

IBT



 früher
EUROTHERM DRIVES

Series







Intelligent Operator Terminal

Model: IBT

Technical manual

Weitere Unterlagen,
die im Zusammenhang mit
diesem Dokument stehen.

Further descriptions,
that relate to this document.

635 - Produkt-Handbuch	UL: 7.1.5.6 	<i>635 - Product manual</i>
637 - Produkt-Handbuch	UL: 7.2.8.3 	<i>637 - Product manual</i>
IBT - Produkt-Beschreibung	UL: 9.5.1 	<i>IBT - Product description</i>
IBT - Erstinbetriebnahme	UL: 9.5.2 	<i>IBT - Getting started</i>

© **EUROTHERM** Antriebstechnik GmbH.
Alle Rechte vorbehalten. Kein Teil der Beschreibung
darf in irgendeiner Form, ohne Zustimmung der Gesell-
schaft vervielfältigt oder weiter verarbeitet werden.

Änderungen sind ohne vorherige Ankündigung
vorbehalten.

EUROTHERM hat für seine Produkte teilweise Waren-
zeichenschutz und Gebrauchsmusterschutz eintragen
lassen. Aus dem Überlassen der Beschreibungen darf
nicht angenommen werden, daß damit eine Übertragung
von irgendwelchen Rechten stattfindet.

Hergestellt in Deutschland, 1998

© **EUROTHERM Drives Limited.**
*All rights reserved. No portion of this description may
be produced or processed in any form without the
consent of the company.*

Changes are subject to change without notice.

EUROTHERM has registered in part trademark
protection and legal protection of designs. The handing
over of the descriptions may not be construed as the
transfer of any rights.

Made in Germany, 1998

Contents

1	TesiMod Operating Concept.....	7
1.1	Operating Terminals.....	7
1.1.1	Loadable Protocol Drivers.....	7
1.1.2	Networking the Terminals.....	8
1.1.3	Programming Unit Interface.....	8
1.1.4	What is TesiMod and which functions does it provide?.....	8
1.1.5	Operating Modes.....	8
1.1.6	Application Description.....	8
1.1.7	Storing the Application Description.....	9
1.1.8	Addressing Variables.....	9
1.1.9	Data Release.....	9
1.1.11	Editors.....	10
1.1.12	Help.....	10
1.1.13	Function Keys.....	11
1.1.14	Soft Keys.....	11
1.1.15	Variables.....	11
1.1.16	Graphics.....	11
1.1.17	Recipes and Data Sets.....	12
1.1.18	Messages.....	13
1.1.19	System Messages.....	13
1.1.20	Programming System.....	13
1.2	Common Features:.....	14
1.2.1	Hardware.....	14
1.2.2	Software.....	14
1.3	Order Information.....	14
2	Operating Modes.....	15
2.1	Setting the Operating Mode.....	15
3	TesiMod Standard Mode.....	16
3.1	Setting the Operating Mode.....	16
3.2	Startup Process.....	17
3.2.1	Startup Process without a Valid User Description.....	17
3.3	Communication in the Standard Mode.....	18
3.4	Operating Concept.....	18
3.4.1	Hierarchical Mask Structure in TSdos.....	18
3.4.2	Mask Structure in TSwIn.....	19
3.4.3	External Mask Selection.....	19
3.4.4	Password Protection, Access Authorisation.....	20
3.5	Masks.....	23
3.5.1	Mask Parameters.....	23
3.5.2	System Masks.....	24
3.6	Variables.....	30
3.6.1	Output Variables.....	30
3.6.2	Input Variables.....	41
3.6.3	System Variables.....	43
3.6.4	Editors.....	69
3.6.5	External Data Release.....	75
3.6.6	PLC-Handshake.....	76
3.6.7	Refreshing One-Time Output Data.....	76
3.6.8	Modified Data.....	77
3.7	Graphics.....	78
3.7.1	Graphical Objects (TSdos).....	78
3.7.2	Images (TSwIn).....	78
3.7.3	Graphics on Operating Terminals.....	79
3.8	Recipes.....	79
3.8.1	Structure of a Recipe.....	81
3.8.2	Processing Recipes and Data Sets.....	81
3.8.3	Data Set Transfer to / from a Controller.....	83

3.8.4	Transferring Data Sets to / from a PC	85
3.8.5	Printing Data Sets	87
3.8.6	Memory Requirements for Storing Data Sets	88
3.9	TesiMod Message System	88
3.9.1	Internal Messages	88
3.9.2	External Messages	96
3.10	Help System	104
3.10.1	Default Help Text	104
3.10.2	Help Text For Masks	104
3.10.3	Help Text For Variables	104
3.11	Function Keys	105
3.11.1	Direct Selector Keys	105
3.11.2	Function Keys of the Controller	105
3.11.3	Soft Keys	105
3.11.4	Function Keys Controlling Parallel Outputs	106
3.11.5	Status LEDs in the Function Keys	107
3.12	System Parameters	107
3.12.1	System Parameters: General Parameters	107
3.12.2	System Parameters: Poll Area	108
3.12.3	System Parameters: Terminal Clock	108
3.12.4	System Parameters: Running Time Meter	108
3.12.5	System Parameters: Message System	108
3.12.6	System Parameters: Variant Buffer	108
3.12.7	System Parameters: Password Management	108
3.12.8	System Parameters: Printer Interface	108
3.12.9	System Parameters: Gateway	108
3.12.10	System Parameters: Data Set Transfer	109
3.12.11	System Parameters: Parallel Outputs	109
3.13	Version Number	109
3.14	Running Time Meter	110
3.15	Parallel Outputs	111
3.16	Screen Saver	111
3.17	Image of the Mask Number	111
3.18	Image of the Mode Selector Switch	112
3.19	Terminal Clock	112
3.19.1	Image of Date and Time	112
3.20	Read Coordination Byte	113
3.20.1	Editing Request Bit (Bit "EA")	113
3.20.2	Editing Status Bit (Bit "EZ")	113
3.20.3	Refresh Request Bit (Bit "RA")	113
3.20.4	Liveness Flag (Bit "LM")	114
3.20.5	Data Set Download Active (Bit "DDA")	114
3.21	Write Coordination Byte	114
3.21.1	External Data Release (Bit "ED")	114
3.21.2	Refresh Acknowledgement (Bit "RQ")	115
3.21.3	Resetting the Password	115
3.21.4	Liveness Flag (Bit "LM")	115
3.21.5	Data Set Download Release (Bit "DDF")	115
3.22	Cyclic Poll Area	115
3.22.1	Byte-Oriented	115
3.22.2	Word-Oriented	117
3.22.3	Image of the LEDs	117
3.22.4	Serial Message Channel	118
3.22.5	Polling Time	118
3.22.6	Size of the Poll Area	118
3.23	Control Codes	118
3.23.1	Triggering Data Set Printouts	118
3.23.2	Setting the Clock in the Operating Terminal	118
3.23.3	Transferring Data Sets from the Controller to the Terminal	119

3.23.4	Transferring Data Sets from the Terminal to the Controller	119
3.23.5	Transferring Data Sets from the Controller to the Terminal (Individually)	119
3.23.6	Refreshing the Message System	119
3.24	Cyclic Variables	119
3.25	Interface Parameters X2, X3	120
3.26	Variable Definition	120
3.26.1	Variable Formats (TSdos)	120
3.26.2	Variable List	120
3.27	Application Programming	122
3.27.1	Configuring the System	122
3.27.2	TSdos and TSwin Programming Systems	123
3.27.3	Getting Started with Programming	126
3.27.4	Project Documentation	127
3.27.5	Project Back-up	128
3.27.6	Optimizing the Transmission Rate	128
3.28	Downloading the User Description	129
3.28.1	Downloading with Windows	131
3.28.2	Application Memory	131
3.28.3	Loading a User Description	131
3.28.4	Activating the Download Function using the Software	132
3.28.5	Activating the Download Function using the Hardware	132
3.28.6	Automatic Download Function	132
3.28.7	Download Cable	133
3.29	Simulation without the Controller	134
5.22	TesiMod - CAN/CANopen	135
5.22.1	General	135
5.22.2	Technical Description	135
5.22.3	Indirect Process Data Communication	135
5.22.4	Request and Response Object Structure	136
5.22.5	Tasks of the Communication Partner	138
5.22.6	Protocol Parameters	138
5.22.7	Communication Relationships	139
5.22.8	Poll Area	141
5.22.9	Physical Interfacing	142
5.22.10	Error Messages	144
5.22.11	EUROTHERM specific addresses and definitions	145
5.22.12	Examples of Communication Relationships	146
	Index	148
	Modification Record	151

1 TesiMod Operating Concept

1.1 Operating Terminals

Our range of products is continuously expanded and tailored to the needs of the market. 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. A large number of the terminals offers 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 operating terminals are equipped with clearly readable, multiple-line, backlit or luminescent displays making them suitable for a wide range of applications. The keyboard is either equipped with mechanical short-stroke keys or membrane keys, all providing defined tactile feedback. 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 labeling.

Up to 8 function keys can be assigned to 24V optodecoupled parallel outputs. The outputs can be used to directly control PLC inputs. The advantage of the outputs is the short response time (<25ms) since they are independent from the serial transmission.

A serial interface is used to connect the terminal to the controller. A second serial interface is provided for the download function and for connecting a printer. The modular design of the hardware allows adaptation to various interface standards. The extent of delivery comprises a 20mA current loop and a RS232-interface. The interfaces are electrically isolated with respect to each other and the main board. Alternatively, we offer interface modules in accordance with the standards RS422 or RS485. A connection to all common field bus systems is also possible. It is possible to use cables with a length of up to 1000m depending on the interface standard and the baud rate selected.

1.1.1 Loadable Protocol Drivers

To keep the number of different terminal types small, the interface X2 can optionally be equipped with two interface standards simultaneously, the interface standards 20mA current loop and RS485, which are the interface standards most commonly used. The interfaces will be labeled as X2.1 and X2.2. In conjunction with the loadable protocol drivers, the operating terminals become independent of the controller with respect to hardware and firmware. The terminal allows exchange of data with any of the PLC types which are currently supported. It is not necessary to order the terminals in a protocol-specific manner (with the exception of field bus connections).

All terminals can be used to output, input and modify data in PLCs, microprocessor-based controllers, process computers or PCs.

Communication is handled either by means of the controller-specific protocol or a standard protocol. The controller-specific protocol ensures access to all system and user data as well as inputs and outputs, flags, timers, counters, etc. For connections to PLCs, the programming unit interface is normally used.

1.1.2 Networking the Terminals

Multipoint connections are supported, provided the controller-specific 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. However, only field bus connections allow multiple operating terminals to be connected to one PLC.

A cost-effective solution for other protocols is possible via the DIN-Meßbus according to DIN 66348. With this multipoint connection, one operating terminal operates as a bus master and gateway (protocol converter) towards the controller. All other operating terminals operate as DIN-Meßbus slave stations. For data addressing, the syntax of the connected controller is used. The bus protocol allows access to all standard TesiMod functions. Appropriate measures ensure a high efficiency between the DIN-Meßbus and the point-to-point protocol used (only available for terminals up to the firmware version 5). Every operating terminal can be operated as a bus master. The interface X3 of terminals operating as a bus master is, however, not available for connecting a printer but is required for the bus connection with a RS485 interface. This connection can be carried out in different ways, depending on the terminal type.

1.1.3 Programming Unit Interface

Since hardware and protocol software for the programming unit are already available on every PLC, there is no need for an additional communication processor, thus providing a low-cost solution to the user. The application programmer does not have to develop and install communication software for his PLC as would be the case with a communication 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 Süttron 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.

1.1.5 Operating Modes

Two modes of operation - the standard and the transparent mode of operation - are available on every TesiMod operating terminal.

1.1.5.1 Transparent Mode

TesiMod terminals operating in the transparent mode can be used as full-size ANSI-terminals. Every key generates a press and release code which is transmitted via the interface. ESC-sequences are used to control the display and LEDs. Various fonts and character attributes are available depending on the display type.

1.1.5.2 Standard Mode

This is the mode of operation in which TesiMod operating terminals provides its full range of functions. In standard mode, decoding of the keys and menu control by the controller is not required. With the standard mode, the operator guidance can be conveniently created, including password management, recipe management, scaling of variable values, graphical elements, tables and messages. The operator can be guided through the operator structure by means of soft keys, selection menus, function keys or the PLC.

1.1.6 Application Description

The programming data of the application description basically consists of three types of data.

Global data are data that apply to the entire project.

Language-specific data are data that apply only to a specific language within the project.

Controller-specific data are data that apply only to the selected controller within a project.

Global data are:

- system settings
- graphics

Language-specific data are:

- mask definition
- text lists
- help texts (help masks)

Controller-specific data are:

- communication settings
- variable list

Since the project data are divided into three subareas, the user can conveniently add a mask definition in another national language to the project or easily switch to another controller type or controller manufacturer.

The entire application description can be created by means of an easy-to-use PC software (TSdos or TSwIn).

Once programmed, the application description will appear to the operator simply as a combination of texts, messages, tables, graphs, graphics and values displayed on the operating terminal. The displayed elements are always combined to form one screen. This screen is referred to as a mask.

With TSdos, different mask types have been developed for different requirements.

With TSwIn, the function of a mask is determined by its contents.

1.1.7 Storing the Application Description

The customized description of all operating and terminal functions is stored in a Flash memory. This memory can be programmed and erased using the terminal, a programming unit is not needed for this process. Programming can be carried out while the memory is mounted, i.e. it is not necessary to open the terminal.

