateway as Bus Master

The DIN-Meßbus allows a number of operating terminals to be connected to a PLC (via the programming unit interface) which is particularly interesting in applications where the PLC-interface itself represents only a point-to-point connection.

This requires the use of a gateway which is responsible for the adaptation of the protocol between the DIN-Meßbus (TesiBus) and the respective PLC-protocol.

At the same time the gateway will function as the bus master on the DIN-Meßbus and poll the slave terminals at cyclic intervals.

The gateway offers all functions of a full-size terminal, however, due to the implementation of the DIN-Meßbus protocol on interface X3 the following functions are not available:

- Printing of messages
- Printing of protocols
- Printing of PCX-files

The gateway offers the following additional functions, which are not available in the standard TesiMod operating terminals:

- Extended poll area
- Cache function for read-only data
- Image of the network status

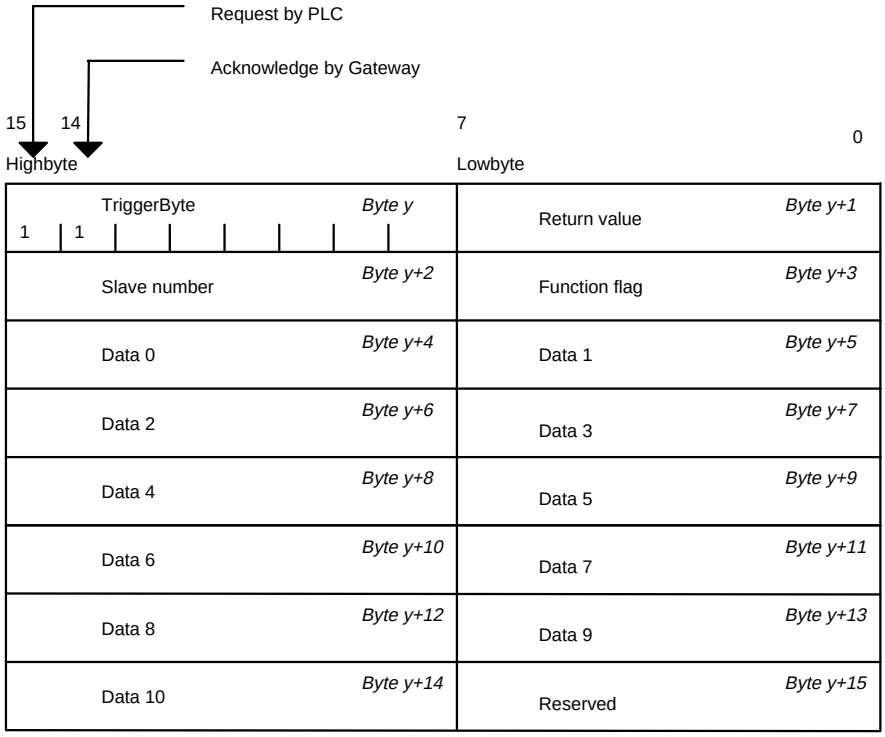
5.3.1.3 Poll Area

The cyclic poll area makes the following functions available on the master terminal or slave terminal, respectively:

- Write Coordination byte (WCB)
- Serial message channel
- LEDs of the function keys

The functionality has been increased in comparison with the standard poll area of the direct PLC-interface connection. The mapping of this poll area to the functions of the terminal does not occur in a static fashion but is rather transmitted via the trigger byte in an event-controlled and slave-specific manner. I.e. the controller enters values into the poll area and transfers it to the gateway via bit 7 (value 0x80) of the trigger tyte. After the function has been executed, the gateway will write the return value and the code 0x40 back into the trigger byte. Subsequently, the data area will again be under the control of the PLC.

The poll area is classified as either byte-structured or word-structured depending on the specified address and has the following fixed structure:



Byte structure y

The functions of the bytes are as listed below:

Slave number:

0	Data are intended for the gateway itself
1...31	Slave number to which the event will be transmitted by order-only

Function code:

1	Transmit new WCB
2	Transmit message
3	Activate/deactivate LEDs

Return value:

0	Ok-acknowledge signal
1	Slave is not ready to receive
2	General DIN-Meßbus error
3	Invalid function code
4	Slave not synchronized

Data contents:

Function *transmit new Write Coordination Byte (WCB)*:

Data 0	—>	WCB
Data 1	—>	free

Function *transmit message*:

Data 0	—>	Message number Highbyte
Data 1	—>	Message number Lowbyte

Funtion *activate LEDs*:

Data 0	—>	LED 1...4
Data 1	—>	LED 5...8
Data 2	—>	LED 9...12
Data 3	—>	LED 13...16
Data 4	—>	LED 17...20
Data 5	—>	LED 21...24
Data 6	—>	LED 25...28
Data 7	—>	LED 29...32
Data 8	—>	LED 33...36
Data 9	—>	LED 37...40
Data 10	—>	LED 41...44

5.3.1.4 Cache Function for Read-Only Data

The cache function will poll a memory area of up to 62 bytes in the PLC and will then broadcast the information to all connected slave-terminals at cyclic intervals . This greatly reduces the load on the bus and allows simultaneous transmission of all data to the terminals.

Since the broadcast service on the communications layer 2 is not subject to monitoring, the slave-terminals implement a definable timeout to monitor the receipt of the broadcasted data packets.

If a slave-terminal has a read request which is located within the cache area, no communication will be carried out via the gateway to the PLC, but the data are rather copied locally from of the terminal cache.

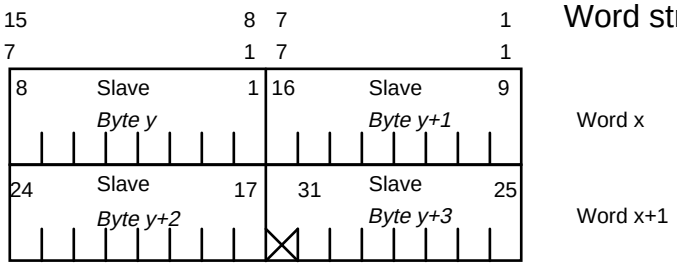
Variables which have been assigned the attribute *display once* and which are located within the cache area, will be displayed once more after receipt of the next cache packet. The gateway transmits the cache data in an equidistant fashion thus permitting the execution of a high-priority and speedy output which is independent of the general cyclic output cycle.

5.3.1.5 Status of the Network

A 4-byte area is used by the gateway to transmit the status of the network to the PLC. Each slave which is detected in the network and which is synchronized, is represented by a bit which is set to "1".

Please note carefully:
During the synchronization of a slave (i.e. its bit is set from "0" —> "1"), **the external data release for this slave will be reset and the function keys for this slave will be deactivated** (the write coordination byte assumes the value "0"). In such an event, the PLC might be required to put the terminal back into the appropriate status via the poll area!

The figure below illustrates the general structure of the 4-byte area which is classified as either byte-structured or word-structured depending on the specified address:



5.3.1.6 Parameters Interface SER1 for PLC-Interfacing

The communication via interface SER1 is effected by the same communications protocol through the same loadable driver as is used for the communication via a comparable direct PLC-interface connections.

The respective interface parameters are listed in the corresponding section of chapter 5.

5.3.1.7 Parameters Interface SER2 for DIN-Meßbus - Master

The DIN-Meßbus - master interface connection requires setting of the following parameters for the interface SER2:

Baud rate:	**19200** Baud
Parity:	**even**
Data length:	**7 bits**
Stop bits:	**1 stop bit**
Handshake:	**no handshake**

Note: It is also possible to select a smaller baud rate. The baud rate setting for the gateway and the setting for the slave terminal must be in agreement.

5.3.1.7.1 Minimal Slave Number

This value specifies the lower limit for the slave numbers which the network will be scanned for.
Range of values: 1...31

5.3.1.7.2 Maximal Slave Number

This value specifies the upper limit for the slave numbers which the network will be scanned for. The value must be greater than or be equal to the minimal slave number (see 5.3.1.7.1). Keeping the number of non-existing slaves within the range of values to a minimum, will contribute to a more speedy synchronization of the slaves by the gateway.
Range of values: 1...31

5.3.1.7.3 Test Cycle for the Synchronization

Specifies the period of time that is allowed to elapse between each test communication when scanning the bus. The input must be equal to a multiple of 1/10 second.
Range of values: 2...255.

5.3.1.7.4 Cache Size

Specifies **in bytes** the size of the read-only data cache. Defining a word-structured area as cache address will require a cache size comprising an **even number** of bytes!
Range of values: 0...62 bytes.

5.3.1.7.5 Cache Address

Specifies the cache address, i.e. the beginning of the read-only data cache. No address must be specified here when operating the bus without the cache function.

The following address levels are possible as starting addresses :

- Flag byte / flag word
- Data word

If variables from other address types are to be cached, it will be necessary to copy them from the PLC to the cache address area.

5.3.1.7.6 Intervals for the Cache Update

This time period specifies the length of time in 1/10 second that is allowed to elapse between the cache update cycles.
Range of values: 2...255

5.3.1.7.7 Network Status Address

Specifies the address to which a 4-byte sequence, representing the network status, is transmitted to the PLC. All commonly used PLC addresses are valid here. No address must be defined, if the network status is not to be transmitted to the PLC.

5.3.1.8 Additional Error Messages

Since the gateway represents a full-size terminal, all error messages which might be generated with the associated direct PLC-interfacing, could be generated here as well. For details on these messages see the respective sub-chapter 5.