UV Eeproms can also be used instead of a Flash memory. UV Eeproms can, however, neither be programmed nor erased while mounted in the terminal. The Flash memory offers a power failure safe storage of masks and operator guidance without battery.

1.1.8 Addressing Variables

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

1.1.9 Data Release

To be able to modify values on the operating terminal, data release must have been provided. Data release can be activated automatically for all variables in a mask (mask parameters).

If the data release is not activated automatically, it must be requested by the operator by pressing the Data Release key on the operating terminal. The status LED in the Data Release key then indicates whether the data release has been provided (LED lights up) or not (LED flashes).

The data release can be denied by the controller (external data release). This requires the variables for the poll area (write coordination byte) and read coordination byte to be defined (in which bits are set or reset accordingly).

1.1.10 Password Protection

It is possible to assign up to eight passwords to protect data and the contents of masks. Each password has a view and an edit level. An access level is assigned to each mask. Depending on the password level of the user, access to masks or even entire menu trees may be allowed or denied. All data contained in a mask is subject to the same access authorization. The access authorization of the passwords is defined in the programming software. The access authorization is reset when the terminal is initialized by the operator, from within the controller or when the password entered is incorrect. Default passwords and access levels are defined when the mask definition is programmed. Passwords can be modified on the operating terminal. A master password can be defined which allows the default values to be restored. Without setting up a password protection function, all masks and variables are accessible.

1.1.11 Editors

It is not only possible to modify controller variables; terminal-specific system variables can also be modified.

A powerful editor is available for every data type. When entered, variables are checked for plausibility. The limits are determined by the user in the variable definition. 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 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 provided by the PLC and - in case of password protection - the correct code word has been entered).

Once in the edit mode, the LED in the Data Release key lights up and the cursor is located on the first editable variable. This value can then be edited using the numerical keypad with sign keys, decimal point and Delete key. The value is confirmed by pressing the Enter key. During this process, the value is checked for plausibility and a system error message is generated, if necessary.

Particularly powerful editors such as the table editor and the selection field editor allow larger amounts of data to be handled. The table editor permits entire columns of indexed variables to be edited. For the column editor type, a large number of editors is available such as the fixed point editor, floating point editor, BCD editor or hexadecimal 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. These are generated individually by means of system variables and soft keys.

1.1.12 Help

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

A flashing status LED in the Help key indicates a malfunction or that a message has been received. The system message (reporting that the value exceeds the upper or lower limit, for example) can be displayed by pressing the Help key. In the case of an error, the Enter key can not be used to exit the editing field. The Enter key will accept only valid values.

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 fields. If the Data Release key is pressed again, the editing mode will be exited and the status LED will go off.

1.1.13 Function Keys

The function keys and their status LEDs play an important role in the TesiMod operating concept, in addition to the masks. Identification strips are available to label the function keys as desired. Labeled identification strips and blank identification strips are supplied. The function keys are user-programmable. They can be used either as direct selector keys for choosing further masks and/or as control keys. The integrated LEDs which can be influenced by the controller serve as a 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.

Function keys can also be assigned default assignments for an entire application definition. A simultaneous use of the function keys as soft keys is still possible. The soft key definition takes precedence over the default function.

Another means in providing a more speedy operation is the editor. The editor keeps track of the position in a mask which was edited last and returns to this position when the mask is called up again.

1.1.14 Soft Keys

All function keys can also be used as soft keys. Soft keys can perform a toggle function within a mask. The current function must be shown on the display. One and the same soft key could be used to switch a pump on and off, for example.

1.1.15 Variables

The type of variables that are allowed such as bit, byte, word, double word or field depends on the connected controller. The concept allows read and write access to all types of data available within a controller, if not defined otherwise.

The variable output and input can be scaled and formatted. The following data formats are available: binary, integer, fixed point, floating point, alphanumeric and selection field. All data formats and variable types can be used for input and output. The scaling allows the conversion of millimeter to inch, degree Celsius to degree Fahrenheit, etc.

Different formats can be used to input variables. A variable can be output once or cyclically. Variables defined as editable are also output once or cyclically. The cycle time can be set in the programming system via system parameters.

1.1.16 Graphics

Graphics are used for a wide range of purposes including basic process visualization functions. Advanced diagnostic capabilities, easy-to-understand and more transparent process control and language independence from the symbols play an important role in this context.

Graphics may eliminate the need for translations into other national languages. They permit operator guidances via pictographs and allow process variables to be presented graphically. Wherever the need for precise numerical data is not so great, graphics in the form of graphs can provide clear information at a glance. Reactions by the operator, PLC and process can be visualized to present an easily comprehensible view of occurrences and process states within the entire plant or parts of the plant. The

employment of graphics can ultimately result in an 'icon-driven' operator guidance instead of a text-based one.

Graphics can, for example, be used to display:

- process variables graphically
- filling levels
- the location of a malfunction in a plant
- frequency distributions
- temperature behaviors
- transient effects
- trends
- the company logo
- uniform mask backgrounds

All texts, variables and graphics can be freely positioned.

Graphics can be created with any program supported by Windows95 or WindowsNT. The user can choose between embedded objects or separately managed objects.

1.1.17 Recipes and Data Sets

With the term „recipes“ we refer to the combination of texts, variables and units. The values for the variables are stored separately as data sets.

Example:recipe: Saw Wooden Frame; Data set: Beech

Text	Variable	Unit
Number of revolutions	1000	min ⁻¹
Feedrate	10	mm/s
Cutting angle	35	°

Creation of the recipe contents is subject to the same options as are available for masks.

The number of recipes in an operating terminal is theoretically limited to a maximum of 250 recipes.

A recipe may contain up to 250 data sets. Each data set can contain up to:

- 255 lines
- 255 variables and
- 255 text elements.

The limits of the system also depend on the memory available.

Unlike masks, recipes can be edited within recipe windows located in an I/O mask regardless of their length. The position and size of the recipe window can be defined in the programming software.

In addition to the recipe, a mask may contain other input and output variables. Recipes may contain input variables only. The variables are stored in the battery-backed CMOS-RAM of the operating terminal. Like mask variables, input variables can be formatted, scaled and be assigned help texts.

Data sets predefined in the programming software are stored in the Flash memory and can be copied into the RAM to be modified. The operating terminal provides functions that allow data sets to be created, deleted, copied and be transferred from and to the PLC. Data sets are managed dynamically in the terminal memory thus making optimum use of the memory area available.

Data sets can be transmitted to the controller either directly or via a communication buffer. Transmissions of data sets to the operating terminal are always carried out via a communication buffer. Recipe management and variable addressing is language-independent. The texts in the recipe are retrieved from the language-specific mask definition. The data sets of a recipe can be output onto a

printer via interface X3. System variables can be used to transfer a data set from the PC to the operating terminal and from the operating terminal to the PC.

1.1.18 Messages

Messages are operating states transferred to the operating terminal. Messages are stored in the message memory either by priority (message number) or in chronological order.

The current software version allows management of approximately 3000 messages within the message memory. The maximum size of the message memory can be predefined in the programming software. Messages are stored in the CMOS-RAM memory along with the recipes and loadable fonts.

There are two methods of transmitting messages to the operating terminals, the parallel and the serial transmission. Both systems can be used simultaneously and are handled with equal priority. 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 in the same way as masks.

Every message text may contain one formatted and scaled variable. Upon generation of a message, the current value of the variable is frozen.

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 to be printed including message number, date, time, message text and variable value. 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 status LED in the Help key will begin to flash thereby providing a visual signal. „Blank“ system messages are automatically suppressed by the system.

1.1.20 Programming System

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

This program greatly facilitates project creation and management. If you have any additional questions, please make use of the extensive help system or call our Hotline. Even while creating the program, the mask is displayed in its true format. All terminals are programmed in the same manner and offer the same functions.

In addition to the full-graphics creation of masks, the programming system includes a project management, a compiler, a documentation generation and a data transfer program for the download function.

The definition of the masks and representation of the variables is carried out in a protocol-independent manner. The reference to the target-PLC is made only by the reference list of the variable definition.

While creating the application definition using the programming system, the terminal can be operated without the target-PLC. This allows you to check and test the masks and the operator interface directly on the operating terminal.

If you wish to carry out initial tests, contact us for a demo-terminal.

Any further questions?

If you have additional questions, we will be glad to be of assistance.
Our hotline number is +49 / (0) 7253 - 9404-0.

1.2 Common Features:

1.2.1 Hardware

- Multiple-line, backlit or luminescent displays
- Filters for glare suppression and increased contrast
- Temperature compensation of the display (if required)
- Function keys with integrated green LEDs
- Identification strips for individual labeling of function keys
- Editing keys
- Control keys
- Mask memory (Flash memory) programmable while mounted in the terminal
- Battery-backed RAM for message memory and recipe data management
- At least two serial interfaces
- Various interface standards
- Battery voltage monitoring function
- Real time clock with date, time
- User mode switch

1.2.2 Software

- Standardized functionality for all terminals
- Two operating modes: standard and transparent mode
- Hierarchical operator guidance with direct selector keys
- All function keys provide a soft key functionality
- Password protection with different access levels
- Integrated help system
- All variable formats
- Pixel graphics: background images, bar graphs, selection images and line graphs (trendlines)
- All operating terminals with recipes
- Loadable protocol drivers for all major controller manufacturers
- Protocol-independent hardware
- Communication monitoring
- Connection of several operating terminal at the programming unit interface
- Bus connections for field bus systems and DIN-Meßbus
- Uniform programming software for all operating terminals

1.3 Order Information

Please call our sales department at
+49 / (0) 7253-9404-0 or
send a fax to +49 / (0) 7253-9404-99
or email us at info@eurotherm.de
to place your order or to obtain further information on one of our products.

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.

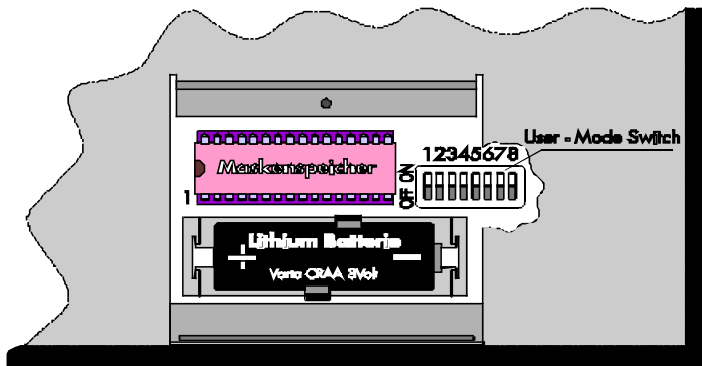
TesiMod terminals operating in the transparent mode represent full-size ANSI-terminals. Each key generates a press and release code which is transmitted via the serial interface as an ASCII character. The displays and the LEDs in the keys are controlled via ESC sequences. The number of fonts and character attributes that are available vary with the type of display. For a detailed description refer to the chapter "TesiMod - Transparent Mode".

In the standard mode of operation, the entire operator guidance (operator interface) is integrated into the terminal. The standardized operating concept completely frees the connected controller from all operator guidance and data display tasks. 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.

To set the operating mode, open the hinged cover at the rear side of the device or open the rear panel.



Switch	ON	OFF
S1	Standard mode	Transparent mode
S2	Unassigned	Unassigned
S3	Demo (without PLC)	Communication with the PLC
S4	Erase Flash completely	Maintain Flash data
S5..S8 (if available)	Unassigned	

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 visualisation 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 customise all menus and dialogs according to your particular application. Extremely sophisticated operator guidance's 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 mode selector 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

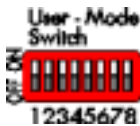


Fig. 1: Mode selector switch

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 initialisation 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 (initialisation period) is required to set up the structures for message management, which will then enable faster sorting of messages later. During the initialisation phase, the following message is displayed:

```
INITIALIZING
MESSAGE BUFFER
.....
.....
```

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

3.2.1 Startup Process without a Valid User Description

```
NO FLASH EPROM
.....
.....
```

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 these message.

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:

```
ERASE FLASH EPROM
.....
.....
```

```
FLASH IS ERASED
FLASH 256 kBYTE
HF XXXXX
.....
.....
```

```
DOWNLOAD
.....
.....
```

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 labelling, 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.

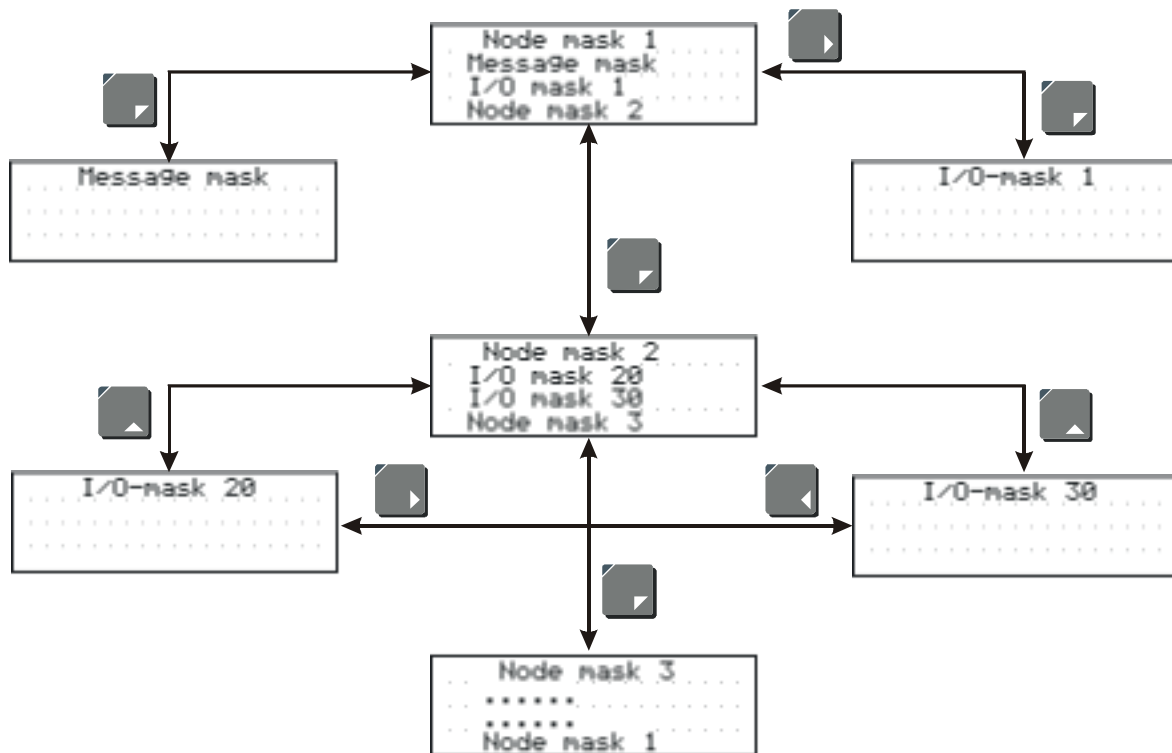


Fig. 2: Hierarchical Mask Structure

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 Authorisation

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 authorisation. 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 authorisation 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 authorisations 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:

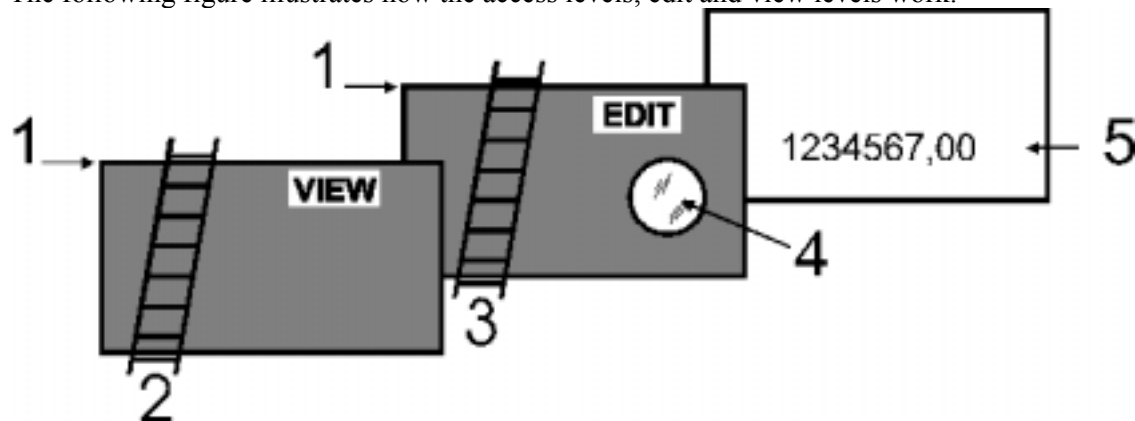


Fig. 3: Access levels

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 authorisation.
- 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 authorisation levels will automatically reset to 0.

It is recommended that at least one password - the master password - is programmed with the highest authorisation 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 authorised
- Mask 6 will be displayed, editing of values is **not** authorised
- Mask 7 will **not** be displayed, editing of values is **not** authorised

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 unauthorised 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 initialisation 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 authorisation 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 authorisation. When a password with sufficient authorisation 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 unauthorised 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 authorised 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 initialisation 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 initialisation 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 reinitialised.

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.

```
BaudrateX2: 9600
Parity:Even D_Bit:8
Übernahme:inaktiv
SN:000 Download:Nein
```

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

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

```
These mask boot
the project
in 5 seconds.
With enter to setup!
```

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

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.

```
Passwordinput
Password:xxxx
```

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

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.

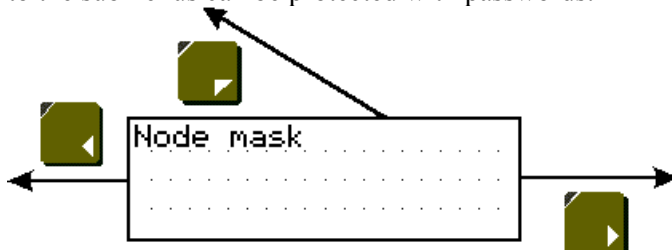


Fig. 7: Optional control keys in the node mask

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

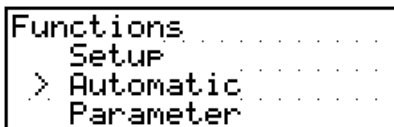


Fig. 8: Example of a node mask

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 guidance's 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

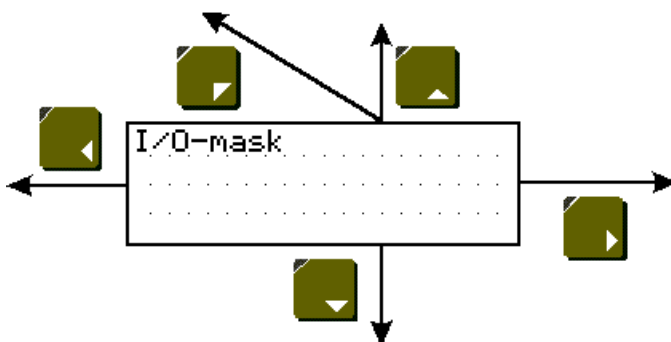


Fig. 9: Possible control keys in the I/O mask

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

```
Parameter
Position: 1000 inc.
speed  : 750 rpm
accel.  : 50000 rpm/s
```

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

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.

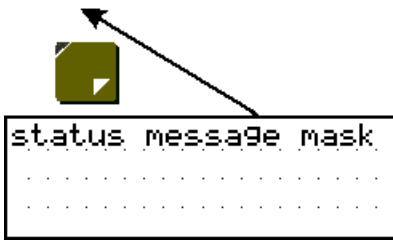


Fig. 12 : Possible control keys of the status message mask



Fig. 13: Example of a configuration mask

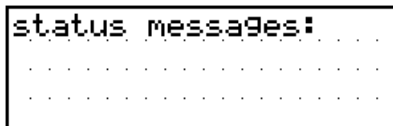


Fig. 14: Example of a status message mask

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.

<u>Variable Type</u>	<u>Size</u>	<u>Range of Values</u>
Bit	1 bit	[0, 1]
Byte	1 byte	[-128 to +127]
Byte	1 byte	[0 to 255]
Word	2 bytes	[-32768 to +32767]
Word	2 bytes	[0 to 65535]
Lword	4 bytes	[-2147483648 to +2147483647]
Lword	4 bytes	[0 to 4294967295]
Lword	4 bytes	$[\pm 1.2 * 10^{-38} \text{ to } \pm 3.4 * 10^{+38}]$
ASCII	42 bytes	[0 to 255]

3.6.1 Output Variables

Output variables are numerical or alphanumeric 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 millimetres, 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.

<u>Format</u>	<u>Data type</u>	<u>formatting</u>	<u>scaling</u>
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:

<u>Operand</u>	<u>Range of values</u>
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:

<u>Operand</u>	<u>Range of values</u>
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:

$$\boxed{\text{Terminal Output value}} = \frac{\boxed{\text{PLC variable value}} \times \text{Factor}}{\text{Divisor}} + \text{Summand}$$

Fig. 15: Formula 1

Output Representation Types:

Certain types of variables can be recognised 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:

<u>Binary</u>	<u>Hex</u>	<u>Decimal</u>	<u>Selection Text (Coded Text)</u>
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.

	<u>Representation without Leading Zeros</u>	<u>Representation with Leading Zeros</u>
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 1 / 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.

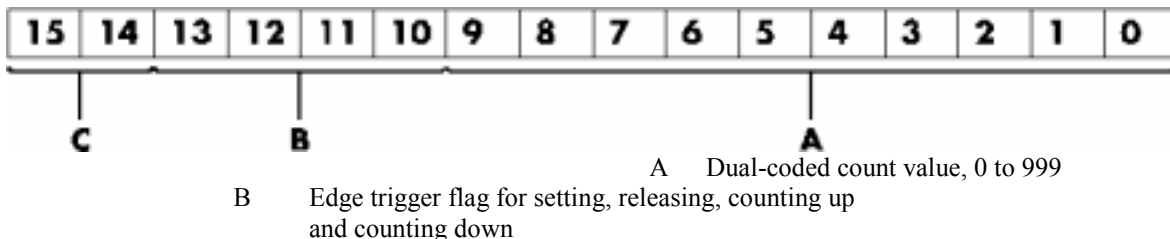
The example lists the output values with one post-decimal place (fractional digit). This produces the following to be displayed on the operating terminal:

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.



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. visualisation 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:

<u>Value</u>	<u>Text</u>
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:

<u>Value</u>	<u>Text</u>
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 visualisation 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:

<u>Value</u>	<u>Image Name</u>
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:

120	
34	
7	
1201	

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.

<u>Bit 3</u>	<u>Bit 2</u>	<u>Bit 1</u>	<u>Bit 0</u>	
0	1	0	0	Direction = 76543210 (TSwin)
<u>Bit 0</u>	<u>Bit 1</u>	<u>Bit 2</u>	<u>Bit 3</u>	
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 centre 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.

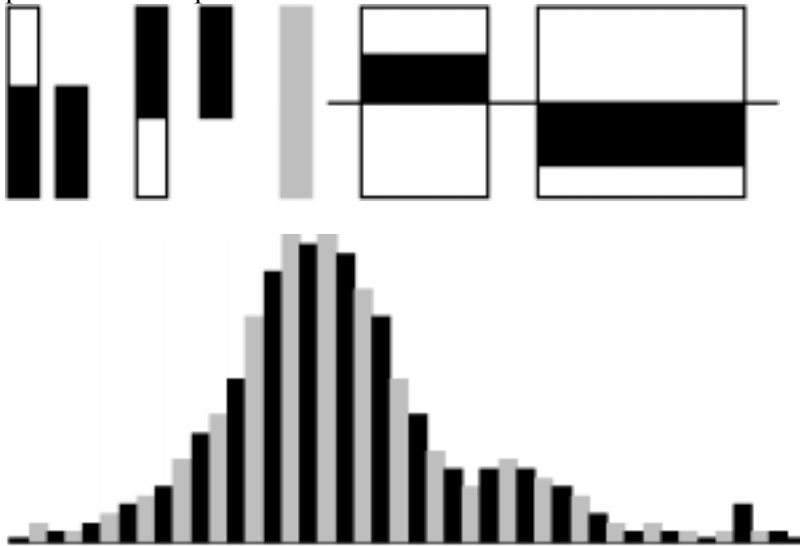


Fig.17: Different forms of bar representation

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
- Visualised monitoring of limit values
- Histograms
- Filling-level indications

Example of a filling-level indication:

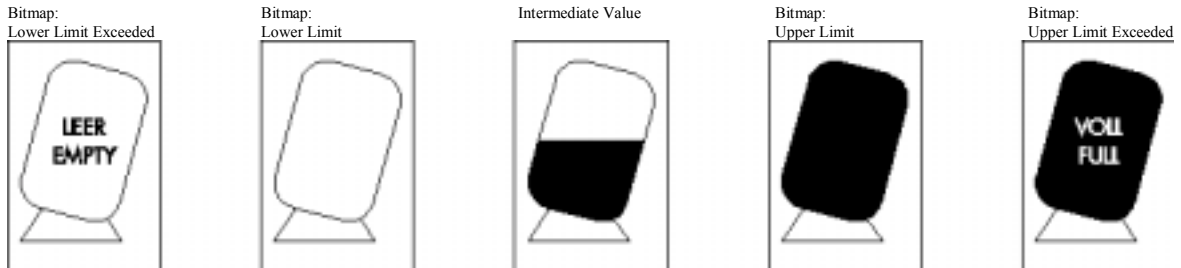


Fig. 18: Example of a filling-level indication

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.

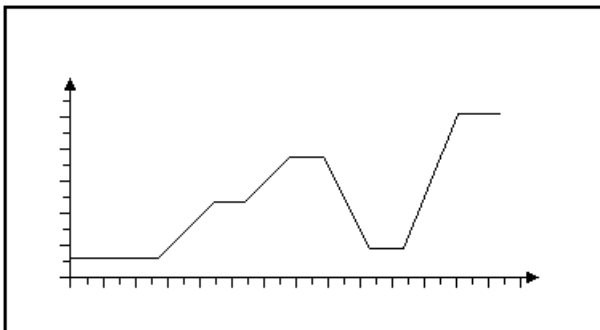


Fig. 19: Example of a curve representation

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 visualisation 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!
---	------------------------------

Alfred
Bernd
Detlef
Erwin

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.

$$\boxed{\text{PLC Variable value}} = \frac{\boxed{\text{Terminal input value}} - \text{Summand}}{\text{Factor}} \times \text{Divisor}$$

Fig. 20: Formula 2

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

$$\boxed{(\text{Input value} \times \text{Factor}) < (\text{Pos. Upper limit} - \text{Divisor}/2)}$$

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.

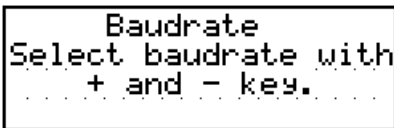
If the system message texts "Value too large" and "Value too small" are deleted, the following applies:

- 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.