The following additional messages might be generated for the gateway:

Code	41	E_CACHE_READ	Error during read-access to the PLC-cache
	42	E_EGB_READ	Error during read-access to the gateway-poll area
	43	E_EGB_WRITE	Error during write-access to the gateway-poll area
	44	E_NETZ_ADDRESS	The address-syntax (variable definition) of one of the slave-terminals does not match the PLC-protocol of the gateway.
	45	E_NETZ_WRITE	Error during transmission of the network status to the PLC.
	46	E_NO_GATEWAY_PARA	No gateway parameters are available. If necessary, store and compile files once more using a new TS programming software.

In the event of these error messages, the sub-code will always supply the error numbers of the respective communications protocol (see error numbers of the respective direct PLC-interfacing in the respective chapter).

5.3.2 TesiMod DIN-Meßbus - Slave

5.3.2.1 General Information

TesiMod operating terminals allow for a simple networking via the DIN-Meßbus regardless of the bus master being used. Every slave-terminal in the network has access to the data of the bus master.

Connecting the slave-terminals to a PLC by means of a gateway , allows the masks and variable definitions to be created in accordance with the same procedure as is used with the direct PLC-interfacing. It is merely necessary to adapt the interface parameters to the DIN-Meßbus.

This offers the user of TesiMod operating terminals the following advantages:

- Random write and read access to any data within the PLC. Data of existing PLC programs can be displayed and modified directly in the operating terminal. It is not necessary to adapt the PLC program in any respect since it is not required that communications data be stored in a specified address area or data type area.

- The transmission of messages, the write coordination byte as well as the status of the function key LEDs is effected by the bus master in the event-controlled mode.

- No configuration required within the PLC.

- The protocol provides error control. Transmission errors are detected and, if possible, eliminated by repeating the transmission. A noise-immune interface hardware in accordance with the RS485 interface standard permits the application even in a harsh industrial environment.

- Interface parameters are stored in the mask definition. Altering of the parameters is also possible in the setup mask of the terminal at any time.

- The graphical operator guidance offers the user a maximum of assistance in defining the variable addressses within the mask definition. The definitions (abbreviations) used here are identical with the definitions used within a PLC program (e.g. F3 = flag 3).

5.3.2.2 Technical Description

The bus master (process computer or gateway to the PLC) is connected to the slave through the DIN-Meßbus (DIN 66348).

The services (user data of the DIN-Meßbus protocol) required for the exchange of data are defined in the TesiBus specification.

To reduce the load on the bus, it is possible for each slave-terminal to be equipped with its own

read-only data cache. The address location as well as the size of this cache will be determined by way of negotiation between bus master (gateway) and slave-terminal during the synchronization process. The bus master broadcasts the cache data to each slave-terminal in an equidistant fashion. The cache data will then be stored locally by each slave-terminal.

If a slave-terminal has a read request which is located within the cache area, no communication will be carried out via the gateway to the PLC, but the data are rather copied locally from the terminal cache.

Variables which have been assigned the attribute display once and which are located within the cache area, will be displayed once more after receipt of the next cache packet. The gateway transmits the cache data in an equidistant fashion thus permitting the execution of a high-priority and speedy output which is independent of the general cyclic output cycle.

5.3.2.3 Parameters Interface SER1 for DIN-Meßbus -Slave

The DIN-Meßbus - slave interfacing requires setting of the following parameters for the interface SER1:

Baud rate:	**19200** Baud
Parity:	**even**
Data length:	**7 bits**
Stop bits:	**1 stop bit**
Handshake:	**no handshake**

Note: It is also possible to select a smaller baud rate. The baud rate setting for the gateway and the setting for the slave terminal must be in agreement.

5.3.2.3.1 Timeout for Order-Reply

Whenever the slave requires data from the controller, it will start a timer. This timer allows monitoring of the following two processes: whether the slave-terminal is polled by the master and whether the response is received from the bus master within this period of time. Any value within the range of 0...65000 milliseconds can be defined for the timeout. Specify a value of 0 if you wish to work without the function timeout monitoring. The absolute time value depends on the number of stations in the network, it should, however, be equal to a value of approximately 2000 to 5000 milliseconds.

5.3.2.3.2 Timeout for Cache-Update

Since the receiving station can not acknowledge the broadcast service, the slave-terminal employs a timeout to monitor the point of time at which broadcasted data were received. This will ensure that local cache data are not “of any given point of time in the past”. Any value within the range of 0...65000 milliseconds can be defined for the timeout. The value depends on the cache-update interval of the bus master (see gateway parameters). The default value should correspond to a value of 5000 milliseconds.

5.3.2.3.3 Slave Number

Allows defining of the slave number of the respective slave-terminal. To be able to load the same mask definition into every slave-terminal, an invalid slave number can be entered here. Subsequently, when connecting the terminal to the network, a correct slave number must be entered into the setup mask with the aid of the system variable SYSCOMSLAVENR. Valid slave numbers range from 1...31. Any invalid values will be set to 255 (OxFF) during the terminal startup.

5.3.2.4 Data Types

In general, all data types available for the comparable direct PLC-interfacing are available here as well.

5.3.2.5 Additional Functions

The transmission of the functions:

- Messages
- Write coordination byte
- LEDs of the function keys

is effected by the bus master (gateway) in an event-controlled mode.

5.3.2.6 Physical Interfacing

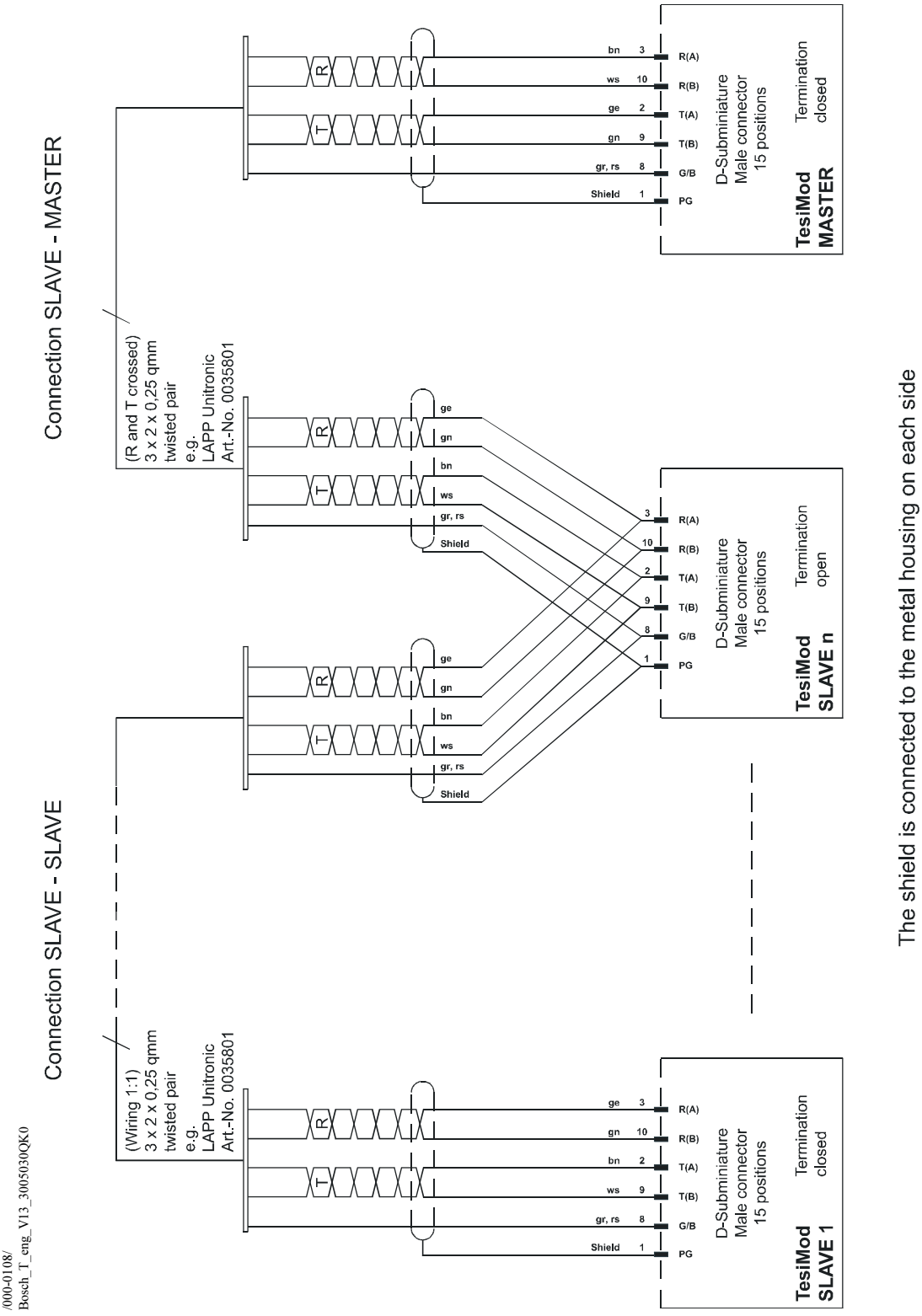
TesiBus - Pinning

Slave-terminals			Gateway	
1	Shield	<———>	1	Shield
2	**TD+**	<———>	**3**	**RD+**
3	**RD+**	<———>	**2**	**TD+**
4	RTS+		4	RTS+
5	CTS+		5	CTS+
6	TxC-		6	TxC-
7	RxC-		7	RxC-
8	SGND	<———>	8	SGND
9	**TD-**	<———>	**10**	**RD-**
10	**RD-**	<———>	**9**	**TD-**
11	RTS-		11	RTS-
12	CTS-		12	CTS-
13	TxC+		13	TxC+
14	RxC+		14	RxC+
15	n.c.		15	n.c.