A rectangular box containing a monospaced font help text for a variable named 'Baudrate'. The text is centered and reads: 'Baudrate' on the first line, 'Select baudrate with' on the second line, and '+ and - key.' on the third line, with dotted lines on either side of the last line.

```
Baudrate
Select baudrate with
+ and - key.
```

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 lose its terminal-internal function.

The following subchapters list the system variables, organised 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 reinitialisation (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 temp. 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 bit
(2) 2 bit

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 Baud
(1) 600 Baud
(2) 1200 Baud
(3) 2400 Baud
(4) 4800 Baud
(5) 9600 Baud
(6) 19200 Baud
(7) 38400 Baud
(8) 57600 Baud (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 power-up 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: 0 to 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) 5bits/Char
(1) 6bits/Char
(2) 7bits/Char
(3) 8bits/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 bit
(2) 2 bit

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 Baud
(1) 600 Baud
(2) 1200 Baud
(3) 2400 Baud
(4) 4800 Baud
(5) 9600 Baud
(6) 19200 Baud
(7) 38400 Baud
(8) 57600 Baud (BT35 / BT35C / TP35 only)

ComHandshakeB

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

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

ComDefaultB

This variable can be used to program new parameters for the interface (X3).

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.

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

ReputRepTextP21

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

ReputRepTextP41

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

ReputRepTextP61

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
(1) 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

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

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

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 the key *Plus*.

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 the key *Minus*.

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 the key *Enter*.

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 the key *Release*.

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 authorisation. 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 authorisation

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

PasswdInactive

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 initialisation)
(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), positive decimal number
Possible values: (0) Initial state
(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:

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

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

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	<ol style="list-style-type: none">1. Variable currently selected: Value is deleted, a new value can be entered2. Cursor has been moved within a positive value: No change3. Cursor has been moved within a negative value: Deletes the negative sign
Key: Minus	<ol style="list-style-type: none">1. Variable currently selected: Value is deleted, negative sign is entered at the least significant position, a new value can be entered2. Cursor has been moved within a positive value: Negative sign is placed in front of the value3. 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 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	<ol style="list-style-type: none">1. Standard Direct input of numerical values2. Mix-Mode Direct input of numerical values3. Incremental No Function
Key: Plus	<ol style="list-style-type: none">1. Standard No Function2. 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	<ol style="list-style-type: none">1. Standard No Function2. 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 alphanumeric keys, alphanumeric 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	<p>Shifts to the previous mask variable if the cursor is located at a single variable or at the first row variable.</p> <p>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.</p> <p>When shifting to a new column variable, the cursor is placed at the bottom row variable (i.e. the last row variable).</p> <p>When shifting to a single variable, the row number remains unchanged.</p>
Key: Cursor down	<p>Shifts to the next mask variable if the cursor is located at a single variable or at the last row variable.</p> <p>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.</p> <p>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).</p> <p>When shifting to a single variable, the row number remains unchanged.</p>
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.

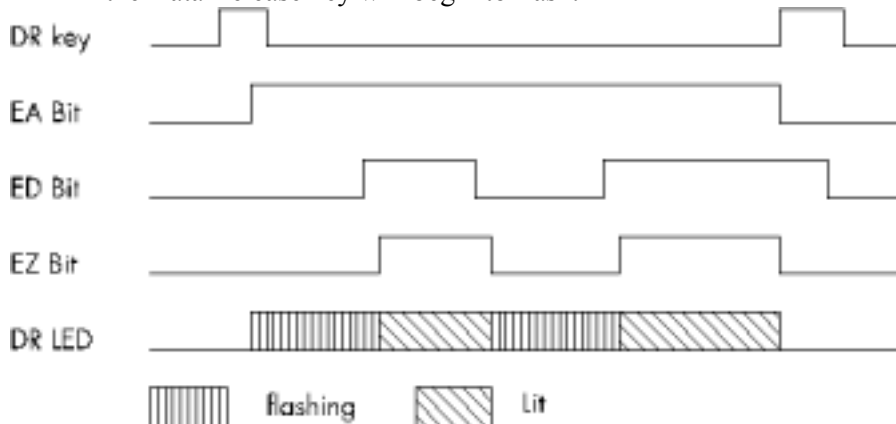


Fig.23: Pulse diagram - external data release

- 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
- DF-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.

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.

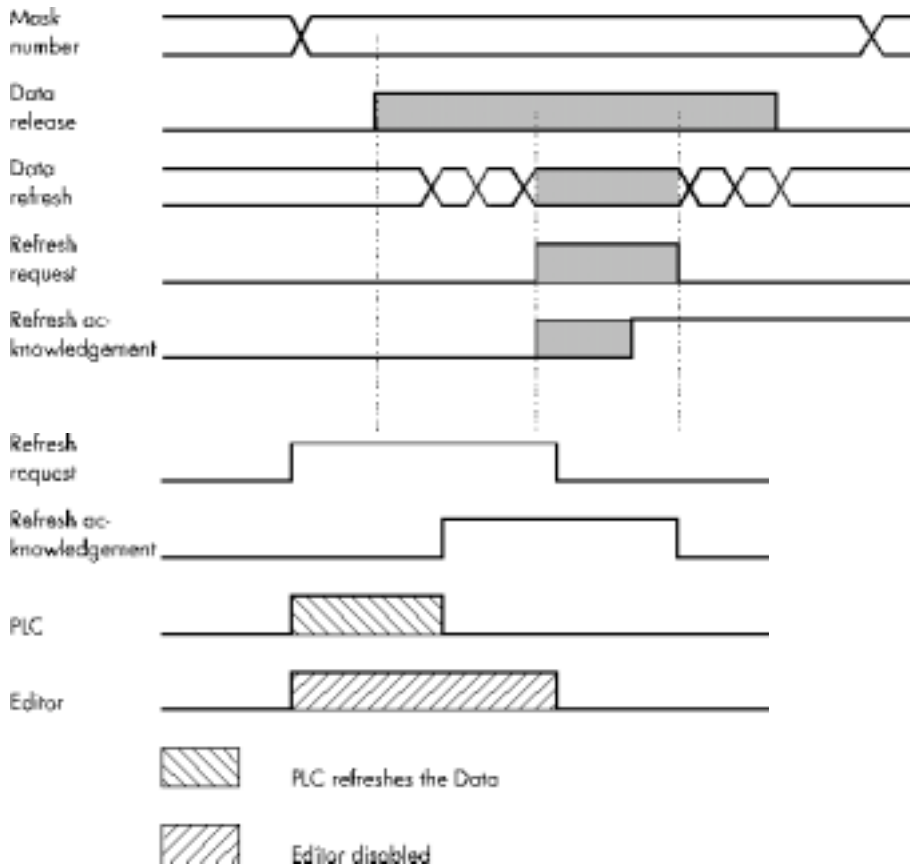


Fig. 24: Pulse diagram - PLC handshake

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.

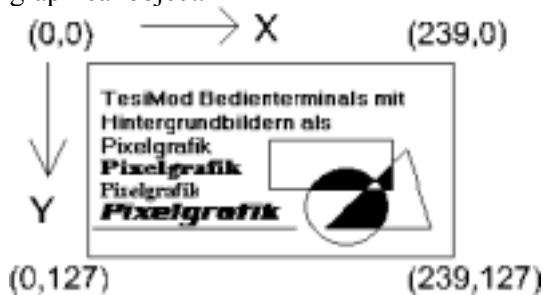


Fig. 25: Example of coordinates

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 organised 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 <i>clamp</i> 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 <i>shaft</i> 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 organised 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*).

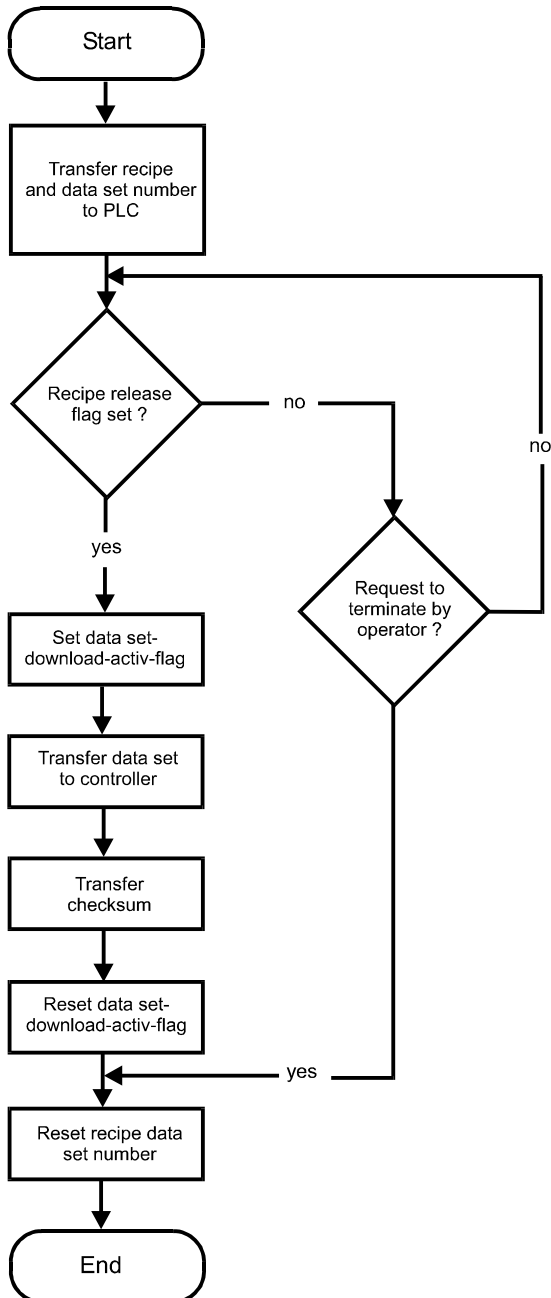


Fig. 26: Data transfer sequence to the controller

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).

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.

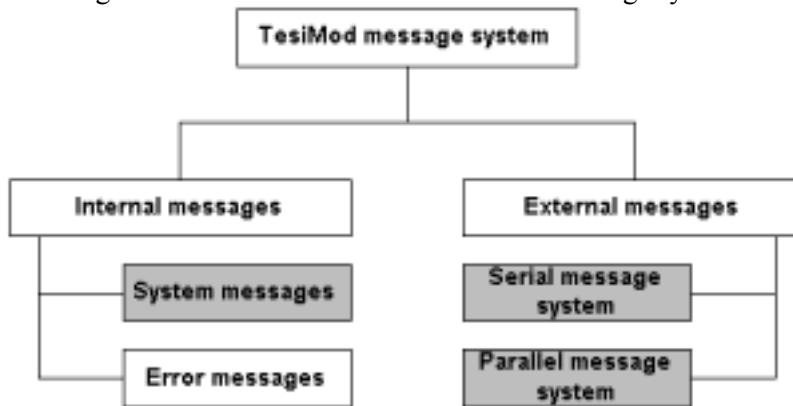


Fig. 27: Structure of the TesiMod message system

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 authorisations (view and edit level) are reset.

System Message 13 "Password unchanged"

The operator did not enter the same new password two times in a row.

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 authorisation.

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.

```
COMMUNICATION ERROR
CODE                X
SUBCODE             XX
RETRIES             XXXX
```

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.

```
ADDRESS ERROR
.....
.....
```

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.

```
FLASH MEMORY FAILURE
.....
.....
```

Is displayed during a download if the Flash-Eprom can not be programmed. This message indicates that the application memory is defective. The starting address of the invalid line in the S3 file is specified in hexadecimal format.

```
CHECKSUM ERROR
.....
.....
```

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.

```
TURN POWER OFF
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.

```
!!!!!! ERROR !!!!!
NO PROTOCOL-DRIVER
IN MASK DESCRIPTION
FOUND
```

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.

```
!!!!!! ERROR !!!!!
PLC TYPE MISMATCH
BETWEEN TERMINAL
AND MASK DESCRIPTION
```

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.

```
KEYBOARD ERROR
PLEASE RELEASE KEY
```

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 keyboard!

```
INITIALIZING
MESSAGE BUFFER
```

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

```
ERASE FLASH EPROM
```

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

```
FLASH IS ERASED
```

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

```
DOWNLOAD
```

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

```
DOWNLOAD 1
FLASH XXX kByte
.....
```

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).

```
DOWNLOAD 2
.....
```

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).

```
AUTO REBOOT
.....
```

The operating terminal will reboot within the next few seconds.

```
INITIALIZING
CPU XX MHz
FLASH XXX kByte
XXXXXXXX YYYYYYYY
```

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

```
IDENTIFY MEMORY-TYP
.....
```

The Flash memory type used is being identified.

```
!!! HIGH VOLTAGE !!!
.....
```

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

```
ERROR ASYNCHRONOUS
SERIAL I/O UNIT 0
.....
```

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

```
SUCONETK-MODUL
HARDWARE-VERSION
NOT CONFORM TO
DRIVER VERSION
```

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.

```
KEYBOARD-MODUL
HARDWARE-VERSION
NOT CONFORM TO
DRIVER VERSION
```

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

```
FIRMWARE UPDATE
SUCCESSFUL
AUTO-REBOOT
```

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

```
SYSTEM ERROR
CODE : XX
SUBCODE : XXXX
RETRIES : XXXXX
```

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 EUROTHERM Antriebstechnik GmbH, Im Sand 14, 76669 Bad Schönborn, hotline no.: +49 72534 04 0.

```
UNEXPECTED INTERRUPT
NR = .....
IP = .....
CALL HOTLINE
```

An unexpected interrupt has occurred. Make a note of the interrupt number (NR) and the level of the program counter (IP) and contact EUROTHERM Antriebstechnik GmbH, Im Sand 14, 76669 Bad Schönborn, hotline no.: +49 72534 04 0.

```
FLASH NOT ERASABLE
.....
.....
```

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

```
WRONG S3 FILES
.....
.....
```

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

```
NO FLASH EPROM
.....
.....
```

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

```
FLASH CHECKSUM ERROR
.....
.....
```

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.

```
TERMINAL-TYP IS XXXX
.....
.....
```

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.

```
MEMORY IS FLASH XXXX
.....
.....
```

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.

```
FATAL ERROR
CODE : XXXX
SUBCODE: XXXX
CALL HOTLINE
```

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.

```
FIRMWARE NOT CONFORM
TO HARDWARE
FIRMWARE 1
HW-VERSION 2
```

If this error message is displayed, contact EUROTHERM Antriebstechnik GmbH, Im Sand 14, 76669 Bad Schönborn, hotline no.: +49 72534 04 0. 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.

```
DATASET STORAGE
FAILURE
.....
.....
```

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 IBT 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 **7FFE_H** via the serial message channel). The internal message buffer is deleted if the deletion variable contains the bit pattern **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 recognised?

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 $7FFF_H$ 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.

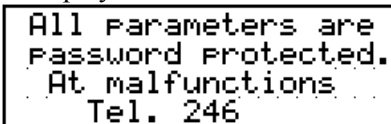
To be able to use the event-controlled transmission (also for polling times of 0,0 s), the parameters file must contain the symbolic variable name and the length of the status messages.

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.



```
All Parameters are
Password Protected.
At malfunctions
Tel. 246
```

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 labelling 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 labelling 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.

<u>Bit is on/off</u>	<u>Bit flashes</u>	<u>Status of the LED</u>
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 authorisation levels and the initialisation values for the authorisation 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 (with EUROTHERM drives address “W 12”)
- Variable for data set number (with EUROTHERM drives address “W 13”)

The following parameters are required to transfer data sets from the controller (on request):

- Variable for recipe number (with EUROTHERM drives address “W 10”)
- Variable for data set number (with EUROTHERM drives address “W 11”)

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.

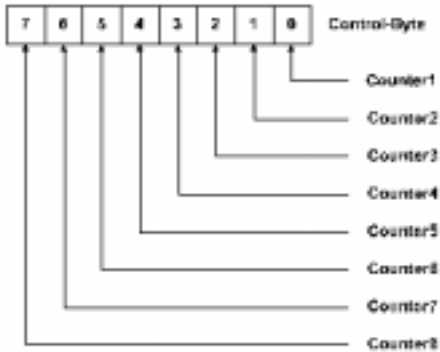


Fig. 29: Structure of the control byte

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.

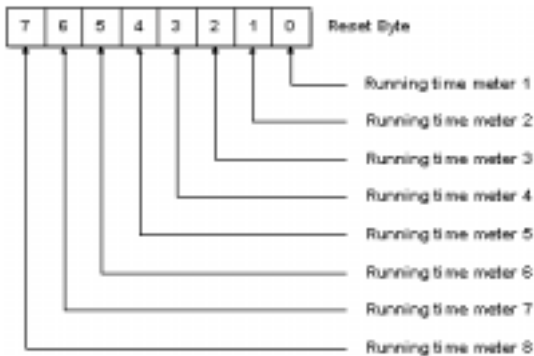


Fig. 30: Structure of the reset byte

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:

0	0	Output Off
1	0	Output On
1	1	Output Flashes

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 initialisation 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:

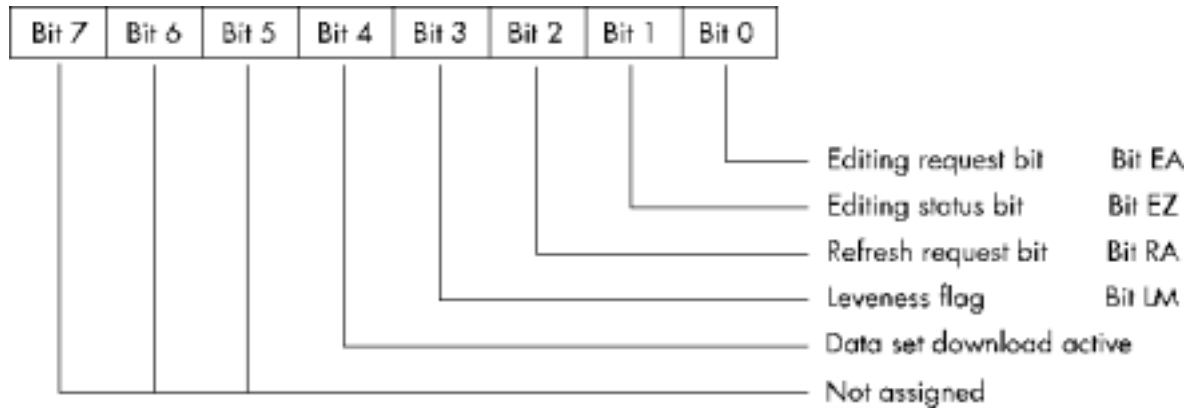


Fig. 34: Structure of the Read coordination byte

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:

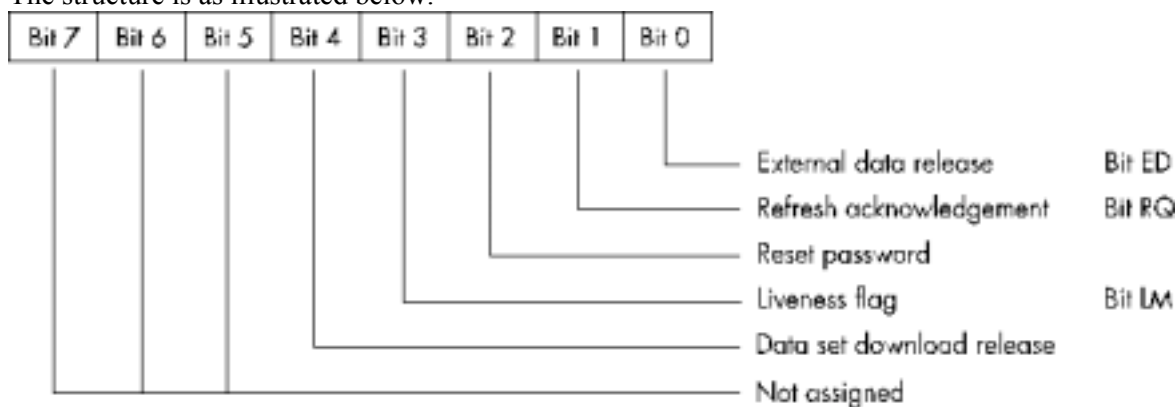


Fig. 35: Structure of the Write coordination byte

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** (3.20.4).

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	LED1	LED1	LED2	LED2	LED3	LED3	LED4	LED4
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +4	LED5	LED5	LED6	LED6	LED7	LED7	LED8	LED8
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +5	LED9	LED9	LED10	LED10	LED11	LED11	LED12	LED12
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +6	LED13	LED13	LED14	LED14	LED15	LED15	LED16	LED16
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +7	LED17	LED17	LED18	LED18	LED19	LED19	LED20	LED20
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +8	LED21	LED21	LED22	LED22	LED23	LED23	LED24	LED24
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +9	LED25	LED25	LED26	LED26	LED27	LED27	LED28	LED28
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +10	LED29	LED29	LED30	LED30	LED31	LED31	LED32	LED32
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +11	LED33	LED33	LED34	LED34	LED35	LED35	LED36	LED36
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +12	LED37	LED37	LED38	LED38	LED39	LED39	LED40	LED40
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +13	LED41	LED41	LED42	LED42	LED43	LED43	LED44	LED44
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Byte address +14	LED45	LED45	LED46	LED46	LED47	LED47	LED48	LED48
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	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	LED1	LED1	LED2	LED2	LED3	LED3	LED4	LED4	LED5	LED5	LED6	LED6	LED7	LED7	LED8	LED8
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 3	LED9	LED9	LED10	LED10	LED11	LED11	LED12	LED12	LED13	LED13	LED14	LED14	LED15	LED15	LED16	LED16
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 4	LED17	LED17	LED18	LED18	LED19	LED19	LED20	LED20	LED21	LED21	LED22	LED22	LED23	LED23	LED24	LED24
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 5	LED25	LED25	LED26	LED26	LED27	LED27	LED28	LED28	LED29	LED29	LED30	LED30	LED31	LED31	LED32	LED32
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 6	LED33	LED33	LED34	LED34	LED35	LED35	LED36	LED36	LED37	LED37	LED38	LED38	LED39	LED39	LED40	LED40
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 7	LED41	LED41	LED42	LED42	LED43	LED43	LED44	LED44	LED45	LED45	LED46	LED46	LED47	LED47	LED48	LED48
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	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.

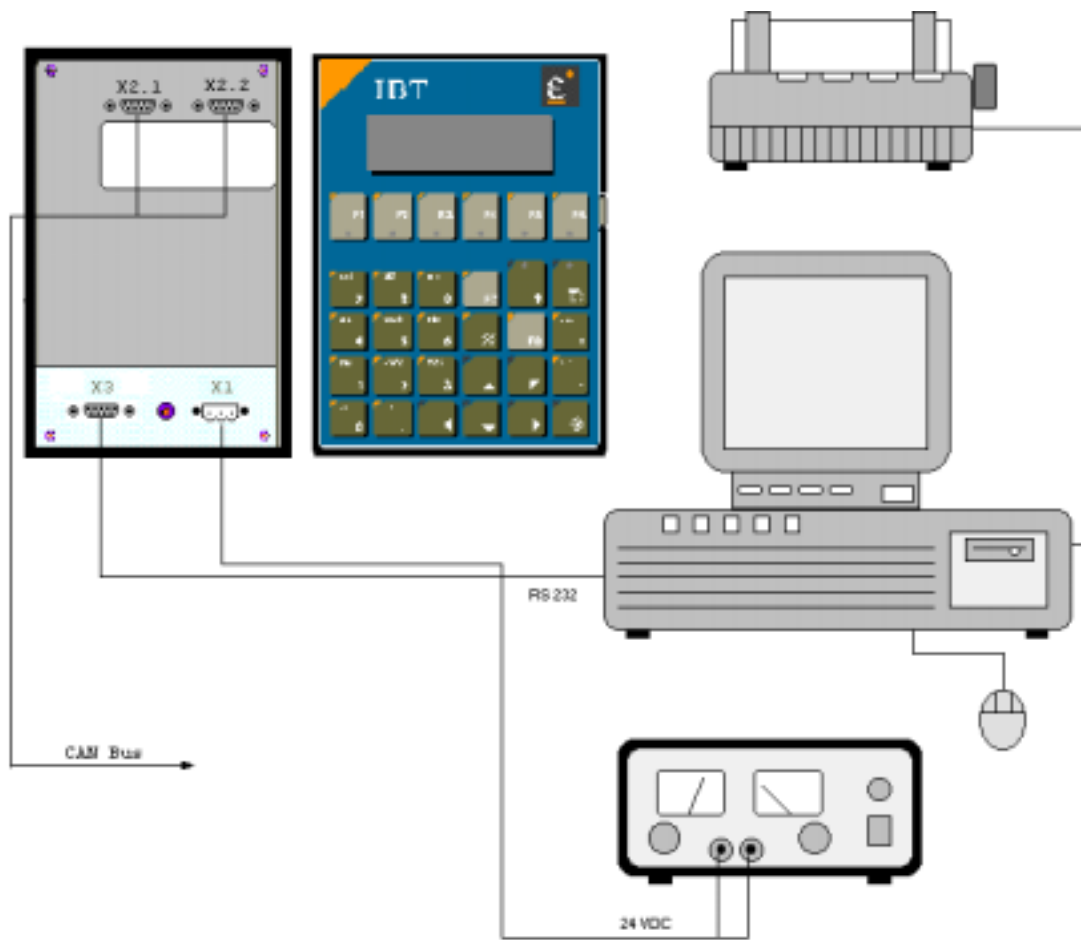


Fig. 39: System configuration

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).

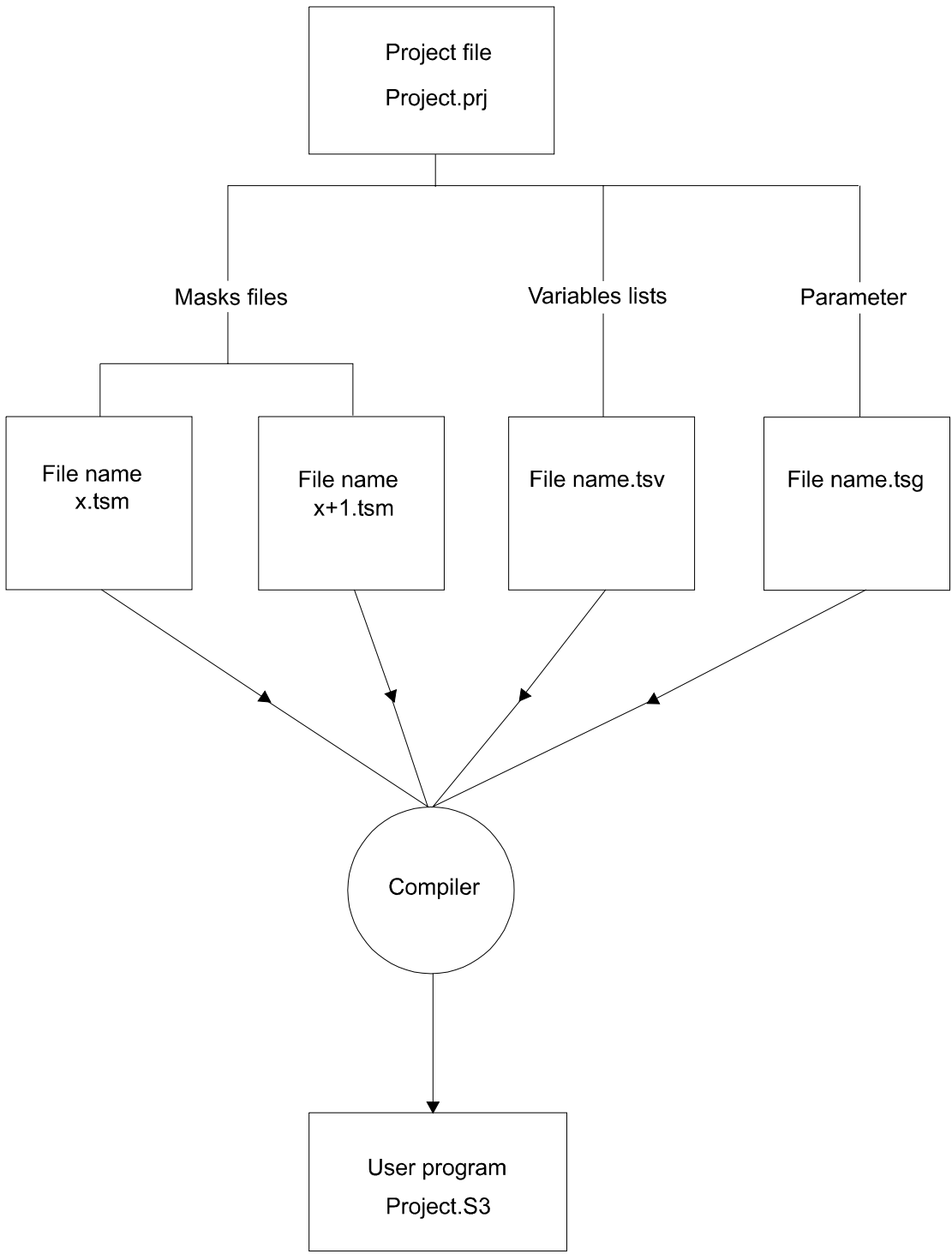


Fig. 40: Project files of the TSdos programming system

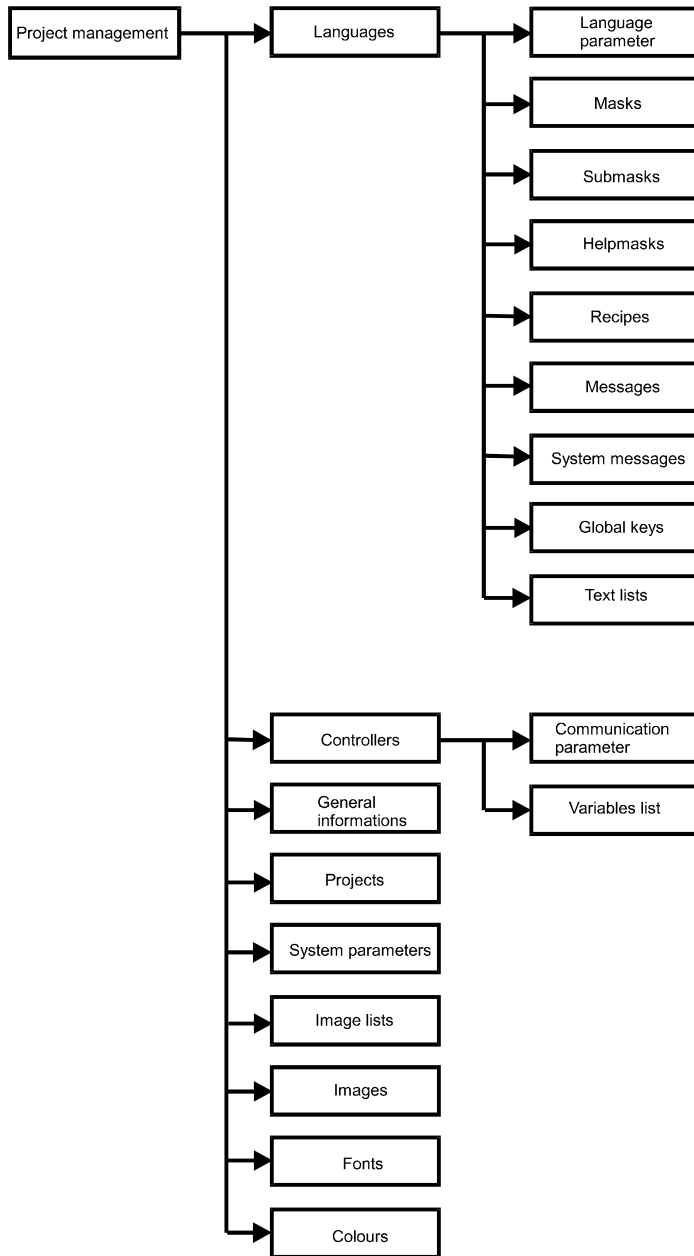


Fig. 41: Structure of the TSwIn programming system

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 organised 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 our 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:

<u>VALUE</u>	<u>FUNCTION</u>
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

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

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:

```
FLASH MEMORY FAILURE  
.....  
.....
```

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):

```
DOWNLOAD .....  
.....  
.....
```

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:

```
ERASE FLASH EPROM . . .  
.....  
.....
```

After the Flash-Eprom has been erased, this is indicated by the following message:

```
FLASH IS ERASED . . . .  
.....  
.....
```

The terminal now automatically activates the download mode and the following message appears:

```
DOWNLOAD . . . . .  
.....  
.....
```

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:

```
READY . . . . .  
.....  
.....
```

After the transfer, the terminal immediately starts the initialisation 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 initialisation 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

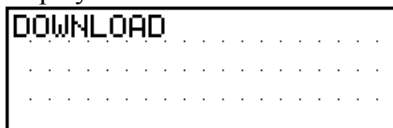
A Flash memory or UV-Eprom can be used as an application memory. The terminal comes fitted with a 128 kByte Flash memory. This Flash memory is installed in a 32-pin DIL-precision socket and can be replaced with the aid of an extraction tool. A standard UV-Eprom of appropriate size and access time can 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:



```
DOWNLOAD .....  
.....  
.....
```

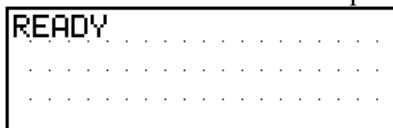
The terminal is now ready to receive a new user description via interface X3 and store it in the Flash memory.

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 initialisation 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

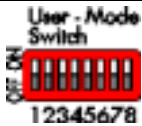


Fig. 42: Mode selector switch

The following message is displayed once power has been applied the terminal:

```
TURN POWER OFF
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

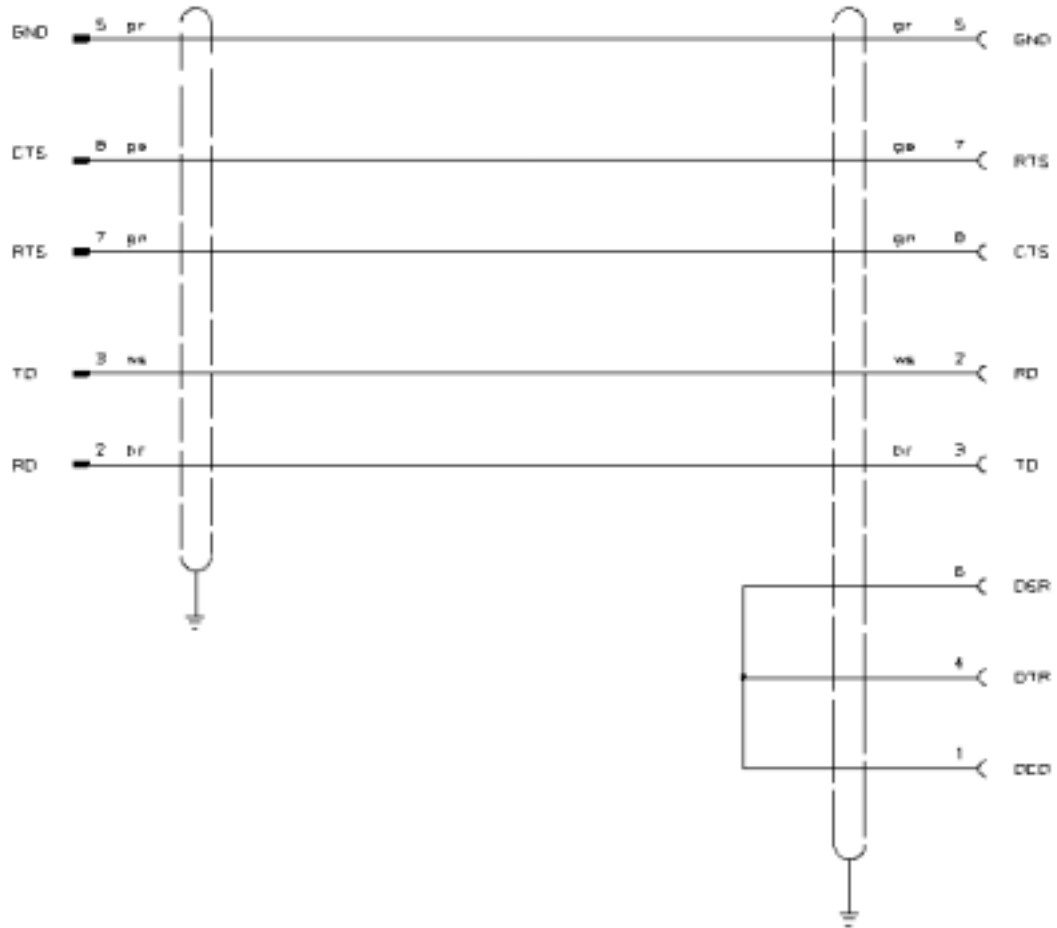
- message data in the message memory
- all RAM-data sets that have not been backed-up
- all system parameters that have been edited online (values of the system variable, interfaces, passwords, etc.) will be lost!

3.28.7 Download Cable

Pinning of the cable connecting interface X3 of the operating terminal to the PC (except IBT).

TesiMod
Operating Terminal

Personal
Computer



D-Subminiature
Male Connector
9-pin

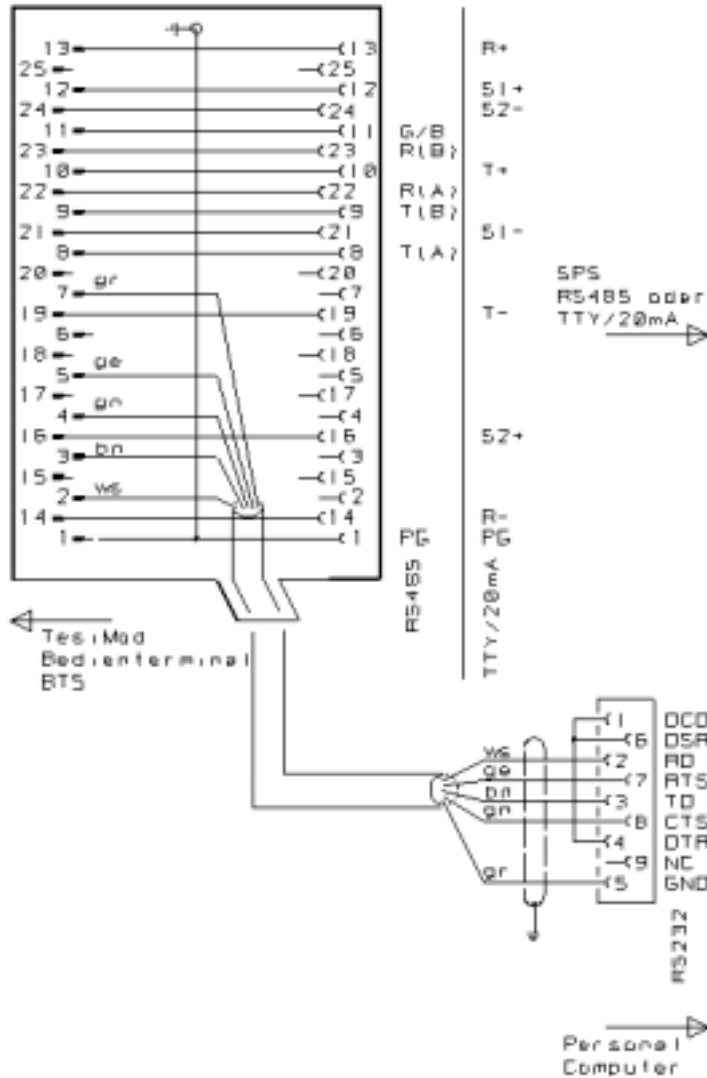
D-Subminiature
Female Connector
9-pin

The shield is connected to the metal casing on both ends.

Pinning of the cable connecting interface X3 of the operating terminal to the PC (BT2 / IBT only).

TesiMod
Operating Terminal IBT / BT2

Personal
Computer



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.

5.22 TesiMod - CAN/CANopen

Abbreviations and concepts:

CAL CAN Application Layer, standardized by CiA (CiA Draft Standard 102, 201, 205, 207)

CiA CAN in Automation International Users and Manufacturers Group e.V.

CANopen Communication Profile, CiA Draft Standard 301

5.22.1 General

TesiMod operating terminals can easily be connected to CAN bus systems thus making TesiMod operating terminals the perfect man machine interface (MMI) for your CAN/CANopen application.

The CAN bus is designed as a high speed bus in accordance with ISO DIS 11898. All connections are electrically isolated.