The following applies:

- all slave-terminals in the network are connected in parallel
- the TD connector of the master must be connected to the RD connector of the slave and vice versa.
- the termination at the first and the last of the terminals in the network must be activated.

5.3.2.7 Bus Cable Gateway - Slave - Terminals



/000-0108/
Bosch_T_eng_V13_3005030QK0

5.3.2.8 Error Messages

Code	30	E_SLAVE_NAK	Even after a repeated attempt, slave was unable to transmit transmission data to master (NAK from the master)
	31	E_SLAVE_TIMEOUT	Even after a repeated attempt, slave was unable to transmit transmission data to master (TA has elapsed)
	32	E_SEND_ORDER_TIMEOUT	Slave was unable to transmit its order within the timeout.
	33	E_ORDER_REPLY_TIMEOUT	Slave has not received the reply to the transmitted order within the timeout.
	34	E_CACHE_TIMEOUT	Slave has not received the data broadcasted by the cache within the timeout.

Table of Contents

5.4 TesiMod - Bosch Interfacing via BUEP19E

5.4.1 General Information

TesiMod operating terminals allow for a simple connection to the Bosch PLCs thus making TesiMod operating terminals the perfect man-machine-interface for your Bosch PLC.
The TesiMod operating terminal is connected to the respective PLC-module.

The data communication on the interface is handled by the PU protocol BUEP19E. With the PLCs CL150, CL200, CL350, CL400, CL500 the communication is possible with the BUEP19E protocol.

The hardware and software components of the TesiMod system are fully adapted to the parameters and marginal conditions of the interface.

This offers the user of TesiMod operating terminals the following advantages:

- Random write and read access to any data within the PLC by a maximum of 4 different ZS500-modules. Data of existing PLC programs can be displayed and modified directly in the operating terminal. It is not necessary to adapt the PLC program to the operating terminal in any respect since it is not required that communications data be stored in a specified address area or data type area.

- The TesiMod operating terminal automatically polls cyclic data in the cyclic poll area. The size and location of this poll area are custom assignable.

- No configuration required within the PLC.

- The PU protocol is handled entirely by the firmware of the PLC. A PLC program (function blocks, etc.) in the PLC is not required for the handling of the communication.

- The protocol provides error control. Transmission errors are detected and, if possible, eliminated by repeating the transmission. An electrically isolated, noise-immune interface hardware in accordance with the 20mA current loop interface standard permits the application even in a harsh industrial environment.

- The parameters of the communications interface X2 are assigned in the TS programming software in a protocol-specific manner and are stored in the mask definition. Modifying of the parameters is also possible in the set-up mask or any other I/O mask of the terminal.

- The graphical operator guidance offers the user a maximum of assistance in defining the variable addresses within the list of variables. The definitions (abbreviations) used here are identical with the definitions used within the PLC program.

5.4.2 Technical Description

The interfacing of the TesiMod operating terminal to the Bosch PLCs is effected by means of the BUEP19E protocol.

The protocol BUEP19E allows random read and write access to all PLC data. The size of the address area depends on the respective PLC.

The following byte order applies to the access in word-mode: **HighByte-LowByte**.

5.4.3 Parameters Interface SER

The TesiMod operating terminal adapts to the default parameters of the interface. It is therefore **not** necessary to modify the interface parameters in the PLC.

Baud rate:	**9600** Baud
Parity:	**even**
Data length:	**8**
Stopbits:	**1**
Handshake:	**no handshake**

5.4.3.1 Protocol Parameter Target Module

As a target module you can choose between CL500, CL400 or CL200.
The CL200 must be selected if the CL150 is used.

5.4.3.2 Protocol Parameter Block Check

As a default, the protocol BUEP19E implements the data block check method CRC16.
The programming device with a EP/AG module, however, uses a data block check method in accordance with LRC8.
To avoid difficulties during the development phase during which the PU and the terminal are alternately connected to the PLC, the terminal allows selection of the block check method LRC8.

5.4.3.3 Protocol Parameter Coordination Flag

The protocol permits defining of a field coordination flag and a sequence coordination flag.

5.4.4 Data Type Structure

a) Alphanumerical Text
Is stored in the memory byte for byte in ascending address order.

b) Counter
The count value is interpreted in binary format. The maximum value is 8191.

c) Timer
Timer functions consist of a time value and a time base. The terminal operates with imaginary unsigned 4-byte variables, even though the data stored in the PLC comprise only 2 bytes.

When read-accessing the timer, the terminal converts the time value and time base into a terminal-internal unsigned 4-byte number, which represents the time value in reference to the time base of 0.01 second.
Ex.: A range of 10 (time base is 1.0 second) and a time value of 999 , are represented or edited, respectively, in the terminal by the value 99900. Scaling of this value to other value ranges is possible by specifying a factor and divisor within the variable definition.
Before writing a timer variable to the PLC, the time value and the smallest possible time base are formed from the terminal-internal unsigned 4-byte value.

d) Floating Point Number
The data with a length of 4 bytes are interpreted in the IEEE floating point format.

e) Binary Variables with a Length of 1, 2 or 4 Bytes
Data with a length of 2 bytes are interpreted in the PLC-conformal byte order for words.
Data with a length of 4 bytes are interpreted in the PLC-conformal byte order for long words.

5.4.4.1 Data Types

Direct accessing of the following data types is possible:
The data types listed below can be accessed in bit, byte or word-mode. The access modes are distinguished by the abbreviations BY, B and W.
The size of the data area depends on the PLC CPU.

Data type				**Access**
T	Timer	(0 ... 127)	Timer no.	W
Z	Counter	(0 ... 127)	Counter no.	W
D	Data block	(0 ... 511)	Byte-addr.	W,By
DB	Data buffer	(0 ... 511)	Byte-addr.	W, By
M	Flag	(0 ... 255)	Byte-addr.	W, By, B
E	Input	(0 ... 63)	Byte-addr.	W, By, B
A	Output	(0 ... 63)	Byte-addr.	W, By, B
BZ	Status of the PLC	RUN/STOP	CL500/400	By
BZ	Status of the PLC	RUN/STOP	CL200	W
DF	Data field	(0 ... 24575)	Byte-addr.	W, By

If the size of of the data field is defined as a linear area the data field number must be set to 255.

5.4.5 Additional Functions

In addition to the random write and read access to PLC variables, a memory area comprising 12 bytes is specified in the mask definition as poll area. The location of this memory area can be specified in the mask definition.

Only marginal conditions regarding this memory area are that the PLC must be able to access in bit-mode and the terminal in word-mode and the memory area must be contiguous.

The data area comprises a maximum of 6 words or 12 bytes.
Example: The cyclic data area is set to DW21 in the TesiMod programming system

WORD address	DW	High byte	Low byte
WORD address +0	DW21	Write coordination byte	Reserved
WORD address +1	DW22	Message channel high-byte	Message channel low-byte
WORD address +2	DW23	Function keys LED1...4	LED5...8
WORD address +3	DW24	Function keys LED9...12	LED13...16
WORD address +4	DW25	Function keys LED17...20	LED21...24
WORD address +5	DW26	Function keys LED25...28	LED29...32

5.4.6 Physical Interfacing

TesiMod 20mA Current Loop - Bosch PLC-Interfacing

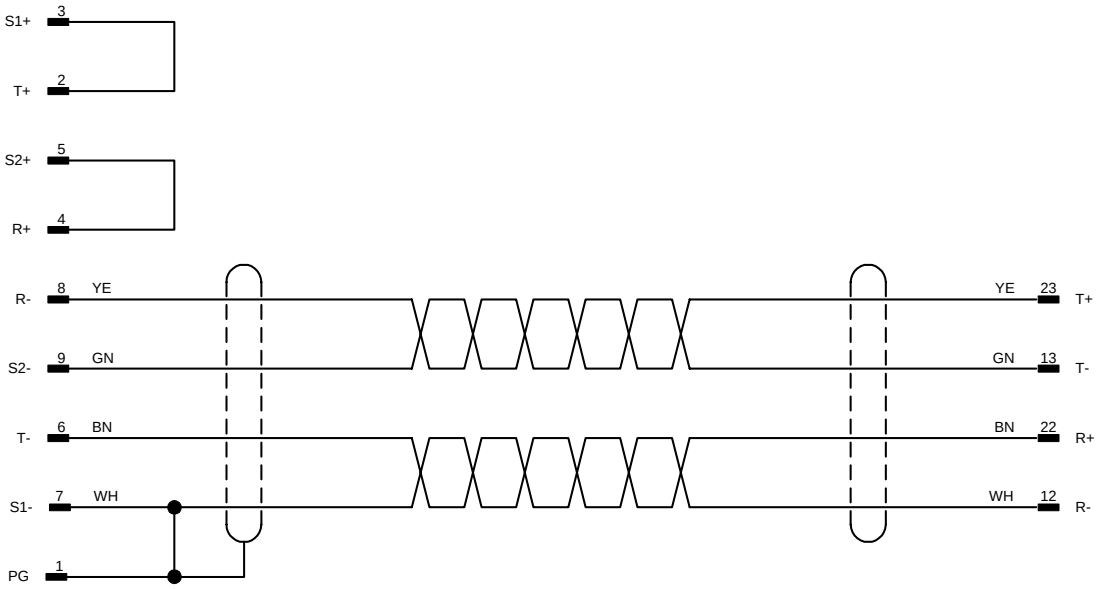
Pin	Designation	Function
1	Shield	Shield
2	T+	Transmit Data, Positive Polarity
3	S1+	Power Source 2, Positive Polarity
4	R+	Receive Data, Positive Polarity
5	R-	Receive Data, Negative Polarity
6	S2+	Power Source 1, Positive Polarity
7	T-	Transmit Data, Negative Polarity
8	S1-	Power Sink 1, Negative Polarity
9	S2-	Power Sink 2, Negative Polarity

Pin assignment TesiMod Universal Interface TTY / 20 mA Current Loop

Pin	Designation	Channel	Function
10	T+	SER1	Transmit Data, Positive Polarity
12	S1+	SER1	Power Source 2, Positive Polarity
13	R+	SER1	Receive Data, Positive Polarity
14	R-	SER1	Receive Data, Negative Polarity
16	S2+	SER1	Power Source 1, Positive Polarity
19	T-	SER1	Transmit Data, Negative Polarity
21	S1-	SER1	Power Sink 1, Negative Polarity
24	S2-	SER1	Power Sink 2, Negative Polarity

5.4.6.1 Connecting Cable TTY / 20 mA - Bosch PLC

TesiMod	Bosch PLC, without CL150
Operating Terminal	PG-Interface X31
Sender active	Sender passive
Receiver active	Receiver passive



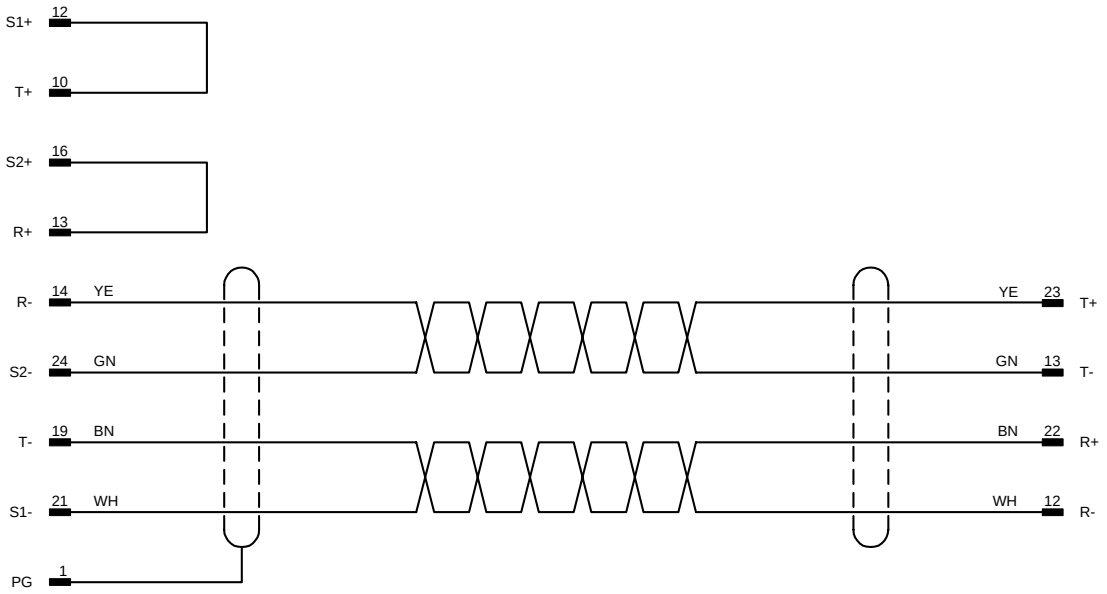
D-Subminiature	D-Subminiature
Male connector	Male connector
9 pin	25 pin

The shield is connected to the shield / metal housing on both sides.

/001-0185/
Bosch_T_eng_V13_300504AQK0

5.4.6.2 Connecting Cable Universal Interface TTY / 20 mA - Bosch PLC

TesiMod	Bosch PLC, without CL150
Operating Terminal	PG-Interface X31
Sender active	Sender passive
Receiver active	Receiver passive



D-Subminiature	D-Subminiature
Male connector	Male connector
25 pin	25 pin

The shield is connected to the shield / metal housing on both sides.

5.4.6.3 Connecting Cable Universal Interface RS232 - Bosch CL150

TesiMod
Operating Terminal BT2/BT5N/BT20N

Bosch PLC CL150/CL151
PG-Interface X31 (V24)

D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
9 pin

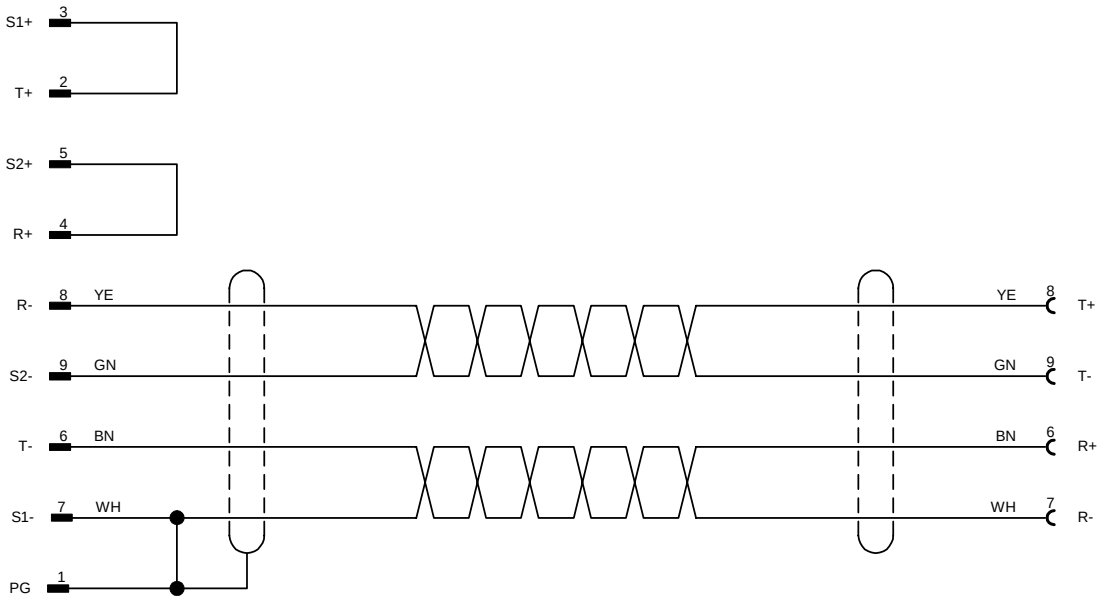
The shield is connected to the shield / metal housing on both sides.

/001-0185/
Bosch_T_eng_V13_300504AQK0

5.4.6.4 Connecting Cable TTY / 20 mA - Bosch CL151

TesiMod
Operating Terminal BT20
Sender active
Receiver active

Bosch PLC CL151
Interface X32 (20mA)
Sender passive
Receiver passive



D-Subminiature
Male connector
9 pin

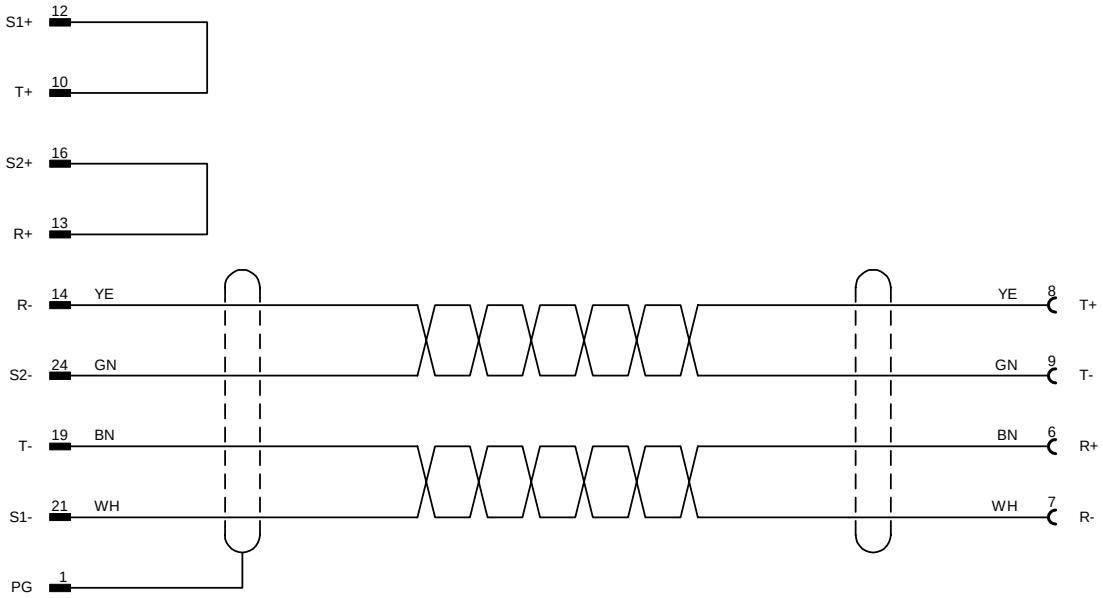
D-Subminiature
Male connector
9 pin

The shield is connected to the shield / metal housing on both sides.

5.4.6.5 Connecting Cable Universal Interface TTY / 20 mA - Bosch CL151

TesiMod
Operating Terminal BT2/BT5N/BT5/BT20N
Sender active
Receiver active

Bosch PLC CL151
Interface X32 (20mA)
Sender passive
Receiver passive



D-Subminiature
Male connector
25 pin

D-Subminiature
Male connector
9 pin

The shield is connected to the shield / metal housing on both sides.