The implementation of CAN in TesiMod operating terminals closely conforms with the specification of the **CANopen Communication Profile**.

CANopen uses a subset of the CAL communication services.

Since a communication profile for man-machine-interfaces has not yet been standardized by CiA, Süttron electronic GmbH has developed its own **indirect process data communication**.

5.22.2 Technical Description

Data traffic is determined by the CANopen communication profile. Based on this communication profile, additional profiles are defined which are used to access modules of the same type in the same way.

A total of eight different communication relationships can be defined with the programming system.

5.22.2.1 Data Objects

All data to be transferred within a CANopen network are accessed as data objects via an object directory. With CANopen, process data objects (PDO) are created for transmitting and receiving data. The PDOs, or transmission packets, can contain multiple data objects (up to 8 bytes).

PDOs for transmitting data are referred to as request-PDOs (PDO tx). PDOs for receiving data are called response-PDOs (PDO rx).

Services of the lowest CAL layer, which require no protocol overhead, are used for transmitting PDOs.

5.22.2.2 Identifiers

With CANopen, an identifier is allocated to each PDO.

Each CANopen station comprises at maximum two transfer-PDOs and two receive-PDOs, i. e. four identifiers in total.

Due to this restriction, only 16 bytes can be sent or received at one time.

5.22.3 Indirect Process Data Communication

Communication with operating terminals involves transmission of data volumes in the CANopen network of more than 16 bytes and can therefore not be transmitted in two PDOs.

For this reason, the operating terminal does not access a data object directly but via special communication objects called request and response objects. These objects tell the communication partner which data object to read from or to write to.

5.22.3.1 Data Exchange Sequence

The **operating terminal functions as a client**: it initiates all services required for its operation.

The **communication partner acts as a server**: it merely responds to requests from the operating terminal.

Thus the operating terminal transmits request objects while the communication partner transmits response objects.

After receiving a request object from the operating terminal, the communication partner interprets the first 4 bytes for data direction and accessed data object.

In the case of a write request, it stores the transmitted data and acknowledges receipt with an empty response object.

In the case of a read request, it transmits the requested data to the operating terminal via a response object.

Request-PDOs and response-PDOs always exist in pairs. Each pair stands for one communication relationship between the operating terminal and **one** station.

With TesiMod operating terminals, up to eight communication relationships can be set up.

5.22.4 Request and Response Object Structure

Request and response objects are each 8 bytes in size. The first four bytes of these objects contain the definition of the accessed data object, the remaining 4 bytes contain the data.

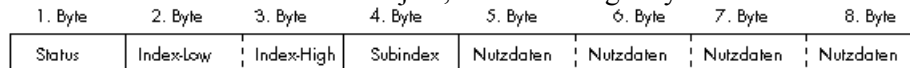


Fig. 1: Structure of request and response objects

5.22.4.1 Status Byte Structure

The structure of the status byte within the request and response object is as follows:

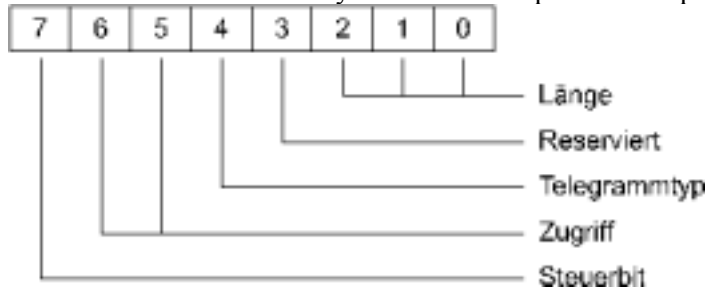


Fig. 2: Structure of the control byte

Length: bit 0 to bit 2

1, 2	Bit access	Length of the bit mask for setting and deleting bits
1, 2, 3, 4	Byte access	Number in bytes
2, 4	Word access	Number in bytes
4	Double-word access	Number in bytes

Telegram type: bit 4

0	Request telegram
1	Response telegram

Access: bit 5 to bit 6

0	Bit access	Bit access in write mode (set/reset)
1	Byte access	Read or write byte
2	Word access	Read or write word
3	Double-word access	Read or write double word

Control bit: bit 7

Request object

0	Write
1	Read

Response object

0	ok
1	Error from controller during telegram processing The first data byte contains the error number

5.22.4.2 Index Bytes

This value specifies the index from the communication partner's object directory.
 The index points to:

- one variable without subindex
- the beginning of an array
- the beginning of a record.

If CANopen is not used, the numbering between communication partners can be applied as desired.

5.22.4.3 Subindex Byte

This value specifies the subindex from the communication partner's object directory.
 The subindex points to one variable (basic CAN variable) of the size of the access.

If CANopen is not used, the numbering between communication partners can be applied as desired.

5.22.4.4 User Data Bytes of Request/Response Object

User data, bit-oriented

In a byte address:

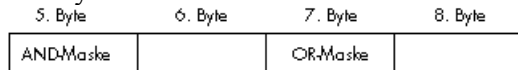


Fig. 3: User data in a byte address

In a word address:

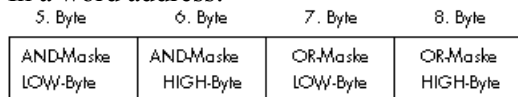


Fig. 4: User data in a word address

The bit masks are to be linked in the controller with the specified addresses using the OR and AND logical operations, where AND-masks delete bits and OR-masks set bits.

User data, byte-oriented

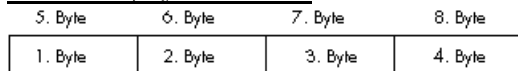


Fig. 5: User data, byte-oriented

User data, word-oriented

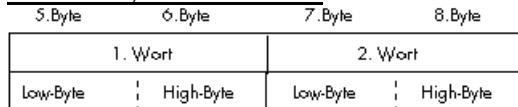


Fig. 6: User data, word-oriented

User data, double-word oriented

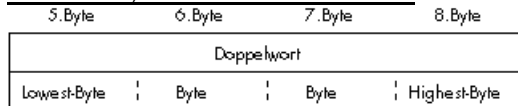


Fig. 7: User data, double-word oriented

5.22.4.5 Response Object with Error

If communication errors occur, a response object with the following structure is returned:

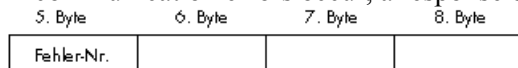


Fig. 8: Structure of a response object with error

5.22.5 Tasks of the Communication Partner

The operating terminal's communication partner functions as a server. As a server, the communication partner is responsible for interpreting incoming telegrams and for replying with a response object. This procedure must be performed within the waiting time specified in the protocol parameters.

A response object must be transferred in reply to every request object.

In the case of an error, the communication partner simultaneously transmits an error number to the operating terminal.

5.22.5.1 Error-Free Communication Sequence

- The status byte of the request object is copied to the response object with the following change:
Bit 7 control bit = 0 no error
Bit 4 telegram type = 1 response object
- Index and subindex are copied from the request object to the response object.
- During write operations of the operating terminal, the user data must be copied to the accessed address
- When the operation terminal reads data, the data must be copied from the accessed address to the address of the user data bytes.

5.22.5.2 Faulty Communication Sequence

If the communication partner detects an error in the request object contents, this can be reported to the operating terminal via an error message in the response object.

The error numbers can be freely defined by the user. The operating terminal always displays the communication error 100. The subcode of the error message contains the error number.

- The status byte of the request object is copied to the response object with the following change:
Bit 7 control bit = 1 error detected
Bit 4 telegram type = 1 response object
- Index and subindex are copied from the request object to the response object.
- Enter error number in byte 5 (is displayed on the operating terminal as a subcode)

5.22.6 Protocol Parameters

5.22.6.1 Baud Rate

The following baud rate settings can be selected:

- 20 kBaud
- 125 kBaud
- 500 kBaud
- 1 MBaud

The default baud rate setting is 20 kbit/s.

5.22.6.2 Max. Waiting Time for Response

The communication partner must return a response PDO within the maximum waiting time for response. Otherwise an error message is issued.

The waiting time can be between 0 ms and 65536 ms.

The default setting is 1000 ms.

5.22.6.3 Delay until Connection Set-up

The TesiMod operating terminal waits for the specified length of time before initiating communication. This ensures that a faulty communication is not initiated due to the bus system not being ready for operation.

The waiting time setting can be between 1 s and 255 s.

The default value is 5 s.

5.22.6.4 Terminal Module Number

In CANopen mode, allocation of the identifiers for request and response objects are computed using the module number.

Values between 0 and 127 can be entered for the module number.

The default value is 1.

5.22.6.5 CANopen Network Management Services

The programming system allows the user to define whether CANopen network management services are to be used.

- NMT service with identifier = 0
 - Byte 1 = 1 Activate station (node)
 - Byte 1 = 2 Deactivate station (node)
 - Byte 1 = 3 Deactivate station (node)
 - Byte 1 = 128 Deactivate station (node)
 - Byte 1 = 129 Reset station (node)
 - Byte 1 = 130 Reset communication at station (node)
- At start-up, the operating terminal issues an emergency message with the identifier = 128 + terminal module number

5.22.7 Communication Relationships

TesiMod operating terminals allow eight independent communication relationships to be defined. See figure 10 in chapter 5.22.11.

5.22.7.1 Mode

Identifiers for the communication relationships can be allocated in two ways:

- Manual: The request and response identifier is entered directly into the relevant fields.
- CANopen: In this case, identifiers are distributed as defined in the CANopen Communication Profile DS301.
The module number of the slave, the communication partner in the CANopen network, is added to a base number and entered as the identifier.

5.22.7.2 Connection Partners

A connection partner can be selected in CANopen mode only.

The selection of a connection partner determines the base identifier. Possible selections for the connection partner are **master** or **slave**.

- Master: The terminal communicates with the CANopen network master.

The first PDO channel is used.

The module number of the operating terminal is added to the base number.

- Slave: The operating terminal communicates with another slave in the CANopen network.

The second PDO channel is used.

The module number of the other slave is added to the base identifier.

5.22.7.3 Module Number

A module number can be entered only in CANopen mode in connection with a communication relationship with another slave.

Here, the module number of the communication partner can be entered directly.

5.22.7.4 Request Identifiers

Request identifiers can be entered in manual mode only.

In CANopen mode, the request identifier is computed using the base identifier and module number while in manual mode, the identifier for the request object can be entered directly.

5.22.7.5 Response Identifiers

Response identifiers can be entered in manual mode only.

In CANopen mode, the response identifier is computed using the base identifier and module number while in manual mode, the identifier for the response object can be entered directly.

5.22.7.6 Address Type

When accessing data of the communication partner, the operating terminal must know the addressing method of the data area to be accessed. The operating terminal requires this information for internal address calculation.

Values that can be entered for the address type are as follows:

- Byte address 1
- Word address 2
- Double-word address 4

5.22.8 Poll Area

Conditions to be observed with respect to the data area of the poll area are:

- the controller must be able to manipulate single bits (activation of the LEDs)
- the operating terminal must be able to read this data area in word mode
- access by the operating terminal can be **in word mode only**
- the address type of the data area in the controller must be a **byte or word address**

Example for addressing the poll area:

Byte-addressed

Word address + 0	Index 100	Subindex 0
Word address + 1	Index 100	Subindex 2
Word address + 2	Index 100	Subindex 4
Word address + 3	Index 100	Subindex 6
Word address + 4	Index 100	Subindex 8
Word address + 5	Index 100	Subindex 10

Word-addressed

Word address + 0	Index 100	Subindex 0
Word address + 1	Index 100	Subindex 1
Word address + 2	Index 100	Subindex 2
Word address + 3	Index 100	Subindex 3
Word address + 4	Index 100	Subindex 4
Word address + 5	Index 100	Subindex 5

	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	LED1	LED1	LED2	LED2	LED3	LED3	LED4	LED4	LED5	LED5	LED6	LED6	LED7	LED7	LED8	LED8
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 3	LED9	LED9	LED10	LED10	LED11	LED11	LED12	LED12	LED13	LED13	LED14	LED14	LED15	LED15	LED16	LED16
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 4	LED17	LED17	LED18	LED18	LED19	LED19	LED20	LED20	LED21	LED21	LED22	LED22	LED23	LED23	LED24	LED24
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 5	LED25	LED25	LED26	LED26	LED27	LED27	LED28	LED28	LED29	LED29	LED30	LED30	LED31	LED31	LED32	LED32
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 6	LED33	LED33	LED34	LED34	LED35	LED35	LED36	LED36	LED37	LED37	LED38	LED38	LED39	LED39	LED40	LED40
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing
Word address + 7	LED41	LED41	LED42	LED42	LED43	LED43	LED44	LED44	LED45	LED45	LED46	LED46	LED47	LED47	LED48	LED48
	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing	on/off	flashing

Fig. 9: Poll area, word-oriented

5.22.9 Physical Interfacing

The optodecoupled interfaces X2.1 and X2.2 are available to integrate the operating terminal into a CAN structure. The CAN bus is designed as a high speed bus in accordance with ISO-DIS 11898.

Shielded, twisted-pair cables should be used as connecting cables. The CAN bus must be terminated at both ends by termination resistors.

The interface connector pin assignment on the operating terminal complies with the CiA Draft Standard 102.