/001-0185/
Bosch_T_eng_V13_300504AQK0

5.4.7 Error Messages

Code 0			
Subcode			
	0	NO_ERROR	Error-free processing
Code 1			
Subcode			
	1	E_SLAVE_NOT_READY	Error originating in level 1 and level 2
	2	E_PROTOCOL	Sequence of the packets
	3	E_FRAME	Character frame error
	4	E_TIMEOUT	Timeout error
	5	E_CRC_BCC	CRC error
	6	E_PARITY	Parity error
	7	E_SEND_ABORT	Abort send process
	8	R_REC_ABORT	Abort receive process
	9	E_BUF_SIZE	Insufficient cyclic buffer
	10	E_NO_DEFINE	No cyclic data defined
	12	E_DEFINE	Cyclic data already defined
	15	E_NO_PROTOCOL	Selected protocol is not supported
	16	E_OVERRUN	Receive buffer overrun
	40	E_SYS_ADDRESS	Illegal system variable

Bosch-specific error message

Code	50	E_QUITTUNG_START	No communication set-up
	51	E_QUITTUNG_OPEN	Incorrect acknowledge signal during communication set-up
	52	E_QUITTUNG_DATA	Incorrect acknowledge signal to transmitted information block
	53	E_NO_RESPONSE_WRONG_CHAR	No response message
	54	E_TIMEOUT_NO_RESPONSE	Timeout - no response message
	55	E_TIMEOUT_BLOCKZEIT	Timeout - block time period has been exceeded
	56	E_TIMEOUT_QUIT_RESPONSE	Timeout - no acknowledge signal
	57	E_ABBRUCH_SPS	EOT -PLC abort

Code 2

Subcode

58 E_RECEIVE_COUNT ..... Number of received data is incorrect

Possible source of error:
Check if in the mask where the error occurs a variable with odd number of bytes will be read by a word or doubleword address.

See module manual

Code 3

Subcode

1 The requested module is not available
16 Module is not accessible
35 The access to this address field is not allowed
36 The address field is secured by a user
37 Timer must not be written to
38 Module number to high
39 Module not available
40 Module is to small

147 Flag area (only CL200/CL150) overrange.
Possible source of error:
- Flag area defined out of M0 - M191 (CL200)
- Flag area defined out of M0 - M151 (CL150)

Code 4

Subcode

32 The requested data type (CommandCode) is not known by the PST
33 Protocol flag not known by the PST

35 The given co-ordination flag is not known by the PST

37 Parameter identifier in the telegram doesn´t fit to the specified parameters
38 Length of block and topic number of data are different

40 Type of telegram unknown
41 Type of command unknown

58 Start address doesn´t fit to the type of operand (Word at odd address)
Possible source of error:
- damaged R500 module

59 Start address defined outside the address area
60 Invalid parameter for specified command
61 Invalid type of operand

64 The PST hasn´t received an identification telegram, yet

99 The given length of data is greater than the requested data area

210 Co-ordination flag is locked

Table of Contents

5.5 TesiMod - Profibus-DP - Interfacing

5.5.1 Technical Description

TesiMod operating terminals allow for a simple connection to the Profibus-DP thus making TesiMod operating terminals the perfect man-machine-interface for systems using the Profibus-DP. Moreover it is possible to connect multiple operating terminals to one master controller.
The speed-optimized PROFIBUS-DP is a variant of PROFIBUS that has been particularly tailored to suit the communication between automation systems and decentral peripherals.

The Profibus-DP implemented in the operating terminal complies with the standard DIN 19245 part 1 and part 3 as well as the European field bus standard EN 50170.

The TesiMod operating terminal complies with the conditions specified by the standards thus allowing its trouble-free integration into the Profibus-DP as a slave.

An additional hardware is used to connect the operating terminal to the Profibus.

Located on this hardware is the ASIC SPC3 which handles the entire PROFIBUS-DP protocol, thus allowing transmission speeds of up to 12 MBaud.

The operating terminal is employed on the bus as if it were a decentral module with up to 32 inputs and outputs and therefore occupies between 8 and 32 bytes IN-data and between 8 and 32 bytes OUT-data. Via the bus, the input and output statuses are exchanged between the master and operating terminal at cyclic intervals. The Profibus-DP protocol is a manufacturer or controller-neutral data transmission protocol. Because no hardware input and output statuses are transmitted in the operating terminal, a data profile must be specified between master and slave which allows the corresponding partner to identify the type of the data that have been transmitted.

All services required for the operation of the operating terminal are initiated by the operating terminal. The operating terminal has a client functionality.
The controller simply reacts to requests from the operating terminal. The controller has a server-functionality.
If the operating terminal is integrated into the Profibus-DP, the master module needs to interpret the arriving data in accordance with the specified profile and also needs to respond in accordance with the profile.
This is generally accomplished by a FB (function block) in the controller, which is capable of reading the request from the IN-data and of writing a response to the OUT-data.

5.5.2 Specification within the Profibus-DP

The specification of the operating terminal in the PROFIBUS-DP is determined by the supplied device-master-data-file SUET081A.GSD (see 5.5.11).

Telegram Length
The data length can be between 8 and 32 bytes or 4 and 16 words respectively. A fixed setting

of this length can be specified by means of the TesiMod programming software and must be set in the master accordingly.

Station Address
The station address can be preset by means of the TesiMod programming software or can be set via the operating terminal to a value between 3 and 124.

Baud Rate
The transmission rate is automatically detected. The maximum rate is 12 MBaud.

Diagnosis
The station-oriented diagnosis is implemented in the operating terminal.
Five bytes of user diagnosis data are transmitted

- 1st byte — Error number
  1 = Communication Error at OperatingTerminal
- 2nd and 3rd byte — Communication Error Code
- 4th and 5th byte — Communication Error Subcode

Communication Code and Subcode are values that are also displayed at the operating terminal.

5.5.3 Data Profile

To allow a direct data access to the various data areas within a controller, a data profile must be determined between master and slave.

The first 4 bytes of the specified telegram length are used as follows:
- Control and length of the telegram,
- definition of the access and
- definition of the data area

5.5.3.1 Structure of the Profile

Request-Telegram

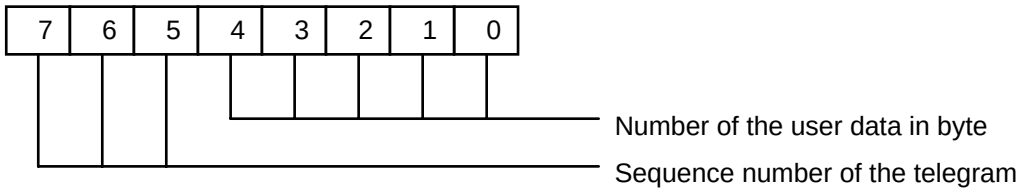
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6		Byte n
Number User Data	Access	Offset Low	Offset High	User Data 1st Byte	User Data 2nd Byte		User Data nth Byte

Response-Telegram

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6		Byte n
Number User Data	Access	0x00	Returncode Error	User Data 1st Byte	User Data 2nd Byte		User Data nth Byte

5.5.3.2 Control Bytes

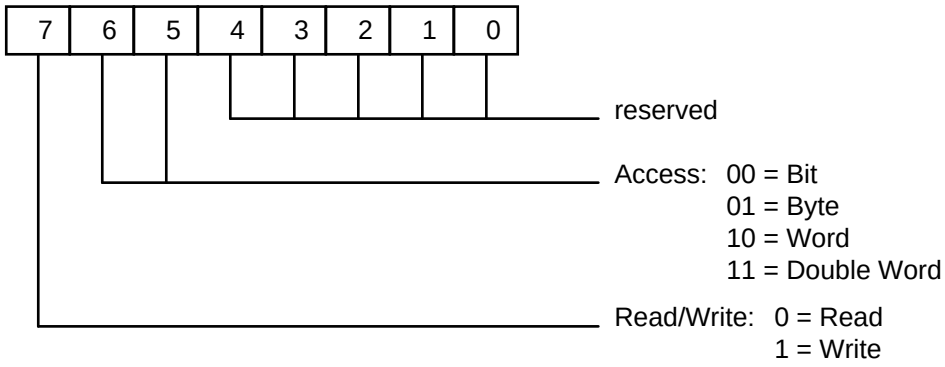
a) Byte 1



Number:
Contains the number of the user data transmitted.

Sequence Number:
Allows the controller program to recognize whether a new request has arrived from the operating terminal.

b) Byte 2



Access:
This value indicates whether bits, bytes, words or double words are read or written from the specified data area.

Data Direction:
Bit 7 marks the data direction, i.e. whether the controller is read from or written to.

c) Bytes 3 and 4
These bytes are used to transmit the offset within the specified data area.

5.5.3.3 User Data

The user data as of byte 5 up to telegram length are assigned in accordance with the access.

5.5.3.3.1 Reading and Writing Bytes

During read and write operations up to 28 bytes of user data can be transmitted - depending on the telegram length and access.
During write operations, the user data are located in the request telegram from byte 5 onwards.
During read operations, the user data are located in the response telegram from byte 5 onwards as well.

Byte 5	Byte 6		Byte n
User Data 1. Byte	User Data 2. Byte		User Data n. Byte

5.5.3.3.2 Reading Bits

During bit read-operations, a bit, a byte, a word or a double word is read depending on the address width of the data area to be read (see 5.5.3.3.1). The requested bits are then masked out and displayed by the operating terminal.

5.5.3.3.3 Writing Bits

Only one bit is set or deleted at one time.
The controller receives a bit mask and a logic information via the request telegram. The operating terminal uses this bit mask and the logic information to set or delete the bit at the destination address. The byte-order of the bit mask in the case of word addresses complies with the protocol parameter BYTE-ORDER.
To a byte address

Byte 5	Byte 6
Bit Mask	Logic Instruction 0 = AND/ 1 = OR

To a word address

Byte 5	Byte 6	Byte 7
Bit Mask Low	Bit Mask High	Logic Instruction 0 = AND / 1 = OR

5.5.4 Tasks of the Control Program

The control program, generally consisting of a function block, is required to handle the requests from the operating terminal in accordance with the data profile.

This process is controller-specific and is therefore described in the following chapters in a controller-specific manner.

5.5.5 Connection to Siemens-PLC

The program of the PLC communicates with the Profibus-DP via the I/O-peripheral area.

An IN and OUT-data channel is assigned to every Profibus-station, to every connected operating terminal.
The assignment is performed via the parameterization of the Siemens PLCs Profibus-DB master module.
Two modules, the IM308B and IM308C, are used in the Siemens-S5 PLCs.

5.5.5.1 Parameterization of the IM308B

To parameterize the IM308B, a device-type-file is available for the COMET200 programming software. (BT081ATD.200)

The element "Configuration" allows the DP-Slave, i.e. the operating terminal, to be configured.
The following parameters are set:
1. Location of the operating terminal in the peripheral area, i.e. the I/O-addresses to be occupied by the operating terminal.
2. The DP-identifier via module 0
   - I/O: X
   - Length: 4, 6, 8, 12, 14 and 16
   - Format: Word
   - Consistency: 0
3. Data for the parameterization telegram are not required, no entry.

This configuration is loaded to the IM308B by means of an EPROM-module. This EPROM also contains the entire parameterization of the master module.

5.5.5.1.1 Data Consistency

A data consistency over the entire specified length is required regarding the exchange of data between the operating terminal and the master module.

The master module IM308B and IM308C ensure a consistency only up to a maximum data length of 1 word.

The selection of appropriate settings in the Profibus-DP driver of the TesiMod operating terminals, however, allow a consistency of up to the maximum selectable data length of 32 bytes.

5.5.5.2 PLC Program

Evaluation of the Control Bytes
The peripheral area assigned to the operating terminal must be polled by the PLC program at cyclic intervals. By means of the sequence number, the PLC Program is capable of determining whether a new request has been received from the operating terminal.

This task is performed by the function block FB110.
FB110 is parameterized with the peripheral address of every operating terminal, thus only one FB is required even if multiple operating terminals are connected.

FB110 is also responsible for *copying bytes 1 and 2* (without any changes) from the request telegram to the response telegram and for writing *0x00* to *byte 3.*

Processing the user data
The processing of the user data is performed by separate read and write FBs that are called by the FB110.

The read and write FBs process the user data in accordance with the data profile.

Error Handling
Any errors that have occurred can be entered in the return code, i.e. in byte 4 of the response telegram. If no error has occurred, byte 4 must be deleted.
Possible error: DB does not exist.

5.5.5.2.1 FB110 - Evaluation Block

The function block (FB) uses the flag words (FW) 246 - 254 as scratch flags.

Furthermore, the FB requires one (any) data word of a DB as parameter which is evaluated when the FB is called. This data word is used to store the telegram number.

FB110 checks the contents of byte 1 - bit 5..7 at cyclic intervals.
If it contains the value 0, the telegram number memory is reset.
A new request telegram has arrived from the operating terminal that must be evaluated and responded to if byte 1 - bit 5--7 does not correspond with the contents of the telegram number memory.

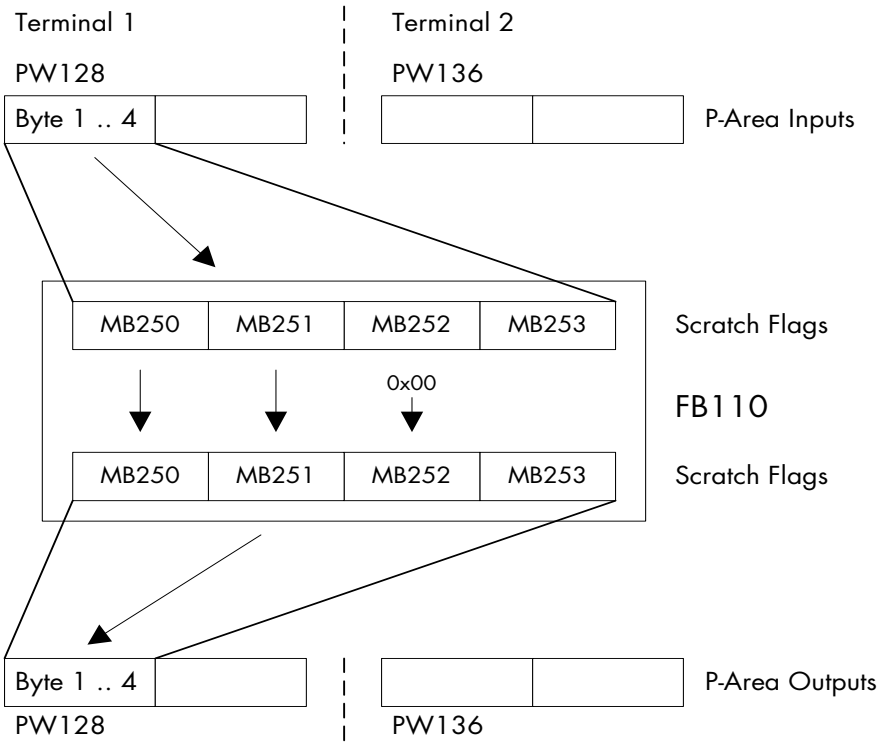
For each operating terminal, the FB110 is called cyclically in the organization block OB1 with the corresponding parameters.

FB110 is called

```
Block OB1

     :SPA  FB 110
Name :PROFIBUS
PEIN :     KF +00128        Peripheral address inputs
PAUS :     KF +00128        Peripheral address outputs
DBNR :     DB 5             Auxiliary data word block
WDNR :     DW 0             Auxiliary data word
     :
     :SPA  FB 110
Name :PROFIBUS
PEIN :     KF +00136        Peripheral address inputs
PAUS :     KF +00136        Peripheral address outputs
DBNR :     DB 5             Auxiliary data word block
WDNR :     DW 1             Auxiliary data word
     :
     :BE
```

Structure of the FB110



5.5.5.2.2 FB111 - Reading from Data Block

This description for the function block is valid from Version 2.0 (PROF02ST.S5D).

The FB interprets the offset in bytes 3 and 4 of the telegram as follows:
- Byte 3 contains the data block number,
- Byte 4 contains the data word number within the DB.

Bit 0 of the 2nd byte is interpreted as a byte identifier if there is a byte access to a word address.
0 = DL - High byte, 1 = DR - Low byte
The numbering of the bytes in the telegram can you read in Chapter 5.5.3.1.

5.5.5.2.3 FB112 - Writing to Data Block

This description for the function block is valid from Version 2.0 (PROF02ST.S5D).

The FB interprets the offset in bytes 3 and 4 of the telegram as follows:
- Byte 3 contains the data block number,
- Byte 4 contains the data word number within the DB.

Bit 0 of the 2nd byte is interpreted as a byte identifier if there is a byte access to a word address.
0 = DL - high byte, 1 = DR - low byte

During a bit-access to the data word, bytes 5 and 6 contain the bit mask and byte 7 contains the logic instruction.
The numbering of the bytes in the telegram can you read in Chapter 5.5.3.1.

5.5.5.3 Protocol Parameters

Address Width: 2
Byte Order: (.)

5.5.5.4 Programming the variables

To programm a variable in the variable list the following values must be entered.
In the column "parameter 1" a 4 digit hexadecimal number as address code must be entered. The high byte transposed into the 3rd byte and the low byte in the 4th byte.
The value in column "parameter 2" transposed at bit access in bit number 0..1 in the 5th..7th byte in the telegram.
The value in column "parameter 2" transposed at byte access in byte number 0..1 in bit 0 oft the 2nd byte.
The numbering of the bytes in the telegram can you read in Chapter 5.5.3.1.

Example of variable list:

Inputs in the variable list				**Access to addresses in the PLC**	
Variable name	Access	Variable number	Bit number	PLC access	PLC address
Var1	DW	124BH	0	Double word	DB18 DW75 + DW76
Var2	W	124BH	0	Word	DB18 DW75
Var3	BY	124BH	0	Byte	DB18 DL75
Var4	BY	124BH	1	Byte	DB18 DR75
Var5	B	124BH	13	Bit	DB18 DW75 bit 13

5.5.6 Connection to Bosch-PLC

The program of the PLC communicates with the Profibus-DP via the I/O-peripheral area.

An IN and OUT-data channel is assigned to every Profibus-station, every connected operating terminal.
The assignment is performed via the parameterization of the Bosch PLCs Profibus-DB master module.
The module MP-DP12 is used in Bosch-PLCs.

5.5.6.1 Parameterization of the BM_DP12

The parameterization of the BM_DP12 is performed with the Bosch DP-software.
The supplied device-master-data-file SUET081A.GSD is read-in by the DP-software.
This makes the data required for the parameterization of the BT-series available in the DP-software.

The BT-series must be configured with a data length of 8, 12 or 16 **bytes**.

5.5.6.2 PLC Program

Evaluation of the Control Bytes
The peripheral area assigned to the operating terminal must be polled by the PLC program at cyclic intervals. By means of the sequence number, the PLC Program is capable of determining whether a new request has been received from the operating terminal.

This task is performed by the supplied PB - BT_MAIN.
The BT_MAIN is parameterized with the peripheral address of every operating terminal, thus only one BT_MAIN is required even if multiple operating terminals are connected.