Plug-in connector on the terminal: 9-pin SubminD female connector strip X2.1

Pin Assignment:

Pin	Ass.	Designation	Function
1	x	res	Reserved
2	x	CANL	CAN_L bus line (dominant LOW)
3	x	0V_C	CAN Ground
4	x	res	Reserved
5	x	res	Reserved
6	x	0V_C	CAN Ground
7	x	CANH	CAN_H bus line (dominant HIGH)
8	x	res	Reserved
9	x	res	Reserved

Plug-in connector on the terminal: 9-pin SubminD male connector strip X2.2

Pin Assignment:

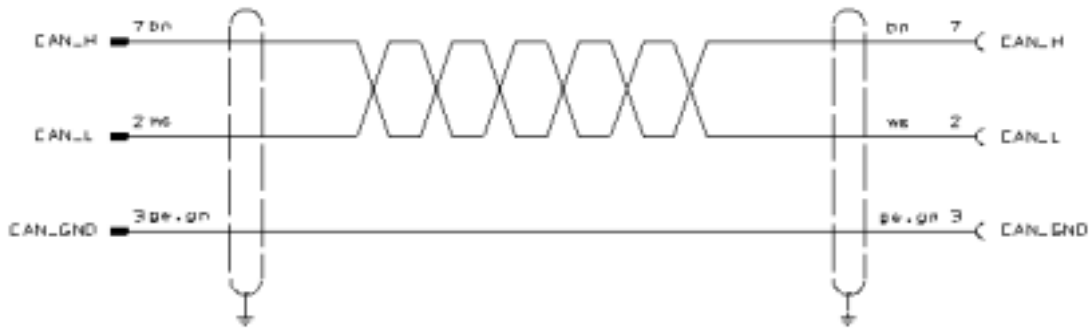
Pin	Ass.	Designation	Function
1	x	res	Reserved
2	x	CANL	CAN_L bus line (dominant LOW)
3	x	0V_C	CAN Ground
4	x	res	Reserved
5	x	res	Reserved
6	x	0V_C	CAN Ground
7	x	CANH	CAN_H bus line (dominant HIGH)
8	x	res	Reserved
9	x	res	Reserved

All signal lines are jumpered from X2.1 to X2.2. The connecting cables should be connected to all pins, including the reserved pins. In this way, the cables can still be used if the bus specifications are expanded in the future.

5.22.9.1 Connecting Cable TesiMod Operating Terminal - CAN Bus

TesiMod
Operating Terminal

Next
CAN Bus Station



D-Subminiature
Male Connector
9-pin

D-Subminiature
Female Connector
9-pin

The shield is connected to the metallic housing at both ends.

Contrary to the recommendations of the CiA Draft Standard 102, the cable is only equipped with the connections needed to meet the current communication requirements.

5.22.10 Error Messages

Code		
1	E_SLAVE_NOT_READY	Slave currently not ready
2	E_PROTOKOL	Sequence of packets
3	E_FRAME	Protocol frame error
4	E_TIMEOUT	Timeout error
5	E_CRC_BCC	CRC error
6	E_PARITY	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 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
50	E_CAN_ERROR	Error from CAN controller
		Error number in subcode
Subcode		
1		Stuff error
3		Terminal has no connection with bus
		No stations connected to bus
4, 5		Bus line has short-circuit
6		CRC error
Code		
51	E_RESPONSE-TIMEOUT	No response from communication partner No partner for request object No recipient for transferred identifier
53	E_NO_RESPONSE_BIT	Wrong response object Message without response bit
54	E_RESPONSE OF NO_REQUEST	Wrong response object Response without request
55	E_NO_HARDWARE	Terminal can not find CAN hardware CAN hardware in terminal missing or defective
60	E_RESET_FROM_MASTER	NMT 0 message with command 129
100	E_DATA_ERROR	Error from communication partner Subcode contains error number Subcode is freely definable

5.22.11 EUROTHERM specific addresses and definitions

For linking the terminals to the EUROTHERM drives, some special features are to be considered:

1. the Definition of the address preselections, and
2. the Definition of the address range and
3. the Definition of the Identifier.

5.22.11.1 Definition of the address preselections

Address preselection	Drive variables
BY	BIAS flags
W	Control range
DW	BIAS Variables

5.22.11.2 Definition of the address range

Preselection	Range (Index)	Subindex	Address width	Explanation
BY	0...255	0	1	BIAS flags 0...255
W	0	0	2	Poll area
W	9	0	2	Read Coordination-byte
W	10	0	2	Recipe number on IBT
W	11	0	2	Data set number on IBT
W	12	0	2	Recipe number of the IBT
W	13	0	2	Data set number of the IBT
W	15	0	2	Mask number
DW	0...255	0	4	BIAS Variables 0...255
DW	1000	0	4	Actual position 1
DW	1001	0	4	Actual position 2 (X40)
DW	1002	0	4	Actual position 3 (Absolute-value encoder)

5.22.11.2 Definition of the Identifier

During allocation of the Request and Response Identifiers, it is to be ensured that the following settings are kept:

- Mode „manual“
- Request-ID 0...2046
- Response-ID Request-ID+1
- Default-Adressweite 4

5.22.12 Examples of Communication Relationships

The TSwIn programming system provides a clear and convenient method of defining eight communication relationships. The figure below shows the TSwIn panel used to define the relationships and illustrates the possible settings.



Fig.10: Communication relationship settings in TSwIn as an example

A table of variables is shown below which corresponds to the communication relationships displayed above. It is also created with TSwIn. The communication relationships are entered into the right column of the table.

	Variablenname	Adresse	Komm.-bez.:
0	Var1	B 0:1.3	1
1	Var2	B 100:10.12	5
2	Var3	B 100:10.12	1 / 2
3	Var4	BY 1:35	1
4	Var5	BY 10:78	1
5	Var6	W 11:24	1
6	Var7	W 100:102	5
7	Var8	W 130:12	1 / 2
8	Var9	DW 100:102	5
9	Var10	DW 130:20	1 / 2
10	Var11	DW 50:25	6
11	Var12	DW 200:20	1 / 4

Fig.11: Variable list in TSwIn as an example

The following applies to the communication relationship 1:

- Index 0 - 99 byte addressed
- Index 100 - 199 word addressed
- Index 200 -300 double-word addressed.

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var1	Bit	0	1	3	1	

Bit access: Bit 3 of index 0, subindex 1 of communication relationship 1.

If no address type is entered, the default value in the communication relationship definition table applies, i. e. for index 0 it is byte address.

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var3	Bit	100	10	12	1	2

Bit access: Bit 12 of index 100, subindex 10 of communication relationship 1.

Index 100 in the communication relationship 1 is a word address, but the default setting is byte address. Therefore, a 2 must be entered for address type.

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var2	Bit	100	10	12	5	

Bit access: Bit 12 of index 100, subindex 10 of communication relationship 5.

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var5	Byte	10	10	78	1	

Byte access: Index 10, subindex 78, communication relationship 1

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var8	Word	130	12		1	2

Word access: Index 130, subindex 12, communication relationship 1

Index 100- 199 in comm. relationship 1 is a word address, but the default setting is byte address. Therefore, a 2 must be entered for address type.

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var12	DoubleWord	200	20	3	1	4

Double-word access: Index 200, subindex 20, communication relationship 1

Index 200- 300 in comm. relationship 1 is a double-word address, but the default setting is byte address. Therefore, a 4 must be entered for address type.

Variable Name	Access	Index	Subindex	Bit Number	Comm.-Rel.	Address Type
Var11	DoubleWord	50	25		6	

Double-word access: Index 50, subindex 25, communication relationship 6

Index

A

Authorization levels 20

C

CAL..... 135
CANopen 135
CiA 135
Communication module 8
Configuration mask 97

D

Definition format..... 121
DIP switch S4..... 129, 132
Download 131
Download cable..... 129

E

Error messages 130
 DATASET STORAGE FAILURE 95
 FIRMWARE NOT CONFORM 95
EUROTHERM-Definitions..... 145
External data release 75

F

Firmware 128
Formatted output of variables 31
Formula for scaling input values 41
Formula for scaling the output of variables 32

H

Help key 69
Help text..... 10
 for input variables 42
How to operate TSdos..... 125

I

Input variable
 Counter..... 41
 Timer..... 41

K

Key
 Data Release..... 69
Key functions
 editor for coded text 72
 in the I/O mask 27
 in the message mask 28
 in the node mask..... 26
 in the status message mask 29
Key functions in the alphanumeric editor 72
Key functions in the BCD-number editor 71
Key functions in the hexadecimal editor 71
Key functions in the numerical editor 70

M

Marginal conditions 115
Mask number 111
Mask types 23
Master password 21
Message mask 97
Mode selector switch
 Standard Mode 16

N

Number of languages 126

O

One-time output variables/cyclic output variables 30

P

Parallel outputs 7
Password 10, 20
Password editor..... 72
Password mask
 system mask 24
Password protection 10
 access level 20
 authorization level 20
 setup mask 21
 startup mask 21
 view level 20
Plausibility 69
Plausibility check
 input variable 42
Post-decimal places 31
Protocol-specific variable list 120

R

Range of values..... 30
Refresh intervals 128
Representation of output variables 32
Representation with leading zeros 33
Response times 7
Running time meter
 reset byte 110

S

Scaled output of variables 31
Status LED
 data release 75
Symbolic name 111
System masks
 main mask 24
 setup mask 24
 startup mask 24
system message
 illegal data set 91
System message
 (floating point) number invalid 91

data set active	91	editors	
data set download	91	EditEnter	68
data set file error	91	EditInvers	67
data set format	91	StatePerm	68
data set memory full	91	error statistics interface X2	
data set protected	91	ComErrorTab	47
data set transfer	91	ComFrameCount	47
data set unknown	91	ComOverrunCount	47
editing mode active	91	ComParityCount	47
interface in use	90	ComStatisticsTab	47
invalid mask no	90	ComSubcodeTab	48
invalid message no	90	help	
invalid password	90	Message	68
loop-through active	91	QuitMessage	68
message buffer full	90	StateHelp	68
message overflow	90	loadable font	
new message	90	ChrsetName	66
no data set address	91	loop-through operation	
overvoltage	90	Pg2Sps	66
password missing	91	Pg2SpsState	66
password unchanged	90	maintenance	
print log invalid	90	Break	67
recipe unknown	91	LCDADCInput	67
replace battery	90	LCDADCOutput	67
value too large	89	User1	67
value too small	90	User2	67
wrong format	89	User3	67
System numbers	88	User4	67
System variables		User5	67
basic functions		menu control / keys	
Boot	44	HardCopy	56
ComVersion	43	Key0	59
handshake	118	Key1	59
IntEraseEprom	43	Key2	59
LCDBackground	44	Key3	59
LCDBackLight	44	Key4	59
LCDContrast	44	Key5	59
MainVersion	43	Key6	60
OsLanguage	45	Key7	60
TurnOnTemp	45	Key8	60
UserVersion	44	Key9	60
communication area x2		KeyClear	58
ComRetryTimeout	46	KeyCursDown	58
communication area X2		KeyCursLeft	57
ComBaudrateA	46	KeyCursRight	57
ComDataLenA	45	KeyCursUp	58
ComDefaultA	46	KeyDot	58
ComHandshakeA	46	KeyEdit	61
ComParityA	45	KeyEnter	61
ComSlaveNr	47	KeyHelp	58
ComStopBitsA	45	KeyHome	58
ComTimeout	46	KeyMinus	60
communication area X3		KeyPlus	60
ComBaudrateB	48	NewMask	55
ComDataLenB	48	Shift	57
ComDefaultB	49	ShiftCase	57
ComHandshakeB	49	TabLeft	56
ComParityB	48	TabPgDn	56
ComStopBitsB	48	TabPgUp	56

TabRight	56
VarTablenR0	55
VarTablenR1	55
parallel message system	
RepmanSortCritP	52
ReputAnzYearP	53
ReputDateP	53
ReputNrP	53
ReputRepTextP	53
ReputRepTextP21	54
ReputRepTextP41	54
ReputRepTextP61	54
ReputTimeP	53
password	
ChangePasswd	61
FlashPasswd	61
MskchgPasswd	61
MskchgResPasswd	61
PasswdInactive	62
password protection	
edit level	20
printer control	
BlockPrint	54
BlockPrintLong	55
PrintAllRep	55
PrintAllState	55
StopPrint	54
real-time clock	
RTCDateFmt	50
RTCDay	49
RTCDayofWeek	50
RTCHour	49
RTCMin	49
RTCMonth	50
RTCsec	49
RTCYear	50
RTCYear2000	50
recipes	
ActDSName	63
DestDSNr	62
DSCopy	62
DSDelete	62
DSDnloadBreak	63
DSDnloadState	63
DSDownload	63
LoadDSName	63
RestoreLineNr	64
RestoreState	64

RezPrintState	65
SaveState	64
SelectDSName	62
SelectDSNr	62
SelectRezeptNr	63
StartRestore	64
StartRezPrint	64
StartSave	64
StartUpload	65
UploadDSNr	65
UploadState	65
running time meter	
Counter1	66
Counter2	66
Counter3	66
Counter4	66
Counter5	66
Counter6	66
Counter7	66
Counter8	66
serial message system	
ClearRepBuf	51
RepmanRepPrint	51
RepmanSortCrit	50
ReputAnzYear	51
ReputDate	51
ReputNr	51
ReputRepText	52
ReputRepText21	52
ReputRepText41	52
ReputRepText61	52
ReputTime	51

U

User data, bit-oriented, CAN	137
User data, byte-oriented, CAN	137
User data, double-word oriented, CAN	137
User data, word-oriented, CAN	137

V

Variable type	
ASCII	30
bit	30
byte	30
Lword	30
word	30

Modification Record

Version	Änderungsgrund	Modification	Kapitel Chapter	Datum Date	Name Name	Bemerkung Comment
V02.50TB98	Ur-Version	<i>Source version</i>		03.12.98	T. Beul	im Eurotherm-Format <i>in Eurotherm design</i>