BT_MAIN is also responsible for *copying bytes 1 and 2* (without any changes) from the request telegram to the response telegram and for writing *0x00* to *byte 3.*

Processing of the user data
The processing of the user data is performed by separate read and write PBs that are called by the BT_MAIN.

The read and write PBs process the user data in accordance with the data profile.

Error Handling
Any errors that occurred can be entered in the return code, i.e. in byte 4 of the response telegram.
If no error has occurred, byte 4 must be deleted.
Possible error: DB does not exist.

5.5.6.2.1 BT_MAIN - Evaluation Block

This description for the function block is valid from Version 1.01.

The FB uses the flag words (FW) 246 - 254 as scratch flags.

Furthermore, the FB requires one (any) data word of a DB as parameter which is evaluated when the FB is called. This data word is used to store the telegram number.

The BT_MAIN checks the contents of byte 1 - bit 5..7 at cyclic intervals.
If it contains the value 0, the telegram number memory is reset.
A new request telegram has arrived from the operating terminal that must be evaluated and responded to if byte 1 - bit 5--7 does not correspond with the contents of the telegram number memory.

For every operating terminal, the BT_MAIN is called cyclically in the OB1 with the corresponding parameters.

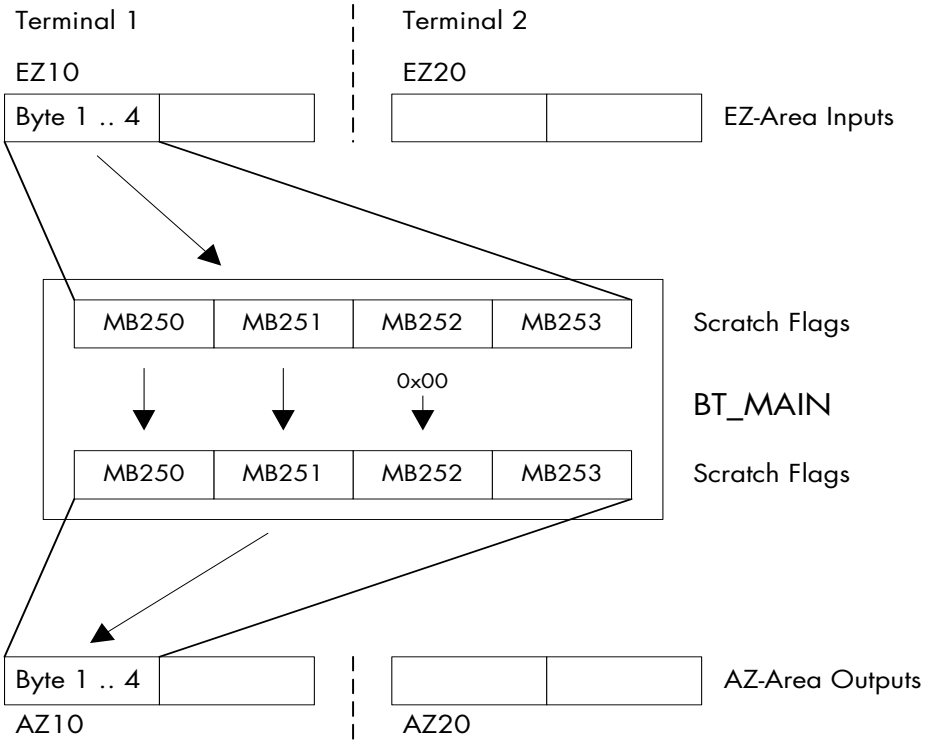
BT_MAIN is called

```
; OB1  Organization Block
; ************************************************
; Profibus-DP-Communication with Suetron Operating Terminal
; Example for integration into OB1
; ************************************************

; Commands required for Profibus
L    W  EZ2,A     ; Address must comply with coupling address
T    W  A,AZ2

; Call once for each operating terminal
BA      -BT_MAIN,4        ;Call for first operating terminal
;                  +---+
P0   W  K10       ; <   ! Address of the input area
P1   W  K10       ; <   ! Address of the output area
P2   W  DB0       ; <   ! Number of the data block
P3   W  D0        ; <   ! Data word number
;                  +---+
PE
```

Structure of the BT_MAIN



5.5.6.2.2 BT_READ - Reading from Data Block

This description for the function block is valid from Version 1.01.

The FB interpretes the offset in bytes 3 and 4 in the telegram as follows:
Byte 4 contains the data block number,
Byte 3 contains the data word number inside the DB (0...255).

The FB doubles the data word number to a even numbered byte number in the DB.
Bit 0 in the 2nd byte is interpreted as a byte number at a byte access in word addresses.
0 = even address - low byte
1 = odd address - high byte

The numbering of the bytes in the telegram can you read in Chapter 5.5.3.1.

5.5.6.2.3 BT_WRITE - Writing to Data Block

The FB interprets the bytes 3 and 4 in the telegram as follows:
Byte 4 contains the data block number,
Byte 3 contains the data word number inside the DB (0..255).

The FB doubles the data word number to a even numbered byte number in the DB.
Bit 0 in the 2nd byte is interpreted as a byte number at a byte access in word addresses.
0 = even address - low byte
1 = odd address - high byte

During a bit-access to the data word, bytes 5 and 6 contain the bit mask and byte 7 contains the logic instruction.

The numbering of the bytes in the telegram can you read in Chapter 5.5.3.1.

5.5.6.3 Protocol Parameters

Address Width: 1
Byte Order: ( )

5.5.6.4 Programming the variables

To programm a variable in the variable list the following values must be entered.
In the column "parameter 1" a 4 digit hexadecimal number must be entered. The high byte transposed into the 3rd byte and the low byte in the 4th byte.
The value in column "parameter 2" transposed at bit access in bit number 0..1 in the 5th..7th byte in the telegram.
The value in column "parameter 2" transposed at byte access in byte number 0..1 in bit 0 oft the 2nd byte.
The numbering of the bytes in the telegram can you read in Chapter 5.5.3.1.

Example of variable list:

Inputs in the variable list				**Access to addresses in the PLC**	
Variable name	Access	Variable number	Bit number	PLC access	PLC address
Var1	DW	124BH	0	Double word	DB18 D150 to D153
Var2	W	124BH	0	Word	DB18 D150 and D151
Var3	BY	124BH	0	Byte	DB18 D150
Var4	BY	124BH	1	Byte	DB18 D151
Var5	B	124BH	5	Bit	DB18 D150 bit 5
Var6	B	124BH	13	Bit	DB18 D151 bit 5

5.5.7 Protocol Parameters

5.5.7.1 Response Timeout

This parameter indicates the minimum intervals of time at which the operating terminal must be polled by the master.
The permissible range of values is 1ms... 65535 ms. The default value is 1000ms.

5.5.7.2 Communication Set-up Delay

This parameter indicates the period of time after which the operating terminal must be polled by the master for the first time.
The permissible range of values is 1000 ... 65535 ms. The default value is 5000ms.

5.5.7.3 Station Number

Any station number between 3 and 124 can be selected. The default value is station number 3.

5.5.7.4 Telegram Length

Can be set to a value between 8 and 32 bytes. The setting must comply with the configuration of the master module. The default value is 20 bytes.

5.5.7.5 Floating Point Format

With the value 0, the 4 byte number is interpreted in accordance with the Siemens format for floating point numbers.
With the value 1, the 4 byte number is interpreted in accordance with the IEEE format.

5.5.7.6 Byte-Order for Word and Double Word

High-low can selected for byte-order, otherwise the byte-order is selected by the TS-software.

If the value is 0, the byte-order of the controller is as follows:
within a word
HIGH-byte is located on the LOW-address and
LOW-Byte is located on the HIGH-address.
within a double word
HIGH-word is located on the LOW-address and
LOW-word is located on the HIGH-address.

If the value is 1, the byte-order of the controller is as follows:
within a word
HIGH-byte is located on the HIGH-address and
LOW-byte is located on the LOW-address.
within a double word
HIGH-word is located on the HIGH-address and
LOW-word is located on the LOW-address.

5.5.7.7 Address Width

The address width specifies the number of bytes per address.
1 = byte address
2 = word address
3 = double word address

The default value is 2.

5.5.8 Additional Functions

In addition to the random read and write access to PLC variables, a memory area comprising 12 bytes (default) is specified in the mask definition as poll area.

Only marginal conditions regarding this memory area:
- the PLC must be able to access in bit-mode and the operating terminal in word-mode
- the memory area must be contiguous.
The address location of this memory area is specified in the mask definition.

This area must be read using a **word access**.

Example: The cyclic poll area is set to data word in the programming system.

Word address	**DW**	**Highbyte**	**Lowbyte**
WORD address +0	DW100	Write coordination byte	Reserved
WORD address +1	DW101	Message channel high-byte	Message channel low-byte
WORD address +2	DW102	Function keys LED1..4	LED5..8
WORD address +3	DW103	Function keys LED9..12	LED13..16
WORD address +4	DW104	Function keys LED17..20	LED21..24
WORD address +5	DW105	Function keys LED25..28	LED29..32

5.5.9 Physical Interfacing

The interface complies with the standard specifications for Profibus-DP.

Assignment:

Pin	Designation	Function
1	nc	not connected
2	nc	not connected
3	RxD/TxD-P	Transmit / Receive Data -P
4	CNTR-P *1)	Control signal (Control-P)
5	DGND *2)	Data Reference Potential
6	VP *3)	Power Supply Plus
7	nc	not connected
8	RxD/TxD-N	Transmit / Receive Data -N
9	CNTR-N	Control signal (Control-N)
Housing	Shield	Shield / Protective Ground

*1) Complies with the TTL-signal
*2) Internally connected with the Shield / Protective Ground
*3) +5V ±5% Output to supply a bus termination; maximum current is 10 mA.

5.5.9.1 Connecting Cable

If the wiring is performed in accordance with the figure shown above, the potential difference between the data reference potential DGND of all connections is not allowed to exceed +/- 7 Volts. Circulating currents through the shield of the bus cable are not allowed. If this can not be ensured, provisions must be made for a potential equalization.

Principally, all cable types specified in EN 50170 as cable type A can be used.
This allows the following cable lengths (depending on the baud rate):

Baud Rate (bps)	Cable Length (m)
9 600	1200
19 200	1200
93 750	1200
187 500	1000
500 000	400
1 500 000	200
12 000 000	100

5.5.10 Error Messages

Code 1
Subcode

1	E_SLAVE_NOT_READY	Slave not ready
2	E_PROTOKOL	Sequence of the packets
3	E_FRAME	Protocol frame error
4	E_TIMEOUT	Timeout error
5	E_CRC_BCC	CRC error
6	E_PARITIY	Parity error
7	E_SEND_ABORT	Send process aborted
8	R_REC_ABORT	Receive process aborted
9	E_BUF_SIZE	Insufficient cyclic buffer
10	E_NO_DEFINE	No cyclic data defned
12	E_DEFINE	Cyclic data already defined
15	E_NO_PROTOCOL	Selected protocol is not supported
16	E_OVERRUN	Receive overrun
40	E_SYS_ADDRESS	Undefined system variable

Code 50 Error during initialisation of the SPC3
Subcode

1	IO_LENGTH_ERROR	Buffer too large
2	INIT_ERROR	No Initialisation of SPC3
4	MEM_FREE_ERROR	No memory available for message buffer

Code 60	E_CONFIG_ERROR	No configuration of master
61	E_CONFIG_INPUT_LENGTH	Input-length invalid
62	E_CONFIG_OUTPUT_LENGTH	Output-length invalid
63	E_CONFIG_FAULT	Faulty configuration data. New parameterization necessary
64	E_CONFIG_UPDATE	Protocol chip needs configuration update. New parameterization necessary
65	E_GO_OFFLINE	No communication over protocol chip. New parameterization necessary
66	E_MAC_RESET	Reset of protocol chip. New parameterization necessary

67 E_WD_TIMEOUT ..... Watchdog time-out. New parameterization necessary

Code 70 E_NO_POLL_TIMEOUT ..... Operating terminal is not polled

Subcode
0 Differentiation for manufacturer
1 Differentiation for manufacturer

Code 71 E_NO_RESPONSE_TIMEOUT ..... Order is not responded to

Code 100 E_DATA_ERROR ..... Base no. for error from PLC-FB PLC error is added to 100

Subcode Contains the offset value of the access upon which the error occurred.

e.g.: 102 ..... Access to DB via FB111/FB112 DB does not exist

5.5.11 Device-Master-Data-File

```
;=======================================================================
;===                                                                 ===
;===   Fa. Sütron electronic GmbH                                    ===
;===   Kurze Straße 29                                               ===
;===   70794 Filderstadt                                             ===
;===   Tel.: 0711 / 77098-0                                          ===
;===   Your contact: W. Ebinger, Extention 23                        ===
;===                                                                 ===
;=======================================================================
;===                                                                 ===
;===   Device-Master-Data-File:    SUET081A.GSD                      ===
;===                                                                 ===
;===   Revision Level: GV1.0 - 15 July 1996                          ===
;===                                                                 ===
;=======================================================================
;===                                                                 ===
;===   Bus station, connecting an operating terminal                 ===
;===   of the BT-series                                              ===
;===                                                                 ===
;=======================================================================
;===                                                                 ===
;===   Attention:                                                    ===
;===   ==================                                            ===
;===   Improper modification of the parameters may lead to           ===
:===   an undefined system behaviour of the Profibus-DP-bus          ===
;===   and is done at one's own risk.                                ===
;===                                                                 ===
;=======================================================================
;=======================================================================
;
#Profibus_DP
;
;***********************************************************************
; General Data
;***********************************************************************
;
; DP-Device Type
Station_Type = 0
;
; Name of the Manufacturer
Vendor_Name = "Suetron electronic, Filderstadt"
;
; Name of the Device
Model_Name = "BT-Serie"
;
; Revision Level of the DP-Device
Revision = "1.0"
;
; Version Identifier of the DP-Device
```

```
Revision_Number = 0x01
;
; Type Number
Ident_Number = 0x081A
;
; Release of the Hardware
Hardware_Release = "SWV1.0"
;
; Release of the Software
Software_Release = "HWV1.0"
;
; Protocol Identifier
Protocol_Ident = 0
;
; Supported Baud Rates
9.6_supp = 1
19.2_supp = 1
93.75_supp = 1
187.5_supp = 1
500_supp = 1
1.5M_supp = 1
3M_supp = 1
6M_supp = 1
12M_supp = 1
;
; Maximum Protocol Processing Time
MaxTsdr_9.6 = 11
MaxTsdr_19.2 = 11
MaxTsdr_93.75= 11
MaxTsdr_187.5 = 11
MaxTsdr_500 = 11
MaxTsdr_1.5M = 11
MaxTsdr_3M = 11
MaxTsdr_6M = 25
MaxTsdr_12M = 50
;
;Level of the Repeater Control Signal CNTR-P
Repeater_Ctrl_Sig = 2
;
;=============================================================================
; Specific Data of the Bus Station
;=============================================================================
;
; Automatic Recognition of the Baud Rate
Auto_Baud_supp = 1
;
; Type of Bus Station (modular or compact)
Modular_Station = 1
;
; Maximum Number of Modules of a Modular Bus Station
Max_Module = 1
```

```
;
; Maximum Length of the Input Data of the Bus Station in Bytes
Max_Input_Len = 32
;
; Maximum Length of the Output Data of the Bus Station in Bytes
Max_Output_Len = 32
;
; Maximum Length of the Input- and Output Data (Sum) in Bytes
Max_Data_Len = 64
;
; Module Identifier
;
Module = "8 Byte Daten E/A" 0x37
EndModule
;
Module = "10 Byte Daten E/A" 0x39
EndModule
;
Module = "12 Byte Daten E/A" 0x3B
EndModule
;
Module = "14 Byte Daten E/A" 0x3D
EndModule
;
Module = "16 Byte Daten E/A" 0x3F
EndModule
;
Module = "4 Worte Daten E/A" 0x73
EndModule
;
Module = "6 Worte Daten E/A" 0x75
EndModule
;
Module = "8 Worte Daten E/A" 0x77
EndModule
;
Module = "10 Worte Daten E/A" 0x79
EndModule
;
Module = "12 Worte Daten E/A" 0x7B
EndModule
;
Module = "14 Worte Daten E/A" 0x7D
EndModule
;
Module = "16 Worte Daten E/A" 0x7F
EndModule
;
; DP-Device does not support the Freeze-Mode
Freeze_Mode_supp = 0
;
```

```
; DP-Device does not support the Sync-Mode
Sync_Mode_supp = 0
;
; DP-Device does not Support the Bus Station Addressing
Set_Slave_Add_supp = 0
;
; Minimum Interval between two Accesses to the Bus Station
Min_Slave_Intervall = 100
;
; Maximum Length of the Diagnosis Information (DiagData)
Max_Diag_Data_Len = 12
;
; Class of Function: I/O
Slave_Family = 3
;============ End of Device-Master-Data-File============================
```

6 Index

Symbole

A

B

C

D

E

F

G

H

I

S

T

U

V

W

Z

A Appendix A

A.1 List of Accessories

Designation	Length	Order Number
Cable BT2/BT5/BT5N/BT20N <-> Bosch PLC (TTY/20mA)	3 metres	1070 917 855
Cable BT2/BT5/BT5N/BT20N <-> PC (Download)	3 metres	1070 917 856
Cable BT20 <-> Bosch PLC (TTY/20mA)	3 metres	1070 917 857
Cable BT20 <-> PC (Download)	3 metres	1070 917 858
Cable BT2/BT5N/BT20N <-> Bosch PLC (CL150-RS232c)	3 metres	1070 920 422
Cable BT2/BT5/BT5N/BT20N <-> Bosch PLC (CL151-TTY/20mA)	3 metres	1070 920 423

Bosch Automation Technology

Australia
Robert Bosch (Australia) Pty. Ltd.
Head Office
Cnr. Centre - McNaughton Roads
P.O. Box 66
AUS-3168 Clayton, Victoria
Fax (03) 95 41 77 03

Great Britain
Robert Bosch Limited
Automation Technology Division
Meridian South
Meridian Business Park
GB-Braunstone Leicester LE3 2WY
Fax (01 16) 289 2878

Canada
Robert Bosch Corporation
Automation Technology Division
6811 Century Avenue
CAN-Mississauga, Ontario L5N 1R1
Fax (905) 5 42-42 81

USA
Robert Bosch Corporation
Automation Technology Division
Fluid Power Products
7505 Durand Avenue
USA-Racine, Wisconsin 53406
Fax (414) 5 54-81 03

Robert Bosch Corporation
Automation Technology Division
Factory Automation Products
816 East Third Street
USA-Buchanan, MI 49107
Fax (616) 6 95-53 63

Robert Bosch Corporation
Automation Technology Division
Industrial Electronic Products
40 Darling Drive
USA-Avon, CT 0 60 01-42 17
Fax (860) 4 09-70 80

Your concessionary

BOSCH

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Antriebs- und Steuerungstechnik
Postfach 11 62
D-64701 Erbach
Fax +49 (0) 60 62 78-4 28

1070 083 630-101 (00.05) GB HB SP AT/PLS Printed in Germany